Fundamental Models and Algorithms for a Distributed Reputation System

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Michael Engler (geb. Kinateder)

aus Donauwörth

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr.-Ing. Dr. h. c. mult. Paul J. Kühn
Tag der mündlichen Prüfung:	17.12.2007

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2007

Acknowledgments

Above all, I want to thank my doctoral adviser Prof. Rothermel, who has enabled me working on this challenging topic in a fascinating three-way collaboration together with Hewlett-Packard research labs in Bristol, UK and the E-solutions division of HP in Böblingen, Germany. His critical questions and guidance contributed significantly to the success of this work. Furthermore, my thanks go to Prof. Dr. Paul J. Kühn for his kind willingness to read and evaluate my dissertation.

The department of distributed systems consists of insightful and motivated researchers. In addition to our sometimes challenging coffee corner meetings, I also enjoyed a lot the collaboration and countless fruitful discussions. I want to especially thank my colleague Timo Heiber for his detailed keen feedback on many of my publications. Additionally, I want to thank the students whose help was invaluable for the success of the UniTEC project, most notably Ralf Terdic, Jochen Widmaier, and Ernesto Baschny.

Many thanks to Martin Sadler and all the researchers at HP labs who kept throwing new ideas around. I really loved the creative atmosphere during my masters thesis and during the following research cooperation. The developers at the HP E-solutions division helped keep my ideas not only flying high but being also grounded in reality. Many thanks for that to Andreas Voith and his team.

I am glad of being able to participate in and contribute to the trust research communities iTrust, Trustcomp, Secoval, and TRECK. The discussions with Audun Jøsang, Liz Gray, Siani Pearson, Jean-Marc Seigneur, Stéphane Lo Presti, Steve Marsh, and many others really helped me on my way.

I cannot thank my parents enough for their continuous support, encouragement, and patience during my studies and even more during the work on this dissertation. Finally, I want to thank my wonderful wife Sara for supporting me ceaselessly and giving me strength during difficult phases. Without her unfailing optimism and love, this work would not have been possible.

Abstract

With the increased significance of the Internet in our everyday lifes, we embrace its benefits as seemingly unlimited information source, warehouse and general communication medium, but sometimes fall prey to its predators. Outside the online world, social network structures of friends or colleagues allow to identify malicious and reputable entities and to communicate recommendations or warnings accordingly. When interacting through open computer networks, these traditional mechanisms used in the physical world for establishing trust are adapted by reputation systems that allow to build trust in entities and create social network structures on a much larger scale.

In this dissertation, we investigate various models and algorithms required for realizing a fully decentralized reputation system with enhanced privacy properties and fine-grained trust modeling. To ensure the former, we bind trust to virtual identities instead of real identities and present extended destination routing, an approach that allows anonymous communication between pseudonyms without exposing any link to a real identity. To enable the latter, we introduce a generic trust model that allows to model trust in various context areas in addition to expressing context area dependencies that are taken into account when updating trust. The model definition permits incorporating several well-known trust update algorithms from the related work. Subjecting the algorithms to a set of evaluation scenarios gives valuable inputs regarding their specific performance. In order to capture the transitivity of trust, we present algorithms to simplify trust networks and then compute the transitive trust with subjective logic operators. Finally, we propose mechanisms to protect trust by firstly laying its foundation in trusted hardware and secondly ensuring the authenticity of recommendations through the integration of an originality statement.

This reputation system can be utilized by users and relying applications alike to determine the trustworthiness of other entities. While these building blocks are all essential for our system, many contributions can be applied to other reputation systems and even to other research areas as well.

Zusammenfassung

Grundlegende Modelle und Algorithmen für ein Verteiltes Reputationssystem

Einführung

In den zurückliegenden 20 Jahren hat sich das Internet von einem Netz für wenige Spezialisten und Enthusiasten hin zu einem Massenphänomen gewandelt. Diese Entwicklung ist sicher zu einem nicht geringen Teil durch die enorme Verbreitung des Computers begründet, welcher sich mittlerweile nicht nur in Büros sondern auch in den meisten Haushalten befindet. Informationen des Branchenverbandes Bitkom[1] zufolge sind weltweit im Jahr 2007 mehr als eine Milliarde PCs im Einsatz. Ebenso ist die Anzahl der Internetnutzer auf über eine Milliarde angewachsen. In Deutschland war im Jahr 2006 in 77 von 100 Haushalten ein PC vorhanden, der EU Durchschnitt lag bei 60. Es gibt vielfältige Gründe für die Nutzung des Internets; typische Verwendungszwecke liegen in den Bereichen Informationsrecherche, Kommunikation oder aber geschäftlicher Transaktionen.

Es kann kein Zweifel daran bestehen, dass das Internet eine unerschöpfliche Quelle an Informationen in den unterschiedlichsten Bereichen ist: Angefangen bei relativ unkritischen, z.B. freizeitbezogenen Informationen, über sensiblere Daten wie Produktbeurteilungen, bis hin zu hochkritischen beispielsweise börsenbezogenen Informationen. Ein Rechercheur benötigt korrekte Informationen für die Suche. Da die Qualität der angebotenen Information jedoch stark variiert, stellt sich die Frage nach den Kriterien,

[1] http://www.bitkom.org

welche es erlauben, die Vertrauenswürdigkeit (trustworthiness[2]) der Information bzw. des Herausgebers der Information einzuschätzen. Dies gilt ebenso für die Einschätzung der Vertrauenswürdigkeit eines Kommunikationspartners und noch in stärkerem Maße für die eines Geschäftspartners.

In der realen Welt verlassen wir uns auf soziale Netze von Freunden, Kollegen, etc., von welchen wir Rat über die Vertrauenswürdigkeit von Informationsgebern, Kommunikations- oder Geschäftspartnern einholen. In der virtuellen Umgebung des Internets ist eine Repräsentation des Vertrauenskonzepts (trust) als zentrale Komponente von erfolgreichen Interaktionen im sozialen wie geschäftlichen Bereich gleichermaßen für den Aufbau sozialer Netze vonnöten.

Die Forschung im Bereich „Vertrauen“, anfänglich ein Kind der Soziologie und Psychologie, hat mittlerweile auch in der Informatik und Informationstechnik großes Interesse hervorgerufen. Das Verständnis von Vertrauen war lange Zeit ein absolutes Vertrauen, insbesondere im Bereich der Krypto-Algorithmen, in die Aussteller von Identitätszertifikaten. Wenngleich kryptographisch zertifizierte Identitäten eine essentielle Basis für viele Dienste darstellen, so sind sie jedoch nicht ausreichend für die Einschätzung der Vertrauenswürdigkeit einer Entität. Es folgten Arbeiten bezüglich Vertrauen in die Aussteller von Attributszertifikaten im Kontext der *Simple Public Key Infrastructure (SPKI)* und *Simple Distributed Security Infrastructure (SDSI)* [RL96, EFL$^+$99], der Richtlinien- (*policy*) basierten Systeme Keynote und Policymaker von Matt Blaze et al. [BFL96, BFIK99] bis hin zu den aktuellen Arbeiten, welche sich mit der Vertrauenwürdigkeit von Zusicherungen (assertions) im Rahmen der WS-Trust Spezifikation [OAS07], der Liberty Alliance [Lib03] sowie des Security Assertion Markup Language (SAML) Standards [OAS06] beschäftigen.

Steve Marsh legte mit seiner Doktorarbeit [Mar94] die Grundlage für Forschungen im Bereich „Computational Trust“. Er stellt einen formalisierten und kontinuierlichen Vertrauensbegriff vor; Vertrauen also, welches nicht mehr absolut ist, sondern tatsächlich die Vertrauenswürdigkeit einer Entität reflektiert. Dies stellt die Basis für den Bereich der Reputationssysteme (*reputation systems*) dar. Reputationssysteme sammeln, aggregieren und verteilen Rückmeldungen bezüglich des früheren Verhaltens bestimmter Entitäten. Basierend auf einer Auswertung des Feedbacks versuchen Reputationssysteme die Vertrauenswürdigkeit einer Entität einzuschätzen und treffen somit eine Aussa-

[2]In dieser deutschsprachigen Zusammenfassung werden bei Fachbegriffen die entsprechenden englischen Begriffe, welche in der folgenden Arbeit verwendet werden, zur Verdeutlichung in Klammern angeführt.

ge über das wahrscheinlich zu erwartende zukünftige Verhalten. Reputationssysteme können in zentralisierte und verteilte Ansätze klassifiziert werden. Im zentralisierten Fall wird die gesamte Reputationsinformation an einer zentralen Stelle vorgehalten, was komplexe Vertrauensberechnungen ermöglicht, da die gesamte Information verfügbar ist. Dies ermöglicht aber auch die Erstellung detaillierter Nutzerprofile und erfordert, dass alle Teilnehmer dieser zentralen Stelle voll vertrauen, birgt daher entsprechende Risiken. Alternativ zu diesem zentralisiertem Ansatz können die Daten auf mehrere Systeme verteilt werden bis hin zu einem Peer-to-Peer Ansatz, bei welchem die Daten völlig verteilt und von lokalen Reputationssystemagenten ausgewertet werden.

Im Rahmen dieser Arbeit werden in neun Kapiteln Modelle und Algorithmen für ein verteiltes Reputationssystem vorgestellt. Nach der Einführung und Motivation der Arbeit in Kapitel 1 wird in Kapitel 2 der Hintergrund des Themas „Vertrauen“ beleuchtet; zum einen aus Sicht der Sozialwissenschaften und zum anderen aus Sicht der Informatik und Informationstechnik. Zur Validierung der theoretischen Ansätze wurde das System prototypisch implementiert. Kapitel 3 stellt die Ergebnisse einer Anforderungsanalyse sowie die an einem Reputationssystem beteiligten Entitäten, deren Interaktionen sowie die gewählte Architektur des Prototyps vor. Sollte die Plattform eines Agenten durch Angreifer kompromittiert werden, könnte dies den Besitzer des Agenten aber auch alle anderen Teilnehmer des Reputationssystems schädigen. In Kapitel 4 wird ein Ansatz beschrieben und bewertet, welcher die Agenten auf die Grundlage einer *vertrauenswürdigen Plattform (trusted platform)* im Sinne der *Trusted Computing Group (TCG)*[3] stellt. Besonderer Wert wurde auf den Schutz der Privatsphäre (privacy) der Nutzer gelegt, welche durch die Verwendung von Pseudonymen und “Mixe” nach David Chaum [Cha81] gewährleistet wird, wie in Kapitel 5 beschrieben. Kapitel 6 stellt das generische Vertrauensmodell vor, welches die feingranulare Modellierung von Vertrauen unter Berücksichtigung von Kontextwissen erlaubt. In das generische Modell werden verschiedene Vertrauensaktualisierungsalgorithmen eingebettet und mittels verschiedener Szenarien evaluiert. Der Begriff der Transitivität von Vertrauen, die Vereinfachung von Vertrauensnetzen und letztendlich die Berechnung transitiven Vertrauens werden in Kapitel 7 erläutert. Sybil Angriffe (*Sybil attacks*) sind eine Bedrohung für Reputationssysteme. In Kapitel 8 werden drei Ansätze zur Abwehr dieser Angriffe erörtert und evaluiert. Kapitel 9 zieht das Resümee über die Arbeit und verweist in einem Ausblick auf zukünftig mögliche darauf aufbauende Forschungsthemen.

[3] `https://www.trustedcomputinggroup.org/`

Die UniTEC Architektur

Dieser Abschnitt gibt einen Einblick in die Rollen, in welchen UniTEC Benutzer agieren, in die Interaktionen, welche sie hierbei tätigen und welche von ihren Agenten angestoßen werden, sowie in den Aufbau des UniTEC Agenten.

Die vorliegende Arbeit wurde im Kontext des UniTEC[4] Projektes an der Abteilung Verteilte Systeme des Instituts für Parallele und Verteilte Systeme der Fakultät der Universität Stuttgart durchgeführt. UniTEC ist ein Repräsentant der verteilten Reputationssysteme und stellt eine Kombination aus Reputationssystem (*reputation system*) und Empfehlungssystem (*recommendation system*) dar. Das Systemmodell von UniTEC ist *Peer-to-Peer* basiert. Um an dem verteilten System teilzunehmen, führt jeder Benutzer auf einem PC einen UniTEC Agenten im eigenen Benutzerkontext aus. Dieser Agent kommuniziert mit den Agenten anderer Benutzer auf Geheiß seines Besitzers hin, arbeitet jedoch auch selbständig, um den Betrieb des Systems sicherzustellen.

Rollen und Interaktionen

Verwender des Systems agieren in einer der beiden folgenden Rollen: als Aussteller von Empfehlungen, also Empfehlende (*recommender*), oder als Anfragende auf der Suche nach Empfehlungen (*requester*). Empfehlungen in UniTEC sind eindeutig einem Kontextgebiet (*context area*) zugeordnet, sind digital signiert und enthalten 3 Hauptkomponenten:

1. *Empfehlungsdaten:* Neben dem Kontextgebiet der Empfehlung kann diese Komponente eine Vielzahl unterschiedlicher Felder enthalten und unterstützt im UniTEC Prototyp die folgenden Feldtypen: Boolean (z.B. `Empfehlenswert?`: ja-nein), Integer und Float (z.B. `Qualität`: 0,8%), Binäre Objekte (z.B. Bilder), sowie Strings beliebiger Länge für textuelle Bewertungen.

2. *Ausstellerdaten:* Insbesondere wird hier die Identität des Ausstellers der Empfehlung hinzugefügt. Zusätzlich kann der Aussteller die Zuversicht in die eigene Empfehlung einbinden.

[4]Das Akronym UniTEC steht für "**Uni**versal **T**rust Architecture for **E**lectronic **C**ommerce". Offensichlicherweise bietet UniTEC nicht für alle Probleme des Elektronischen Handels eine adäquate Lösung. Dennoch bietet das System einige herausragende Merkmale, wie beispielsweise die feingranulare Vertrauensmodellierung unter Einbeziehung von Kontextwissen oder den Schutz der Privatsphäre, welche die Systeme der Verwandten Arbeiten nicht in diesem Maße offerieren.

3. *Metadaten:* Die Metadaten enthalten eine eindeutige Empfehlungskennung, einen Zeitstempel der Erstellungszeit der Empfehlung, sowie eine Zugriffskontrollliste (*access control list, ACL*). Letztere wird beim Zugriff auf die Empfehlung ausgewertet und ermöglicht es, diesen Zugriff Anfragenden zu gewähren oder zu verwehren.

Aussteller von Empfehlungen publizieren diese jeweils auf ihren lokalen UniTEC Agenten. Die Anzahl verfügbarer Empfehlungen dient als Metrik für das Wissen bzw. die Expertise (*knowledge*), welche der Aussteller in dem jeweiligen Kontextgebiet zu haben scheint. Diese Information wird an andere Agenten über entsprechende Benachrichtigungen versandt.

Anfragende stellen eine Suchanfrage an ihre lokalen Agenten, welche diese an eine Gruppe von geeignet erscheinenden Agenten weiterleiten. Diese Gruppe, auch Nachbarschaft (*neighborhood*) genannt, enthält Empfehlende dreier überlappender Klassen: Empfehlende mit hoher Expertise, wie bereits beschrieben, Empfehlende, denen im Kontextgebiet der Anfrage vertraut wird, und einem Teil zufälliger Empfehlender.

Die Anfragenachricht beinhaltet unter anderem eine eindeutige Anfragekennung zur Identifizierung der Anfrage, einen Suchstring bestehend aus Kontextgebiet sowie Anfragefilter zur Spezifikation der Anfrage, die Identität des Anfragenden und letztendlich die sogenannte Vertrauenskette (*trust chain*). Kann ein Agent die Anfrage beantworten, schickt er die passenden Empfehlungen zusammen mit der momentanen Vertrauenskette direkt an den Anfragenden zurück. Des Weiteren leitet ein Agent die Anfrage unter bestimmten Umständen auch an die eigene Nachbarschaft im passenden Kontextgebiet weiter und fügt der Vertrauenskette das eigene Vertrauen in die jeweiligen Nachbarschaftsmitglieder bei. Die einzelnen signierten Teilglieder der Vertrauenskette einer Nachricht enthalten somit das Vertrauen des Anfragenden in den ersten Empfänger, das Vertrauen des ersten Empfängers in den zweiten Empfänger u.s.w.. Diese Ausbreitung der Anfragenachricht in den Nachbarschaften wird unter anderem durch einen Zähler in der Nachricht, welcher die maximale Entfernung der Anfragenachricht vom Anfragenden begrenzt, terminiert.

Der Anfragende erhält in Antwortnachrichten die angefragten Empfehlungen der Empfehlenden jeweils mit der passenden Vertrauenskette. Der UniTEC Prototyp enthält einen Algorithmus, um aus den Vertrauensketten das transitive Vertrauen des Anfragenden in die Empfehlenden zu berechnen. Kapitel 7 stellt weitergehende Mechanismen für die Berechnung transitiven Vertrauens vor.

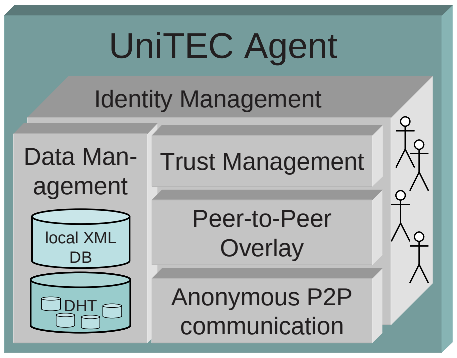


Abbildung 1: Architektur des UniTEC Agenten

Ein essentieller Schritt für das Lernen, für die Aktualisierung des Vertrauens, ist nun die Bewertung (*feedback*) des Anfragenden an den UniTEC Agenten bezüglich der Qualität der empfangenen Empfehlungen. Diese Bewertung wird im Agenten gespeichert, und führt gemäß des gewählten Vertrauensaktualisierungsalgorithmuses (*trust update algorithm*) zu einer Stärkung des Vertrauens im Falle positiven Feedbacks, bzw. zu einer Abschwächung bei negativem Feedback.

Aufbau des UniTEC Agenten

Wie aus Abb. 1 ersichtlich besteht der UniTEC Agent aus fünf Hauptkomponenten:

Die Speicherungskomponente (*data management component*) bietet ein Framework für das Ablegen von und den kontrollierten Zugriff auf sowohl lokal für den Benutzer relevante Information als auch Information, welche für den Betrieb des UniTEC Systems an sich benötigt wird. Die lokalen Daten des Benutzers werden in der *XML Datenbank Xindice* gespeichert. Hierzu gehören insbesondere Information über die eigene und fremde Expertise, über das Vertrauen in fremde Empfehlende, sowie die Empfehlungen selbst. Die für den Systembetrieb benötigten Daten werden größtenteils im Speicher gehalten und beruhen auf einer Implementierung der verteilten Hashtabelle Chord [SMK$^+$01] (*distributed hashtable, DHT*).

Um die Privatsphäre der Nutzer zu schützen, interagieren sowohl Aussteller als auch Anfragende nicht unter der realen Identität, sondern verwenden virtuelle Identitäten, auch Pseudonyme (*pseudonyms*) genannt. Die Identitätsmanagementkomponente (*identity management component*) des UniTEC Agenten verwaltet sowohl die eigenen Pseudonyme und erlaubt deren Zuordnung zu Kontextgebieten, als auch fremde Pseudonyme, über welche im Laufe der Interaktionen Informationen gesammelt wurden.

Um zu verhindern, dass beim Weiterleiten von Anfragen bzw. Antworten auf Anfragen die Verbindung zwischen eigenen Pseudonymen und der echten Identität aufgedeckt werden kann, enthält die Kommunikationskomponente (*anonymous peer-to-peer communication component*) Mechanismen zur anonymisierten Kommunikation zwischen Pseudonymen. Diese Mechanismen werden von der darauf aufsetzenden *Peer-to-Peer Overlay Komponente* bei der Durchführung des kurz vorgestellten Algorithmus zur Weiterleitung von Anfragen an die entsprechende Nachbarschaft für das jeweilige Kontextgebiet verwendet.

Die letzte der fünf Komponenten dient der Verwaltung des Vertrauens (*trust management component*) und leistet Beiträge in 3 Teilbereichen: Zum einen ermöglicht diese Komponente die Auswertung von Vertrauensketten zu transitivem Vertrauen beim Empfang von Empfehlungen. Des Weiteren wird das Vertrauen des Anfragenden in die Empfehlenden aktualisiert, nachdem das Feedback über die Empfehlungsqualität erhalten wurde. Letztendlich wird die Expertise der Empfehlenden aktualisiert, und zwar sowohl beim Empfang entsprechender Expertisebenachrichtigungen als auch beim Empfang von Empfehlungen.

Identitätsmanagement und Datenschutzaspekte

Nachdem im letzten Abschnitt die Gesamtarchitektur von UniTEC vorgestellt wurde, steht hier nun die Funktionalitäten der Identitätsmanagementkomponente und der Kommunikationskomponente im Vordergrund.

Wie bereits angedeutet interagieren Benutzer in UniTEC nicht direkt miteinander, sondern über virtuelle Identitäten. Bei der Generierung eines Pseudonyms in der Identitätsmanagementkomponente wird ein Schlüsselpaar, bestehend aus einem öffentlichen und einem privaten Schlüssel, sowie eine Pseudonymkennung zur Identifikation des Pseudonyms erstellt. Die Schlüssel werden unter anderem zur Geheimhaltung der Kommunikation mit dem Pseudonym, sowie zur Signierung der Empfehlungen, welche im Namen dieses Pseudonyms veröffentlich werden, verwendet. Als Kennung dient der SHA-1 Hash des öffentlichen Schlüssels. Durch die Peer-to-Peer basierte Architektur von UniTEC kommt für die Veröffentlichung der öffentlichen Schlüssel kein zentraler Verzeichnisdienst (*directory*) in Frage. Anstelle dessen wurde ein verteilter Ansatz gewählt, welcher auf der verteilten Hashtabelle Chord basiert. Jeder UniTEC Agent nimmt automatisch am *Chord Ring* teil und speichert somit einen Anteil der Daten des gesamten Systems.

Die Herausforderung bei der Verwendung von Pseudonymen liegt nun darin, sicherzustellen, dass die Verbindung zwischen realer Identität und ihrer virtuellen Identitäten nicht aufgedeckt werden kann. Der im Rahmen dieser Arbeit entwickelte Ansatz namens *Extended Destination Routing (EDR)* basiert auf David Chaums *Mix Konzept*[5]. In der Kommunikationskomponente des UniTEC Agenten ist solch ein Mix Dienst vorhanden, welcher bei seiner Aktivierung ein Schlüsselpaar für die Mix-Tätigkeit erzeugt. EDR sieht vor, dass für jedes Pseudonym Datenstrukturen für die Wegsuche (*routing header*) erstellt werden, ähnlich der Zwiebel im *Onion Routing*, wobei sich hier im Inneren der Struktur die IP Adresse des UniTEC Agenten des Pseudonyms befindet. Um diese IP Adresse herum werden die Verschlüsselungsschichten für die Mixe mitsamt deren IP Adressen eingefügt. Für jedes Pseudonym registriert der Agent neben dem öffentlichen Schlüssel auch diese Datenstrukturen in der verteilten Hashtabelle. In Kapitel 5 werden die Protokolle zum Registrieren der Routing Header, zum Entdecken von Mixen, sowie zum Senden einer Nachricht zwischen Pseudonymen ausführlich erläutert und evaluiert.

Modellierung und Aktualisierung von Vertrauen

Ein essentieller Bestandteil eines Reputationssystems ist das Vertrauensmodell (*trust model*) welches beschreibt, wie Vertrauen in einem digitalen System repräsentiert werden kann. Der dazugehörige Vertrauensaktualisierungsalgorithmus (*trust update algorithm*) gibt vor, wie dieses Vertrauen beim Eintreten gewisser Ereignisse aktualisiert wird.

In der Vertrauensforschung wurden bereits zahlreiche Vertrauensmodelle und Aktualisierungsalgorithmen mit den unterschiedlichsten Eigenschaften vorgestellt, wobei jedes bzw. jeder für eine bestimmte Benutzergruppe besonders geeignet erscheint. Durch

[5] Anstatt eine Nachricht direkt von einem Sender an einen Empfänger zu senden, wird die Nachricht verschlüsselt an einen Mix gesendet, welcher sie dann an den Empfänger weiterleitet. Der Mix leitet Nachrichten jedoch nicht direkt weiter, sondern sammelt erst eine gewisse Anzahl von Nachrichten, mischt diese bezüglich der Reihenfolge und sendet sie dann an die jeweiligen Empfänger. Mechanismen wie feste Nachrichtengrößen sowie Blindnachrichten stellen sicher, dass ein Angreifer zwar sehen kann, dass eine Sendergruppe mit einer Empfängergruppe kommuniziert, nicht aber, welches Senderindividuum an welches Empfängerindividuum sendet. Um den Schutz vor nicht-vertrauenswürdigen Mixen zu erhöhen, kann eine Kaskade von Mixen (mix cascade) verwendet werden. Die Struktur, welche an den ersten Mix gesandt wird, hat die Form einer Zwiebel mit zahlreichen Verschlüsselungsschichten und der eigentlichen Nachricht im Inneren. Jeder Mix kann eine Schicht der Zwiebel entfernen und findet so das nächste Ziel heraus, an welches die Nachricht weitergesandt werden muss.

die Subjektivität von Vertrauen sollte dem UniTEC Benutzer freigestellt werden, den für die eigenen Zwecke am besten geeigneten Algorithmus auszuwählen. Für die verteilten Algorithmen, etwa der Anfrageverarbeitung mit den in der Anfrage enthaltenen Vertrauensketten, ist aber ein gemeinsames Verständnis von Vertrauen notwendig. Unser Beitrag liefert ein *generisches Vertrauensmodell*, welches es erlaubt, zahlreiche existierende Vertrauensmodelle und Vertrauensaktualisierungsalgorithmen in das UniTEC System einzubetten. Hierdurch wird eine Vergleichbarkeit der Algorithmen anhand verschiedener Szenarien ermöglicht

Die folgenden Dimensionen einer Vertrauensbeziehung wurden identifiziert und durch entsprechende Metriken realisiert: Das Vertrauensmaß (*trust measure*) quantifiziert eine Vertrauensbeziehung und reicht von völligem Misstrauen bis hin zu totalem Vertrauen. Die Vertrauensgewissheit (*trust certainty*) beschreibt, mit welcher Wahrscheinlichkeit die Vertrauensmetrik zutreffend ist. Beispielsweise ist die Gewissheit nach nur einer Erfahrung im Allgemeinen gering und steigt mit der Anzahl gleichartiger Erfahrungen an. Der Vertrauenskontext[6] (*context*) repräsentiert den Bereich, in welchem vertraut wird. So ist beispielsweise anzunehmen, dass man einer Person zwar durchaus vertrauen kann, das eigene Kind zu betreuen (Kontext 1), nicht aber das Auto zu reparieren (Kontext 2). Die „Vertrauensdirektheit“ (*trust directness*) unterscheidet zwischen funktionalem Vertrauen (*functional trust*) und Referenzvertrauen (*referral trust*). Funktionales Vertrauen beschreibt das Vertrauen in eine Person im Kontext selbst, also beispielsweise das eigene Kind zu betreuen. Referenzvertrauen wiederum beleuchtet das Vertrauen in eine Person, an andere Experten im Kontext zu verweisen, also andere Babysitter empfehlen zu können. Letztlich ist die Dimension der Vertrauensdynamik (*trust dynamics*) zu nennen, welche die Änderung des Vertrauens über die Zeit hinweg abbildet. Verschiedene Aspekte haben Einfluss auf diese Dynamik, beispielsweise die eigene Erfahrung (*experience*) mit einer anderen Entität realisiert durch das Feedback nach einer erhaltenen Erfahrung, die Zuversicht des Ausstellers (*recommender confidence*) in die eigene Empfehlung, der quantifizierte Nutzen (*utility*) der Transaktion, welche bewertet wurde, die Abhängigkeit des zu aktualisierenden Kontextes zu dem Kontext in welchem die Erfahrung gemacht wurde.

[6] UniTEC unterstützt nicht nur verschiedene Kontexte, sondern insbesondere auch Abhängigkeiten der Kontexte untereinander. Es wurde das Konzept eines gewichteten gerichteten Graphen verwendet, um Kontextabhängigkeiten abzubilden. Dieses erlaubt es, eine Vertrauensaktualisierung in einem Kontext auch in den verwandten Kontexten in abgeschwächter Form durchzuführen. Letztlich wird so ermöglicht, dass durch das Wissen über die Kontextabhängigkeiten auch bereits nach wenigen Erfahrungen eine gute Einschätzung der Vertrauenswürdigkeit einer Person möglich ist.

Um die Anwendbarkeit des entwickelten Modells aufzuzeigen, wurden 4 Vertrauensaktualisierungsalgorithmen in das generische Vertrauensmodell eingebettet: die vielzitierte Arbeit von Alfarez Abdul-Rahman und Stephen Hailes [ARH00], Audun Jøsangs Beta Reputation System [Jøs02], das ReGreT System von Jordi Sabater [Sab03] sowie der UniTEC Vertrauensaktualisierungsalgorithmus basierend auf geometrischem Lernen [KR03]. Zur Bewertung der Qualität dieser vier Algorithmen wurde ein Set an Testszenarien definiert, welche jeweils aus einer Aneinanderreihung von Feedbacks bestehen. Die Testdurchführung erlaubte nun erstmalig eine Vergleichbarkeit der Algorithmen und zeigte deren Eigenheiten, Stärken und Schwächen. Zusammenfassend kann gesagt werden, dass Abdul-Rahman-Hailes an der diskreten 4-Wert-Metrik leidet und ReGret stark von der Feedback-Historiengröße abhängig ist. UniTEC ist ein einfacher und doch effizienter Algorithmus, welcher jedoch nur den aktuellen Vertrauenswert und die letzte Erfahrung berücksichtigt. Der Algorithmus des Beta Reputation Systems bietet insgesamt die ausgewogensten Resultate.

Transitivität von Vertrauen

Wie bereits in Absatz zur UniTEC Architektur festgestellt, empfängt der Agent des Anfragenden die Antworten auf die Anfrage inklusive der Vertrauensketten. Bedingt durch den Algorithmus zur Anfrageweiterleitung ist es auch möglich, dieselbe Empfehlung über verschiedene Pfade, also mit verschiedenen Vertrauensketten, zu erhalten. In diesem Abschnitt wird nun die Frage erörtert, wie sich das transitive Vertrauen eines Anfragenden in einen Empfehlenden aus einer Vielzahl einzelner Vertrauensaussagen bestimmen lässt.

Ein einfacher Algorithmus wurde für diese Aufgabe im Rahmen des UniTEC Prototyps entwickelt. Vertrauen wird als reale Zahl im Intervall $[0,1]$ modelliert, wobei 0 absolutes Misstrauen und 1 absolutes Vertrauen darstellt. Das transitive Vertrauen einer einzelnen Vertrauenskette wird als Produkt des Vertrauens der einzelnen Teilglieder berechnet. Das finale transitive Vertrauen des Anfragenden in den Empfehlenden ist das Maximum dieser Produkte. Wenngleich dieser Algorithmus einfach und verständlich erscheint, so kann man durch dessen optimistische Natur, bedingt durch die Wahl des Maximums, eine Verfälschung des Vertrauens feststellen. Dies soll am folgenden Beispiel verdeutlicht werden: Anfragender A fragt seinen besten Freund B (maximales Vertrauen) und den vagen Bekannten C (leichtes Vertrauen) nach deren Vertrauen in D. Angenommen B lehnt nun D komplett ab (maximales Misstrauen), C hingegen gibt eine stark positive Aussage (maximales Vertrauen) hinsichtlich D ab. Der Algorithmus würde nun den Pfad über B und somit den Rat des besten Freundes ignorieren und den Pfad über C verwenden.

Ein komplexer Algorithmus zur Analyse von Vertrauensnetzen mittels subjektiver Logik (trust network analysis using subjective logic, TNA-SL) wurde in Zusammenarbeit mit *Prof. Audun Jøsang* von der Queensland University of Technology in Brisbane, Australien sowie *Elizabeth Gray* vom Trinity College Dublin in Irland entwickelt. Dieser Algorithmus besteht aus 3 Teilen: Zum einen wird eine Beschreibungssprache für Vertrauensgrafen definiert mittels derer Graphen in einer bestimmten Form[7] beschrieben werden. Des Weiteren wird ein Algorithmus vorgestellt, welcher Graphen schrittweise in diese Form überführt. Letztendlich wird beschrieben, wie ein Vertrauensnetz in dieser Form mittels subjektiver Logik ausgewertet und das resultierende transitive Vertrauen berechnet wird.

Abwehr von Sybil Angriffen

Der Begriff der Sybil Angriffe (*Sybil attacks*) wurde geprägt durch John Douceur [Dou02]. Douceur beschreibt, dass Angreifer in einem System, in welchem virtuelle Identitäten ohne zentrale Zertifizierungsstelle erstellt werden, eine beliebige Anzahl von Pseudonymen erstellen und somit die ehrlichen Teilnehmer überrennen können. In diesem Abschnitt werden die Schutzmechanismen des UniTEC Systems gegen diesen Angriffstyp vorgestellt.

In Kapitel 4 wird erörtert, wie vertrauenswürdige Hardware (trusted platforms) verwendet werden kann, um den Schutz vor Angriffen gegen den UniTEC Agenten selbst zu stärken. Eine Eigenschaft von vertrauenswürdigen Plattformen ist das Ausstellen von Pseudonymen über eine sogenannte Datenschutzzertifizierungsstelle. Durch die Kontrolle der Pseudonymausstellung werden Sybil Angriffe effektiv verhindert.

Des Weiteren wurde ein Konzept entwickelt, um durch die Kombination eines Zahlungssystems mit dem UniTEC Reputationssystem während des Zahlungsvorgangs ein Originalitätsmerkmal (originality statement) unter besonderer Berücksichtigung der Datenschutzaspekte zu erstellen. Dieses Originalitätsmerkmal ist über entsprechende *Hashwerte* an die Empfehlungskennung und den Empfehlungsinhalt gebunden und

[7]Mehr Details hierzu finden sich in Kapitel 7.5.

wird von der Clearingstelle des Zahlungssystemanbieters digital signiert. Die Clearingstelle wird ein Originalitätsmerkmal nur einmal pro Zahlungstransaktion digital signieren. Anschließend kann dieses Originalitätsmerkmal vom Empfehlenden der Empfehlung beigefügt werden. Eine Erweiterung des Konzeptes beinhaltet zusätzlich das Einfügen des Transaktionswertes in das Originalitätsmerkmal. Der Anfragende kann nun beim Erhalt einer Empfehlung am Vorhandensein des Originalitätsmerkmals erkennen, dass der Empfehlende tatsächlich eine Transaktion in gewisser Höhe getätigt hat. Dieses Verfahren schützt effektiv gegen Sybil Angriffe. Zwar kann ein Angreifer immer noch beliebig viele Pseudonyme erstellen, die Vertrauensmechanismen in UniTEC sichern jedoch, dass der Angreifer reputable Pseudonyme verwenden muss, um den Opfern der Angriffe falsche Empfehlungen zu präsentieren. Das Steigern der Reputation dieser Pseudonyme erfordert nun aber finanzielle Transaktionen des Angreifers und wird somit insbesondere durch die Berücksichtigung des Transaktionswertes bei der Vertrauensaktualisierung unrentabel. Generell erhält Vertrauen durch die Integration von Zahlungssystemmechanismen einen monetären Gegenwert.

Resümee

Im Rahmen dieser Arbeit wurden Modelle und Algorithmen entwickelt, um ein verteiltes Reputationssystem zu realisieren. Diese Dissertation leistet folgende Beiträge zum Stand der Wissenschaft:

- Das vorgestellte Reputationssystem folgt einer völlig *verteilten Architektur* ohne zentrale Komponenten.
- Das *generische Vertrauensmodell* erlaubt die feingranulare Modellierung von Vertrauen unter Berücksichtigung zahlreicher Dimensionen, insbesondere der Unterstützung von Kontextabhängigkeiten.
- Die Generik des Modells erlaubt die *Einbettung verschiedener existierender Vertrauensaktualisierungsalgorithmen*. Hierdurch konnten diese Algorithmen erstmalig zueinander in Relation gestellt und gemeinsam evaluiert werden.
- Neue Mechanismen zur Berechnung *transitiven Vertrauens* wurden entwickelt.
- Das Reputationssystem ist gegen Sybil Angriffe durch die *Kombination aus Zahlungssystem und Reputationssystem* geschützt.

- Alle vorgestellten Algorithmen legen besonderen Wert auf den *Schutz der Privatsphäre* aller Nutzer. Insbesondere wurde ein Ansatz zur *anonymisierten Kommunikation zwischen Pseudonymen* im UniTEC System entwickelt und evaluiert.

Die wissenschaftlichen Ergebnisse [KR03, KP03, JGK03, KR04, KTR05a, KBR05, KTR05b, JGK06] wurden in internationalen Zeitschriften und auf Konferenzen im Gebiet der Vertrauensforschung publiziert. Des Weiteren wurde anhand eines Prototyps des UniTEC Agenten die Machbarkeit der vorgestellten Ansätze aufgezeigt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The phenomenal growth of the Internet that we experienced during the last couple of decades, together with the fact that computers can be found not only in business environments but also in most households to the point of being a commodity nowadays, led to a widespread public acceptance of this communication medium. There are numerous reasons why people "go online", amongst which getting *access to information*, *communicating with people* and *doing business* are the most common usage scenarios. However, the Internet has some perils in store for the ingenious user and solely relying on good faith during online interactions may lead to unpleasant surprises.

There is no doubt that the Internet offers enormous amounts of *information* in all kinds of different areas, ranging from purely leisure-relevant and possibly dispensable information, like who is currently number one in the US-single-charts, to more critical areas, like product reviews or even stock exchange data. Since the quality of the available information varies, we require *correct* information especially in these critical areas. However, what are the criteria that enable us to decide, whether a certain information provider is *trustworthy* or not?

Modern communication media in general and the Internet in particular are increasingly removing us from familiar styles of interacting and conducting *business* in ways that traditionally rely on some degree of pre-established trust between business partners. Moreover, most traditional cues for assessing trust in the physical world are not available through those media. We may now be conducting business with people and organizations of which we know nothing and are faced with the difficult task of making decisions involving risk in such situations.

Outside the online world, we rely on social network structures of friends, colleagues etc. to find trustworthy information providers, communication and business partners. In the virtual environment of the Internet, a realization of the concept of trust as an essential component of successful interactions in social life as well as in business relationships is required as a central element for building social network structures.

The topic of trust in open computer networks has received considerable attention in the network security community and e-commerce industry. Important work stretches from the first discussion of trust as a computational concept in the noteworthy dissertation of Steve Marsh [Mar94], to trust in attribute certificates mentioned in the in *Simple Public Key Infrastructure* (SPKI) and *Simple Distributed Security Infrastructure* (SDSI) context [RL96, EFL$^+$99] and policy based trust in the Keynote and Policymaker work of Blaze et al. [BFL96, BFIK99].

State-of-the-art technology for stimulating trust in e-commerce includes cryptographic security mechanisms for providing confidentiality of communication and authentication of identities. However, while having a cryptographically certified identity or secure communication via an encrypted channel is fundamental, it is not enough for making informed decisions if no other knowledge about a remote transaction partner is available. Trust therefore also applies to the truthfulness of specific claims made by parties who request services in a given business context as described in the standardized Web Services Trust Language (WS-Trust) [OAS07], and trust between business partners regarding security assertions as described in the Liberty Alliance standard [Lib03] and its successor SAML2.0 [OAS06] for single sign-on purposes.

Trust also applies to the honesty, reputation and reliability of service providers or transaction partners, in general or for a specific purpose. In this context, the process of assessing trust becomes part of *quality of service* (QoS) evaluation, decision making and risk analysis. *Reputation systems* model these structures up to a certain degree supporting users in their decision whom to trust and whom to avoid. The goal of these systems is to minimize the risk of interactions with strangers.

The reputation data can be stored at a centralized service, which allows rich trust computation due to all relevant data being available. However, storing all information at a central service requires high trust of all participants in that service and raises privacy concerns because of many valuable detailed user profiles are stored at a single party. An alternative to such a centralized reputation storage is to distribute the information in a peer-to-peer fashion over multiple nodes with local reputation agents. Whereas many results have been published already in the area of centralized reputation systems, research in distributed reputation systems has only gained significant momentum during the last years.

1.2 Problem Statement

The *goal of our research* is to develop models and algorithms for a reputation system that fulfill the following set of criteria:

- The approach must enable a *distributed* reputation system, in other words it may not rely on any centralized entities.
- The developed algorithms must be *fault-tolerant*. The failure or takeover of any node must not lead to the failure of the whole system. In addition, the mechanisms must be stable enough to ensure that malicious entities cannot easily overwhelm honest entities in the system.
- The *modeling of trust* must be *fine-granular* such that a detailed assessment of the trustworthiness of the entities is possible.
- The *protection* of the participants' *privacy* is of utmost importance.

1.3 Published Results

The results presented in this dissertation represent the insights gained during the course of the UniTEC[1] research project. This project was conducted by the author under the guidance of Prof. Kurt Rothermel as head of the Distributed Systems Department at the Institute of Parallel and Distributed Systems of Universität Stuttgart, Germany.

Large parts of the scientific results made in this dissertations were published as full papers in international conferences [KR03, KP03, KR04, KTR05a, KBR05], a refereed

[1] The acronym UniTEC stands for "**Uni**versal **T**rust Architecture for **E**lectronic **C**ommerce". Admittedly, UniTEC does not save the world, as the name implies, but nevertheless provides several properties especially with regards to fine-grained trust modeling and user privacy protection that none of the reputation systems of the related work can provide to this degree.

workshop [JGK03] and international journals [KTR05b, JGK06]. In addition to these publications, a prototype of UniTEC was developed, which allowed us to experiment and show the feasibility of the theoretical concepts.

We present in the following the structure of this dissertation.

1.4 Outline

In Chapter 2, we introduce the *topic of trust* from two different directions. Firstly, we discuss the insights from the area of *social science* research, since these served as valuable inputs for our design of the trust model. Secondly, trust research results are presented from an *information technology* point of view, including trust in cryptographic keys, trust in platforms and trust in agents respectively reputation systems.

Chapter 3 outlines the *functionality* of the *prototype* that we developed during the course of the UniTEC project. UniTEC was designed as a distributed reputation system that allows users to publish data items or recommendations, request data items and estimate the trust of a requester in the recommender. Starting therefore with the basic interactions of requesting and publishing recommendations, we derive a set of requirements that the UniTEC prototype has to fulfill and then present the incorporated models, namely the system model, trust model and the model for representing expertise. After that, we break down the basic interactions to a detailed interaction description and finally present the architecture of the developed prototype with its different components.

One danger for UniTEC agents but also for every reputation system lies in *attackers compromising* the *platforms* that the agents reside upon. If a platform has been hacked, the agent's owner may firstly receive wrongful information tampered with by the attacker, secondly an attacker could issue wrongful recommendations on behalf of the owner and thereby at least damage the owner's reputation, and thirdly an attacker could issue wrongful trust statements in the owner's name, also at least damaging the owner's reputation. In Chapter 4, we present an approach based on *trusted platform technology* that allows, among other things, to determine whether a platform is still in a certain safe state.

The *privacy protection* of the participants is a major design goal for UniTEC and in general for our reputation system research. We present in Chapter 5 how we attach trust not to real identities but to virtual identities or pseudonyms. Furthermore, we introduce *extended destination routing* (EDR), our approach for enabling anonymous communication between these pseudonyms. This approach enables us still to build profiles, which are a prerequisite for reputation systems, but prevents linking these profiles to real world identities. We evaluate the security of EDR and the performance of its implementation in the UniTEC prototype. While this concept is particularly useful for UniTEC, it can be applied to other application areas relying on pseudonyms as well.

The related work has put a lot of effort in developing different trust models and associated algorithms to update this trust upon certain events. In Chapter 6 we present our trust model, which supports fine-grained representation of trust in various context areas and takes into account context area dependencies when updating trust as well. This trust model builds on insights from social science research and is generic in a sense that it allows to incorporate different existing trust models into UniTEC. We show how to map different trust update algorithms relying on local models onto the generic UniTEC model and evaluate how these algorithms perform in various scenarios.

If a recommendation has been received directly from a recommender, the trust of the requester in the recommender can be estimated immediately depending on previous experiences with that recommender. However, if the recommendation was passed from the recommender via a chain of intermediaries to a requester, the question arises how to calculate transitive trust. This may become even more complex if a single recommendation can be received via completely different paths. In Chapter 7 we provide firstly a structured notation for trust graphs. Secondly, we describe an approach to determine transitive trust based on network simplification of the trust graph. Finally, we point out how a simplified trust network can be evaluated based on subjective logic.

Due to the fact that UniTEC allows to create an arbitrary number of pseudonyms, it might appear to be prone to the so-called Sybil attack, a prominent attack on many reputation systems. In Chapter 8 we demonstrate how to combine UniTEC with the SET payment system without endangering the strength of the pseudonyms. This approach protects UniTEC from Sybil attacks and is generic in that it can be applied to other payment systems with a dedicated payment gateway as well.

Chapter 9 concludes this dissertation with a summary of its contributions and an outlook on possible future research areas.

Chapter 2

Background

"Trust is not the world's only fundament; however, a very complex but structured concept of this world cannot be constructed without a complex society which cannot be constructed without trust."

Niklas Luhmann [Luh00]

There are different research areas where the issue of trust has been touched. Originally, according to Martin Endreß [End02], trust has raised interest mainly in theological, psychological and philosophical research. Since the 1990ies, increased activities of trust research were conducted in the areas of economy, organizational theory, political sciences and sociology. Finally, mainly since the turn of the millennium, the discipline of computer science has recognized the full potential of trust in various areas.

Therefore, the answer to the question "What is trust?" varies greatly, depending on the research area that the queried person is coming from. In this chapter we provide the reader with several sample trust definitions to underline these differences and cover relevant background information on trust and reputation from two different points of view. Firstly, we investigate the insights of the field of the social sciences, mainly from sociology and psychology. Secondly, we give an overview on the trust research conducted in the area of information technology. Finally we summarize our findings.

2.1 Trust Insights from the Social Sciences

“[. . .] trust is composed of two elements: an effective attitude of optimism about the goodwill and competence of another as it extends to the domain of our interaction and, further, an expectation that the one trusted will be directly and favorably moved by the thought that you are counting on them [. . .]”

Karen Jones [Jon96]

In this section, we investigate findings from the social sciences, which highly influence the definition of our trust model and our understanding on trust update presented in Chapter 6 and the concept of trust transitivity discussed in Chapter 7. After the introduction, we briefly present the prisoner’s dilemma, a game conducted by game theorists and followers of rational choice theories, and point out in Subsection 2.1.2 the implications that can be gained for trust. In Subsection 2.1.3 we cover important characteristics of trust, like future direction, subjectivity and asymmetry. In the following Subsection 2.1.4, we discuss the functions of trust and answer the questions why we do trust and why we want to be trusted. After having pointed out the benefits and limitations of trust, we move on to discuss how trust is built and destroyed in Subsection 2.1.5. Trust can be differentiated in different classes, for instance the trust amongst persons or the trust between persons and organizations, which we discuss in Subsection 2.1.6. Finally, after having presented many results with regards to trust, we cover the concept of mistrust in Subsection 2.1.7 and put it in relation to trust.

2.1.1 Introduction

Authors agree [End02, Luh00] that trust as a prerequisite of social processes and an elemental factor of social life is a fundamental phenomenon of sociology. However, trust is not solely a social phenomenon but build in an interaction environment that is affected by both, psychic and social effects and which cannot be assigned to a single effect class exclusively.

Historically, as Martin Hartmann points out in [HO01], the concept of trust is discussed already in 1651 by the English philosopher Thomas Hobbes in his *Leviathan* [Hob51], one of the most influential works of political philosophy. Hobbes states that a contract between two parties relies on trust, if the contractually specified services are exchanged at different times. Therefore, if one party performs a service before the other, it has to trust the other party in the meantime. This trust relies on the existence of a third party independent of the other two that is able to put sanction mechanisms in place in case of a dispute. Hobbes therefore points out three interesting aspects which we investigate further in the following sections: trust as an interpersonal relationship, future direction and the importance of sanction mechanisms.

Carolyn McLeod [McL07] provides an insightful summary of trust. She states that *"One's attitude is conducive to trust if it conveys the following: an acceptance of risk, especially the risk of being betrayed; an inclination to expect the best of the other person (at least in domains in which one trusts him or her); and the belief or optimism that this person is competent in certain respects"*. Therefore, trust is directed towards the future and involves expectations towards another person, that this person will not try to harm what is entrusted in some way if the possibility to do that presents itself. This implicitly also assumes that there is no direct control over the trusted person, otherwise the possibility of harm would be no issue. Other authors [Bai01, ST03] explicitly state the requirement for the possibility for betrayal. Furthermore, Niklas Luhmann goes in [Luh00] even so far as to require a certain interest for betrayal in the person, in order for an experience with that person to be valuable for building trust.

This definition also highlights the importance of *goodwill* of the trusted person (trustee) towards the trusting person (trustor), similar to Karen Jones' definition of trust presented in the beginning of Section 2.1 and the work of Annette Baier in [Bai01]. We further investigate the motivation for trusting and being trustworthy in Subsection 2.1.4.

Finally, let us conclude the introduction by pointing out that *it is not always trust we are talking about*. Olli Lagerspetz discusses in [Lag01] an illustrating example of letting a friend stay overnight and allowing him to sleep in the kitchen ... where the knives are. Do I trust that he does not take a knife and assault me in my sleep? Although it goes without saying that I let a friend sleep in the kitchen, this action is not necessarily called a matter of trust. Lagerspetz describes this action as acting without thought, which is not to be confused as acting thoughtlessly. It is simply normal and perfectly healthy to not take into account some suspicions. Obviously, there is also no mistrust towards this friend, but the question of trust was *simply not asked* in this example. Therefore, it is worth thinking about *when to talk about trust* and *when not*.

	Bob is silent	**Bob betrays**
Alice is silent	Both stay in jail for 6 months	Alice is sentenced to 10 years Bob goes free
Alice betrays	Alice goes free Bob is sentenced to 10 years	Both stay in jail for 3 years

Table 2.1: The prisoner's dilemma

2.1.2 Rational Trust: Excursus to Game Theory

James S. Coleman proposed an economical approach from rational-choice theory for understanding trust relationships, as Martin Endreß describes in [End02]. In general, rational choice theory dictates, that a human actor will choose the one action from all possible actions that maximizes the personal gain. From the point of view of the trustor who does not know the future behavior of the trustee, he can strive to understand the partner's motivational structure and figure out what would be the profit and loss in case of trust or a misuse of trust. Therefore, according to this definition of trust, individuals trust rationally, if the rate of the probability of the trustor behaving trustworthy to the probability that he does not is larger than the rate of potential loss to potential gain.

Although Coleman's purely economic approach is controversial among sociologists, it is based on an interesting model from game theory called the prisoner's dilemma. Alice and Bob have been caught by the police and are charged for robbing a bank. Since there is no real evidence, the judge separates Alice and Bob and makes the same offer to both of them: "If you confess your crime, and your partner stays silent, you are set free but I will use your information to sentence your partner for 10 years. If you both stay silent, you each get 6 months in jail for having a weapon. If you both confess, you are convicted to 3 years in jail." From the point of view of Alice, as can be seen in Table 2.1, she gets free by betrayal (instead of 6 months in jail) if Bob is silent and even if Bob betrays, Alice still gets sentenced only to 3 years if she betrays (instead of 10 years if she is silent). The dilemma is that whatever the other person chooses, one is better off by betrayal whereas this local optimum is not the best overall solution of each person staying in jail for 6 months. However, even if Alice and Bob could communicate, they might not trust each other enough to risk the worst outcome of staying in jail for 10 years.

Tit for tat was introduced by Anatol Rapoport, a Russian-born American psychologist, as an effective solution strategy for the iterated prisoner's dilemma. This approach recommends the prisoner to at first cooperate, thus keeping silent in the actual choice of betraying. In the following iterations, the prisoner should each time respond exactly as the opponent does, therefore betray if the opponent betrayed and keep silent if the opponent kept silent. The four conditions that this strategy depends on have interesting implications on our understanding of trust: Firstly, the agent cooperates initially. Secondly, the agent retaliates if provoked. Thirdly, the agent quickly forgives. And finally, the number of iterations is relevant for the strategy to be successful: Obviously, since the agent initially cooperates, a certain minimum number of interactions is necessary. In addition to that, the exact number of interactions should be unknown, since if the agent knew that there was the last interaction, he would betray and gain a higher profit.

To summarize the implications on our understanding of trust, it would be a mistake to assume that rational individuals will necessarily or automatically act cooperatively, even if these actions are beneficial to all group members. The mere possibility that the second agent in the prisoner's dilemma might not cooperate could lead the first agent to do the same, even just to defend himself. However, being optimistic regarding the partner by initially cooperating gives the opportunity to build the first positive experience and therefore start the trust generation process. In general, the participants learn from the experiences and act accordingly.

2.1.3 Characteristics of Trust

After having gained some insights from Game Theory, we list in the following several important aspects or characteristics of trust.

Future Direction

Trust is directed towards the future [Luh00, Tho06]. More clearly, the time lag between an act of trust and the expected return or action is of central importance and represents the risk of trusting, as stated by Coleman in [End02]. As we pointed out in Section 2.1.1, Thomas Hobbes took already in 1651 a similar view in his work Leviathan [Hob51], where he noted the importance of trust to overcome the challenge of services of a single contract being performed at different times.

Risk and Benefit

Annette Baier points out in [Bai01] various stages of trust awareness, starting from unconscious trust, moving to awareness of a risk together with confidence that the risk is acceptable, followed by recognizing why we can accept a certain risk and finally coming to an understanding what can be gained or lost when taking this risk. Baier illustrates that when a trustor trusts a trustee, the trustee is granted the ability to betray this trust, but at the same time the trustor is confident that the trustee will not use this possibility. Trust can be seen as an accepted vulnerability for possible but not expected bad intentions.

Luhmann [Luh01] states that a matter of trust is different from hope. It is a case of trust, when the trusting expectation decides the outcome of a decision; otherwise it is merely hope. He raises the example of a mother leaving her children in the care of a babysitter. Obviously, the mother hopes that the babysitter will be nice to the baby, will not disturb the baby and so on. Trust however is only involved, when the mother would regret having gone out altogether. Thus, trust is related to a *critical alternative*, whereas the damage in case of a betrayal of trust may be higher than the advantage of a successful outcome.

Alexander Thomas [Tho06] cites Deutsch [Deu62], who also stresses the fact, that trustworthy behavior increases the own vulnerability, that it happens towards a person outside of personal control. Where Luhmann sees merely the possibility of higher damage in betrayal opposed to the benefit of a successful cooperation, this possibility is strongly emphasized by Deutsch. He agrees with Schweer and Thies in [ST03] that the potential damage in case of a betrayal is in general higher than the benefit of honored trust.

Risk and Rationality

Trust can hinder our ability to assess someone's character clearly. According to Karen Jones [Jon96] trust may make us resistant to evidence that contradicts our optimism about the trustee. Also Niklas Luhmann [Luh00] notes that when a decision has to be made, humans do not necessarily judge rationally the involved risk and the reasons for trusting.

Russel Hardin however stresses the proper sequence of trusting and making a decision that is based on trust but involves a certain risk. He states in [Har01], that trust is no risk and no game. Obviously, it can be risky to move oneself into a position where others can do harm. But generally one does not first calculate the risk, and then decide whether or not to trust. Instead, one already trusts to a certain degree, and then decides – depending on the risk – whether and how to act.

Also Olli Lagerspetz cautions [Lag01] that one has to be careful when tying trust too closely to risk. There are many situations in life when we trust without risking a lot if at all. For instance we trust our friends, but to suggest that we risk trusting them would be a strange way of defining friendship.

Latency

Hardin points out in [Har01] that trust is not a conscious strategy. As Annette Baier illustrates [Bai01] saying "trust me" does not make a lot of sense, since if we do not trust already, simply saying this does not give us cause to do so. Therefore trust cannot be willed, as Diego Gambetta agrees in [Gam01]. Coming back to Olli Lagerspetz, an actor in general does not realize his attitude as an attitude of trust. To do that would mean to consider the possibility of betrayal. To trust, however, exactly means the opposite of not considering this possibility.

According to Luhmann, a trust relationship has to be latent in its nature in order to work the generalization. The trustor has to act open and trusting towards the trustee, otherwise the first step towards mistrust could have been laid already. Trust in the very end cannot be justified. Although trusters will in most cases be able to specify why they trusted in a certain case, this is mostly due to their self-esteem and social justification if trust was misplaced.

Finally, Annette Baier states, that *people live in a climate of trust like in an atmosphere: we take it for granted and only notice it when it is polluted or violated.*

Subjectivity

> *"Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action. [...] When we say we*

trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him."

Diego Gambetta [Gam00]

According to the aforementioned quote from Diego Gambetta, Trust can be seen as a certain threshold of a probability distribution that a person will perform a certain action, which has been gained from all sorts of evidence. He now asks how high the probability threshold needs to be for someone to cooperate in the action, whose success depends on whether or not the other person cooperates too. His answer is, that *the optimal probability threshold*, when we are certain of trusting the person strong enough to cooperate, *varies*. This variation can be influenced by different criteria, like if the person is risk-loving, if the cost of misplacing trust is much higher than the gain of potential trust, or if the pressure of following the cooperative action is too high despite possible mistrust etc.

Also Luhmann states, that the perception and assessment of risk is a very subjective matter. This is what makes people different from each other and produces different types of risk-loving, risk-averse, trusting and mistrustful people.

Asymmetry

Trust is asymmetric in two different senses. Firstly, due to the aforementioned subjectivity, a trust relationship is inherently asymmetric, which means, that if Alice trusts Bob, Bob does not necessarily trust Alice automatically. Secondly, the process of building and destroying trust is asymmetric as well. Ulf Bernd Kassebaum points out [Kas04] the effect (he calls it "paradox"), that it takes much time and effort to build trust, whereas trust can be destroyed so easily. Furthermore, where trust has been destroyed once, it is much harder to rebuild.

Reciprocity

According to Martin Schweer and Barbara Thies [ST03], *reciprocity* is another key element for trust. When we invest trust, we expect an interaction partner to invest trust as well. Interpersonal relationships tend to fail if one of the partners feels that his or her advance in trust is not returned.

We should note that this element of reciprocity does contradict the aforementioned element of asymmetry. While in our view the argument of Schweer and Thies is valid in the case of interpersonal trust, one can derive examples where this is not necessarily the case, e.g. business-to-consumer (B2C) commerce or eBay Powersellers in consumer-to-consumer (C2C) auctions. For both of these examples, trust of the buyer in the sellers is absolutely required, however, this trust does not influence the trust of the seller in the buyer.

2.1.4 Functions of Trust: Why Trust at all?

In this subsection, we discuss the functions of trust, what it is used for and why humans do need to trust. Furthermore, we investigate the motivation why humans want to be trusted.

Precondition

Authors agree [Gid95, End02] that the main precondition for trust is not the absence of power, but the *absence or incompleteness of information*. Luhmann expresses this fact in stronger words [Luh00] and states boldly: “Trust builds on deception”. He clarifies that although in reality the actor has not enough information to decide how to act, the actor ignores this missing information willingly.

Reduction of Complexity

Trust is used to reduce social complexity. This finding of Luhmann has been recognized as relevant by many researchers in the social science community. In his words, the outside world is overly complex for every kind of real-world systems; the future holds too many possibilities for the system to react upon. Therefore, systems create an own selected environment, an inner view, and cut back this future to a manageable presence.

In other words, the social environment is so complex that no person can understand and process all the available information. Therefore, mechanisms for the reduction of this complexity are required also in social situations in order to retain the ability for acting. According to Luhmann, trust is the central mechanism for this complexity reduction. The more advanced a society is, the more important this complexity reduction becomes. As soon as a human being cannot perform all actions necessary in a society, trust becomes necessary so that division of labor is possible.

Reduction of Risk

The past dominates over present and future in trusted worlds. When looking in the past, there are no other possibilities left, so complexity is reduced already [Luh00]. One expects the known past behavior to continue in the future. However, trust is not merely a function of past behavior, instead it uses this information but strives to determine the future. Therefore, during the act of trusting, the complexity of the future is reduced as if only some possibilities for this future were possible. The reduction of complexity goes hand in hand with a reduction of perceived risk [ST03]. Based on this trust, an individual acts as if certain possible (but negative) outcomes of an interaction could simply not occur.

Somebody who trusts accepts a certain level of risk or vulnerability. This level is minimally the failure of the trustee to do what the trustor expects. The trustor can reduce this risk by monitoring the trustee or imposing certain constraints on the trustee's behavior. However, according to [McL07], "[...] the more monitoring and constraining s/he does, the less s/he trusts that person [...]".

Reduction of Cost

Trust replaces required social control and lowers transactional cost. According to Offe [Off01], there is no need to monitor trusted persons, nor to buy what they give me freely, nor to force them to do what I would expect. Trust starts where we stop controlling and enforcing.

Increased Action Space and Improved Cooperation

Just appearing in front of others requires a certain amount of trust [Luh01], more specifically the trust not to be misinterpreted but to be understood in the way that one likes to show her- or himself. This matter of trust is so significant for some people that their mere appearance (and even more acting) in front of others is difficult for them. Through the lack of trust, their action space is limited. The possibility for action increases hand in hand with an increased ability to trust.

Trust enhances cooperation [McL07] and makes cooperating with people less complicated, because it removes the incentive to check up on other people. Trust also leads to more spontaneous communication [ST03] with regards to the person does not continuously think about whether or not to pass on certain information.

According to Piotr Sztompka [End02], any action is happening under the conditions of complexity and uncertainty and trust is necessary to overcome these issues. Functions of trust include increased potential for acting, the omission of control structures, increased tolerance for ambiguity due to a mutual feeling of confidence, and increased social capital.

Improved Life

Nobody is able to manage all things alone [Bai01]: We need the help of other people in our daily life. Therefore we need to trust others to perform services for us and hope they will not harm us or the persons and things we care about. Entrusting goods or people to somebody involves a discretionary power that the trustee is granted for judging what exactly is necessary for properly caring for the good or performing the service.

In general, trust enables having a decent life, since it allows for instance children to depend completely on their parents, and old or disabled people depend on their care providers. Therefore, trust might very well be the basis of society, for – according to McLeod [McL07] – without trust in fellow citizens to honor social contracts, the contracts probably would not exit. Russel Hardin presents in [Har01] the interesting observation that a person who is generally optimistic has more possibilities to learn from experiences, and therefore find other trustworthy entities.

Motivation for Being Trustworthy

One century after Hobbes, the Scottish political economist and philosopher Adam Smith gave his insights [Smi82] on the motivation for being trustworthy. He assumed that anybody conducting business will worry a lot about the loss of reputation and therefore will stick very closely to each deal. He states that what can be gained through 20 successful contracts can never be gained through fraud, since in business just the semblance of fraud leads to financial loss. Gambetta [Gam01] concludes from this that it might be hard to rely on altruism, but even harder to avoid relying on a reputation of trustworthiness. In our view, although the statement of Adam Smith might have been true in general in the 18th century, the Internet of 2007 offers an enormously increased potential for fraud that is especially hard to detect or to protect against or sanction for that matter. Therefore, we need additional mechanisms to detect fraud and communicate information about fraudulent parties to other entities in the process of doing business.

The research community is still discussing whether the motivation of a trustee to play by the rules is important for him or her being trustworthy, or not. Three different levels of assumptions regarding a trustee's motivation are identified by Offe in [Off01]: The least assumption would be that each actor will try to maximize the personal gain while not risking sanctions from third parties. The next level is whether an actor will also see the overall advantage and will take into account common welfare. The strongest assumption for the actors motives is the expectation that he will react with goodwill to my trusting him.

Annette Baier has proposed this so-called goodwill view [Bai01], which states that if I trust another person, then I depend on the goodwill of this person. This strong statement of being trustworthy out of goodwill, for solely altruistic reasons, is not self-evident. A person could act trustworthy for selfish reasons (level one in the aforementioned categorization by Offe), because of fear of sanction mechanisms, or because of a large potential for gain in the longer term. However, it seems reasonable that trust founded on altruistic reasons is stronger and more stable, because actors do not need to expect their partner to betray their trust at each new lucrative opportunity. Self-interest alone, therefore, is seldom the sole motivation in healthy trust relationships. Finally, McLeod notes [McL07] that being trusted can improve self-respect and allow also to be more respectful towards others.

2.1.5 Building and Destroying Trust

In this subsection, we investigate how the ability to trust is initially created. Based on that, we point out how trust is build from iterated experiences and generalization, and what conditions are required or help the trust building process.

Basis for Trust

The ability to trust has to be learned, and the basis for trust is already posessed by an infant. According to the psychiatrist Erik H. Erikson (as stated in [ST03, End02]), the socialization process consists of eight stages of development. The first of these stages is called the “trust versus mistrust” stage and occurs during infancy from approximately birth to one year. A child will develop trust and security and a basic optimism if it is cared for, loved and comforted when needed. If the parents are unreliable, reject the child or fail to fulfill his basic needs, the infant will develop mistrust. Thus, the sense of basic trust is built primarily during the months of a baby child, and reflects its *primary experiences*. The loss of the ability to trust also is the result of experiences, mainly due to injuries during the phase of primary experiences as a baby child or due to child molestation.

Erikson states that it is not necessary for parents to be perfect and immediately fulfill all the infant’s wishes. The child should learn to let the mother out of sight without anxiety and rage because she has become an inner certainty as well as an outer predictability. Parents therefore should strive to find a proper balance while not completely eliminating the capacity for mistrust, which is necessary to develop the ability to discriminate between honest and dishonest persons. If, however, mistrust wins over trust during this stage, the child will become withdrawn, insecure, and will in general lack self-confidence.

According to Kassebaum [Kas04], results from research in attachment theory show that in fact this learned trust not only insures a secure attachment of a child, but is also highly relevant for its further development and the formation of general trust.

Learn from Experience

According to the learning theoretician Rotter [ST03], trust is a result of experiences and builds on the expectation to be able to rely on a certain statement or promise of another person. This trust is first put to the test in the family. In the following, all kinds of personal relationships can be seen as the testing and learning of trust relationships. Luhmann states that our understanding of this learning is still limited. According to him, it is possibly not enough to generalize simple experiences with the environment.

Furthermore, it is unclear how this generalization starts, how a child transfers good experiences with trusting the mother to the father, to siblings and finally also to strangers.

Nevertheless, authors agree [End01, Gam00, Har01] that being able to trust or being trusted is based on *past experiences*, which for instance have proved a person to be trustworthy, a technique to work, and an expert to be competent. Trust is therefore always generated within the context of an *interaction history*. Furthermore, in addition to learning directly from one's *own experiences*, it is also possible to learn from the *experiences of others*. Therefore, in a more general sense, we integrate external knowledge elements from interaction histories into the own internal knowledge. Gambetta notes that rational people will not only search for proof for their beliefs, but will also communicate this proof to other people.

Iterate

According to Piotr Sztompka [End02], a culture of trust is the result of a *steady number of positive experiences* with both trusting others and being trusted oneself. Therefore, the *durability of a relationship* and its *interaction history* are significant preconditions for building trust.

However, the number of positive experiences is as important for stabilizing trust as is the fact that the participants are *aware of an interaction history*, whereas the trustee has offered reasons for being trusted and the trustee has accepted the risk of trusting. Offe [Off01] stresses the role of the moral obligations of the trustee who simply by being trusted is obliged to honor this trust and not betray. In addition to moral obligations, rational self-interest can be a further mechanism for stabilizing trust relationships. The loss of credibility that occurs when betraying trust could not only have local effects on the trustor (whose trust had been betrayed) but might also negatively effect other existing trust relationships of the trustee and hinder new trust building.

Luhmann points out [Luh00] that trust prospers in a context where the relationship of the actors has a certain duration with a certain mutual dependence and a moment of unpredictability. One typical sequence of building trust looks like the following: A trustor has to define a situation where trust is necessary and where she risks a misuse of trust. The trustee has to have the possibility to and also some interest in misusing this trust. In the following action, the trustee has to disregard this interest and instead honour the trust of the trustor. This sequence has to be repeated multiple times with increasing effort or involved interest.

Generalize

The process of generalization, which has been originally defined in the context of learning theory in behavioral psychology, can be applied to the phenomenon of trust in three different ways: Firstly, single points of experiences are generalized into a feeling of trust. Secondly, existing knowledge in one context is generalized to other contexts. Thirdly, experiences with already known people are generalized into a feeling for strangers.

Luhmann recognizes the first area and states that a trust relationship has three structural components: the substitution of an inner order for a more complex outer order, a process of learning, and a symbolic fixation of the result in the environment. This symbolic fixation refers to the process of taking spot tests as being relevant for the whole system. The indicators for trustworthiness of a certain person or organization are checked periodically as feedback whether a continuation of trust is still in order. According to Luhmann, this fixation is a necessary simplification, since the reality is too complex for real control.

The second area relates to the definition of generalization in learning theory. According to Luhmann and Rotter, generalization in learning theory refers to an organism responding in a similar manner to stimuli that are to some degree different from the ones it was trained on. Thus, generalization allows to transfer behaviors learned in one context to novel situations. Trust judgments generalize individual experiences, extend to other similar cases, and stabilize an indifference against discrepancies. Additionally, generalized expectations are not only learned through individual experiences, but also through adopting experiences of others or even taking on recommendations from mass media.

The third area is based on observations from Hardin [Har01]. When trusting, we can gain or loose, and this experience is added to the knowledge that is used for future interactions. When meeting a complete stranger, there is in general insufficient information or none at all available. What we therefore do is generalizing past experiences with other persons. If these past experiences have been mostly positive, one might be inclined to trust this stranger. If they were negative, one might pessimistically mistrust the stranger. This effect is called *dispositional trust*.

Conditions

In general, according to [McL07], the reasons for trusting somebody can be numerous and very subtle; they can even rely on body language for instance. To further pin down these subtle reasons, Schweer and Thies note in [ST03] that people have different normative expectations how an interaction partner should act in order to be able to trust him or her. Therefore, people have *subjective knowledge* regarding the prototype of a trustworthy person in a certain context. This knowledge determines the first impression of another person: the impression is positive if the found reality does match the normative expectation, otherwise negative. Also, building trust requires also a certain optimism about the competence of the trustee in a certain area.

Furthermore, an important precondition for building trust is the potential and possibility for betrayal. According to Annette Baier, trust is betrayed and not just disappointed. Disappointment would be the case, when somebody just relied on or hoped for a certain behavior. This reliance could be disappointed, but not betrayed. Gambetta [Gam01] generalizes Baier's requirement towards the trustee. He states that trust requires the liberty of both, the trustee and the trustor. The trustee needs to be able to betray our trust or to cooperate. However, also the trustor needs to be able to decide, whether or not to perform the action in question.

According to Offe [Off01], one of the main challenges of building trust lies in overcoming the transition from close-area trust (e.g. trust in friends and potentially in my friends' friends) to trusting strangers. He states that there is a phase before trusting, which is motivated by the obvious benefits of trust, and which is characterized by the testing and verification of the trustworthiness of relevant potential interaction partners. Somebody who is trustworthy will accept the thorough examination willingly without giving cause to negative judgment.

Trust and control or sanction mechanisms need to be balanced. Schweer and Thies mention the results of a study [DGdV01] performed by Carsten de Dreu et al. that examines trust in the business space. One of the results is that the possibility for sanction mechanisms might affect trust of a negotiation partner adversely if he follows a cooperative strategy. According to the authors, the higher power of the partner may lead to conflict avoidance which may lead to a lower trust on both sides. The aforementioned element of reciprocity is weakened here.

This matter, however, is not resolved yet, as Martin Hartmann states in [HO01]. According to his arguments, trust is not limited to partners of equal power, which invest in an interaction that is beneficial to them both. Instead, issues of trust can be seen also in relationships, with a significant imbalance in power between the participants. Examples are given of the trust between children and their parents, and of sick people and their caretakers, which the contractual approach cannot fully grasp.

2.1.6 A Taxonomy of Trust: Personal Trust versus System Trust

According to [ST03], research in psychology differentiates between *personal trust* and *system trust*. In this subsection, we investigate the concept of system trust and discuss one special type of system trust that is called *institutional* or *organizational trust*.

System Trust

While personal trust occurs in inter-personal relationships, system trust can be described as trust in the correctness of certain principles. The necessity of system trust is explained by Luhmann [Luh00], again with the problem of high complexity. He states that trust is at first a personal and therefore limited trust, used to overcome moments of uncertainty regarding the behavior of other human beings. However, trust has to be expanded, as the complexity increases, and the other humans are recognized partly as originators of this complexity. Personal trust is therefore changed into a system trust that implies a conscious abandonment of possible further information and continuous control or monitoring. However, similar to personal trust, the function of system trust also lies in the reduction of complexity.

Anthony Giddens, according to Martin Endreß [End02], describes the dependence of system trust on mechanisms of disembedding, whereas disembedding refers to lifting social relations from local contexts of interaction and their restructuring across time and space. One example that is discussed is the usage of money. Money enables us to interact with individuals or institutions that are unknown, possibly untrusted and removes the need to know them better. Trust in money means first of all trusting in the variety of money's usage possibilities, in the stability of a currency and so on. This, however, works only because of our trust that this money is accepted as a transactional medium by other entities.

A second example would be systems of scientific experts, whose knowledge also influences the knowledge of non-scientific actors. This "borrowed" knowledge is assumed to be applicable to other contexts as the original one. This trust quite often relies on the belief in principals that are unknown to oneself. More clearly, this trust does rely on experiences of the general functioning of an expert system and not necessarily on concrete experiences with the expert. This example is generalized by Luhmann who states that trust in functional authority is another kind of system trust that no longer relies necessarily on personal trust. Trust in a car mechanic to fix a car, or in a surgeon to correct an appendix are examples of authority trust, where authority is a replacement for complexity that cannot altogether be handled by each single individual

Interaction is the key factor in which personal and system trust differ. Personal trust becomes relevant in direct interactions between human beings. In the case of institutions, however, the direct interaction does often not take place. If it does take place, we are again looking at the case of personal trust towards a representative of the institution, not the institution itself. As in the case of personal trust, a risk is also involved with system trust but with fewer mechanisms of control, if at all. The effect of reciprocity, which is relevant in case of personal trust, is difficult to realize in the case of system trust.

Institutional or Organizational Trust

Some authors (Giddens, Sztompka) limit the general system trust to what they call *institutional trust*, gained for instance through professional interactions. This trust is based on the central idea of an institution (e.g. justice in case of the constitutional state as institution), as Endreß mentions [End02]), its services (e.g. economic growth) and control or sanction mechanisms. Institutional trust, the same as system trust, relies on the actions of other persons adhering to this central idea as well.

Offe cautions [Off01] that we cannot trust in an institution more than we trust in the institution's individual actors. Although institutions are expected to follow certain defined rules, they are imperfect in the sense that their rules cannot cover all eventualities. However, institutions can be trust-instilling, if their rules are publicly known and offer certain moral values. If these values not only oblige myself but make me believe that they have a similar importance to "all others", I may actually trust these others not because I know them or have made experiences with them, but since I know they follow the same institution with its rules and values.

In general, people in modern societies rely more and more on institutional trust generators or trust intermediaries by building institutional frameworks like for instance regulatory authority, standardization bodies or insurances. Examples for this include the German “Stiftung Warentest”, TÜV, the Better Business Bureau or consumer protection institutions.

2.1.7 Trust versus Mistrust

In sociology, mistrust is not seen merely as missing trust. According to Luhmann [Luh00], mistrust is not merely the contrary to trust, but can be seen as a functional equivalent to trust. Trust, as discussed above, reduces social complexity, it eases life by taking over risk. If a person decides not to trust, the original problem of complexity still remains. Therefore, expectations have to be formed in a negative way.

Luhmann states that the existence of thresholds is important for the distribution of trust and mistrust in time. Thresholds are an artificial boundary that level and therefore simplify the areas before and after the threshold. Many different experiences within one area are seen as similar and the basic behaviour is not changed as long as the threshold is not crossed. If this is the case, a possibly small step may result in a large change. Not every disappointment installs doubt regarding the trusted environment, but the threshold determines when trusted behavior switches to mistrust. The position of these thresholds is not so much given by nature as it is determined by the history of the system itself, which includes for instance the history of self-profiling and the means of simplification.

According to more recent insights [ST03] it is doubtful whether a single dimension is sufficient to capture both trust and distrust. Schweer notes that individuals themselves differ with respect to whether they see trust and mistrust as the endpoints of a single dimension or whether two different dimensions are perceived. In the case of two dimensions, two states are required to represent “not trusting”, which is not to be confused with mistrust, and “not mistrusting”. From a common sense perspective, this seems reasonable, since the fact of “not mistrusting somebody” does not necessarily imply “trusting somebody”. The actions of a person should differ significantly based on whether a person does not trust somebody or whether this person mistrusts somebody.

Trust is a strange conviction, as Gambetta reasons [Gam01] that does not so much rely on proof but rather on the absence of counter-evidence. According to Offe [Off01], trust is mainly, but not solely, measured with negative experiences of not interacting in certain cases. Examples for positive experiences are the acceptance of entering relationships where control is not possible or too costly to maintain, and the frequency and duration of such relationships. It is very hard to invalidate strong mistrust through positive experiences, since mistrust generally keeps us from allowing the experiment to happen that could result in these positive experiences. Therefore mistrust has the inherent ability for self-fulfillment. Also Luhmann notes that mistrust can result in a "self-fulfilling prophecy" since there is a tendency for mistrust to reinforce itself. A system that is tuned badly can come to an equilibrium with the environment not by correcting itself due to its effects but by reinforcing itself and use its own effects to create new causes as well. A social system that needs mechanisms of mistrust needs also mechanisms for its control to prevent a reinforcement. These mechanisms may be socially recognized entities disclaiming insignificant acts of mistrust as involuntary acts, or as errors, therefore as actions that allow mistrusting behavior but hide the mistrustful attitude. Additionally, institutions of retribution, penance and forgiveness stabilize such a system and determine points in time when a certain act is concluded and no longer gives reason for mistrust.

A system may require both concepts: trust and mistrust. Luhmann raises the example of the mobility of a 2-year-old child, where members of the family are legitimately mistrustful, but at the same time have mutual trust in this mistrust. In organizations, some controls have to operate under specific mistrust conditions. Some roles, for instance researchers, are even required to address trusted content with mistrust. Sztompka [End02] speaks of an institutionalization of mistrust as trust-building mechanisms in the area of politics. According to Endreß, this paradox can be generalized further and extends to areas like the Better Business Bureau or consumer advice centers.

2.2 Trust in Information Technology

After having presented insights from social science trust research, we here span an overview of the understanding and usage of the concept of trust in the area of information technology. We structure this section as follows: Firstly, we cover the area where trust was first touched in an IT context: *trust in cryptographic keys*. Secondly, we highlight *trust in platforms* as a relatively new research area. Thirdly, the topic of *trust in agents*, in other words *reputation systems*, is outlined.

2.2.1 Trust in Keys

As an integral part of business transactions, trust is easier to establish in person than online. Yet as organizations adopt e-business processes to leverage the benefits of the Internet, the issue of trust – knowing who is initiating a transaction, and whether the information being sent has been altered in transit – takes on increased importance.

RSA Security[1]

The topic of trust appeared first in an IT context with regards to trusting cryptographic keys. As stated by the aforementioned definition of RSA Security, trust refers here to being sure of the identity of a certain user. This can be achieved via different authentication techniques relying on either symmetric or asymmetric mechanisms[2].

For large-scale distributions, most commonly, an asymmetric approach based on *public key cryptography* is used. Simply put, a user owns at least one key pair consisting of a public and a private key. The private key is known just to the user, whereas the public key is publicly known and can be stored in a directory for everybody to access. Asymmetric cryptographic algorithms ensure that data encrypted with the public key can be decrypted with the corresponding private key, and vice versa. Together with cryptographic hash functions that allow to generate a fixed-length digital fingerprint of variable-length input data, these algorithms are the basic building blocks for digital signatures, which guarantee to a recipient that the signed data came from the person who signed it, and that it was not altered since it was signed. Furthermore, these building blocks are used by higher level protocols like SSL or TLS that provide authentication of the communication partners[3], integrity protection and confidentiality of the communication.

The issue of trust is highly relevant for the question how to securely publish the public key and bind it to the owner's identity. The method of choice consists of digital certificates. They contain a well defined (e.g. by the X.509 standard) set of fields including the user's identity and public key, bound together by one or more digital signatures

[1] http://www.rsasecurity.com

[2] Giving an in-depth introduction to cryptography is outside the scope of this dissertation, therefore we highlight in the following solely the issue of trust in keys. More information regarding the cryptographic details can be found in [Sch96].

[3] Although in the Internet, client authentication is not widely used due to the limited availability of client certificates.

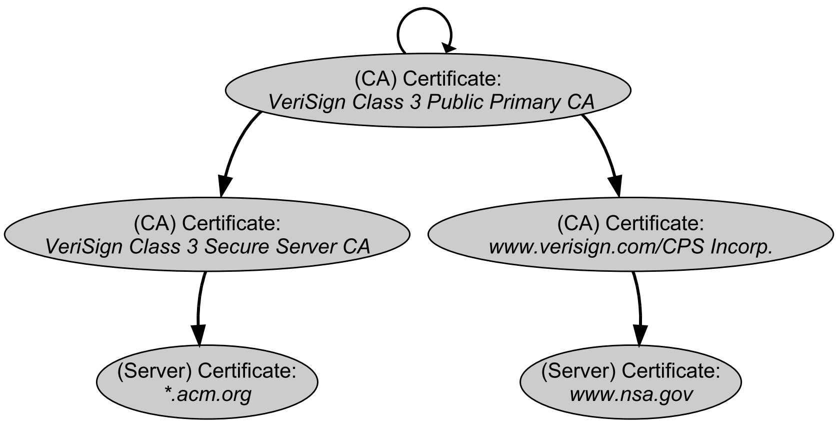


Figure 2.1: The hierarchical PKI trust model

of trusted third parties (TTP) who vouch for the claim that this link is true. Thus, the question of whether a certain key really belongs to a certain user is replaced by the question whether a certain TTP (or group of TTPs) is sufficiently trusted to make this statement. Two approaches for organizing this step are briefly presented in the following: *hierarchies of trust* through *Public Key Infrastructures* (PKI) and *webs of trust* through *Pretty Good Privacy* (PGP).

Public Key Infrastructures – Hierarchies of Trust

In PKIs, centralized entities called certification authorities (CA) are introduced to issue certificates to users containing the aforementioned claims. For scalability reasons (among others), these CAs are organized into hierarchies. Within these hierarchies, user certificates are issued mainly by the lower level CAs, higher-level CAs issue certificates to lower-level CAs, and certificates of top-level CAs are self-signed, as depicted in Fig. 2.1.

From a certificate user's perspective, trust in a CA can be setup manually by importing this CA's certificate into a trusted root certificate storage, managed e.g. directly by the operating system, a web browser or an application server. The storage most likely contains certificates of many top-level CAs already. Trust is automatically induced to all certificates that were signed by these trusted root certificates, and from there recursively further down the hierarchy.

In addition to the hierarchical CA structure, the concept of cross-certification was introduced, which allows two CAs to establish a bilateral trust relationship outside the normal hierarchy. Thus, certificates issued by CA_1 will be accepted as valid by users trusting CA_2.

To determine whether or not a certain received certificate is trusted requires the discovery of a certification path from this received certificate up to a trusted root certificate. Protocols like the lately revised *Server-based Certificate Validation Protocol* (SCVP) [FHM+06] can be used to ease certificate usage for applications and delegate the certification path construction and validation to a so-called SCVP server.

Pretty Good Privacy – Webs of Trust

Originally, pretty good privacy was published in 1991 by Philip Zimmermann as a software for encrypting email communication [Zim95]. While the PKI approach uses a logically centralized trust model, PGP relies on a completely distributed approach.

An X.509 certificate used mainly in the PKI context contains a single digital signature of the CA that issued the certificate. A PGP certificate on the other hand also contains the user's public key and identifying information, but is self-signed. In addition to that, it may contain many digital signatures of other entities, who – by signing the certificate – state that they believe that the contained public key belongs to the specified owner. Thus, where trust in the PKI context originated solely from the CA validating a user's identity, trust in PGP originates from a multitude of individuals performing the validation process.

The PKI automatism for accepting a certificate as valid resulted from the ability to find a certification path up to a trusted root certificate. Due to its distributed approach, PGP's automatism works differently. Every user has a so-called local *keyring*, where the own public-private key pair is stored. Additionaly, the keyring contains the certificates of the (possibly personally known) people whose identity were validated by the user by signing the corresponding certificates. A trust value in the categories "complete trust", "marginal trust" or "notrust" is manually assigned to each of these certificates, depending on the estimated quality of the certificate owner as certificate validator. Once a new certificate is received, the automatism will accept the certificate as valid if it contains one signature from a completely trusted or two signatures from marginally trusted entities.

Although relying on simple mechanisms, the PGP approach is an interesting and successful example of a distributed program building a web of trust between individuals, and therefore can be seen as one of the first distributed reputation systems (see Subsection 2.2.3).

2.2.2 Trust in Platforms

> *Trust is the expectation that a device will behave in a particular manner for a specific purpose.*
>
> Trusted Computing Group[4]

This definition of trust captures very well the intentions of the Trusted Computing Group, an industry-driven standardization body that took over in 2003 the work on trusted computing started 1999 by the Trusted Computing Platform Alliance (TCPA).

The basic idea of the trusted platform approach is to place a trusted chip inside the PC that supports – among other things – integrity measurements of the platform's hardware and software components and secure storage of these measurements. Upon bootup of a trusted platform, the chip can measure the integrity of each step of the boot process and therefore determine whether the platform is still in the specified state in which case the boot process can proceed to the next step.

The reader should note that there is no mention of a trustworthy platform, more clearly, the trusted platform work can not determine flaws that already exist within a platform's code. However, it can detect later changes to the platform, e.g. done by viruses or hackers. More information regarding trusted platforms can be found in Chapter 4.

2.2.3 Trust in Agents: Reputation Systems

> *Trust: a subjective expectation an agent has about another's future behavior based on the history of their encounters[. . .]*
>
> Lik Mui [Mui03]

The similarity of this definition by Lik Mui to Gambetta's definition presented on page 13 highlights the close ties of the trust research that originated in the software agent

[4] https://www.trustedcomputinggroup.org

field to the work done in the social sciences. The terminology used in this area, however, is confusing at best. We therefore clarify briefly the wording before presenting in the following the topic of trust in agents, in other words *reputation systems*. After covering the basic building blocks, we highlight some examples for centralized and distributed reputation systems.

Terminology

Trust Management. This term has been originally defined by Matt Blaze, Joan Feigenbaum and Jack Lacy in their early work on *Policymaker* [BFL96] and refined in the follow-up work on *KeyNote* that has been published as an Internet RFC in [BFIK99]. The general idea is based on the building blocks certificates, security policies and trust relationships in the certificate issuers. The challenge is to determine, whether a particular set of credentials satisfies the specified policies and therefore is "trusted" for a certain action.

Reputation System. Resnick et al. point out in [RKZF00], that a reputation system *"[. . .] collects, distributes, and aggregates feedback about participants' past behavior. Though few of the producers or consumers of the ratings know each other, these systems help people decide whom to trust, encourage trustworthy behavior, and deter participation by those who are unskilled or dishonest."* In addition to the information in that quote, it is important to note that it does not matter from a reputation system view, whether the target entity that is trusted is a software agent, a virtual identity, or a real-world person. The focus of trust management systems lies mainly on the automatic authorization decision of actions, whereas reputation systems strive to provide users with the necessary information to make a decision, but do generally not make the decision themselves. During the last decade, the terms trust management system and reputation system have been often used interchangeably.

Recommendation System. Recommendation systems enable their participants to find and issue recommendations about the quality of products or services. Often, recommendation systems are combined with reputation systems to allow finding high-quality recommendations from trusted sources. A typical example of a recommendation system is the Amazon[5] product review process where users can state their opinion on the offered goods. With the later addition to rate review

[5] `http://www.amazon.com`

quality and thereby to rate the reviewer, which allows the ranking of reviewers like e.g. “Top-100-Reviewer”, Amazon added reputation system aspects to their recommendation system.

Collaborative Filtering System. Collaborative filtering is a technique to determine the similarity between users. This similarity can be used in a recommendation system context to give specific recommendations to a user based on the recommendations of other users with a similar mindset. This concept is for instance used in Grouplens, a collaborative filtering system for Usenet news introduced by Konstan et al. in [KMM$^+$97, SKB$^+$98]. Grouplens is based on the idea, that if Alice and Bob both like articles X, Y, and Z, and Alice likes article W, then Bob might like article W as well. The term recommendation system in this context has a dual meaning, since it is not only the users recommending articles but also the system recommending suitable articles to its users.

Building Blocks

A reputation system consists minimally of the four basic building blocks of the *trust model*, the *trust update algorithm*, the *storage* for persisting inputs and outputs of the trust update algorithms and finally the *protocols* for communicating the data.

As pointed out in the terminology, the main purpose of a reputation system is to determine whether a certain entity is trusted or not. The trust model describes how trust is represented in the system. Different representations are used in the related work, ranging from binary ratings, over discrete metrics, to probabilities. The granularity of the model determines the expressiveness of the reputation system’s trust statements.

In addition to the trust model, a computation engine is the second basic building block of a reputation system that allows to initially determine and in the following update trust based on certain inputs. Typical inputs include a trustor’s experiences with the trustee or the experiences of other parties with the trustee. The output is the computed trust of the trustor in the trustee in the representation of the trust model.

The data storage and protocols to communicate this data are closely intertwined. A reputation system architecture can be categorized in a centralized or distributed reputation system, depending on whether the reputation information is stored in *centralized* location for all the participants or whether it is *distributed* over agents that reside on each participant’s computer.

In the following, we first discuss several examples of centralized reputation systems and then move on to cover a selection of distributed approaches.

Centralized Reputation Systems

A centralized reputation system whose quality has been investigated in various studies [Del01, BG05, RZSL06] is the feedback system used at the Internet auction site eBay[6]. This system allows buyer and seller of an auction to rate each other after a transaction has taken place. The rating itself consists of either positive, neutral or negative feedback plus a single line of text describing the reason for the rating. The trust model is quite simplistic as well and consists of an integer number. The higher the number, the more trusted an eBay user appears. This is highlighted on the website by colored stars next to the user name whereas the color corresponds to a certain range in the trust value. The trust update algorithm which determines the trust rating subtracts the number of negative from the number of positive ratings but takes into account only one rating per rater. Thus if user A rated user B on 6 occasions, the trust value of B is increased or decreased at most by 1. Since the ratings could be different, the trust value is increased by 1 if the number of positive ratings among the 6 is higher than the number of negative ratings. The trust value is decreased by 1 if the number of negative ratings is higher than the number of positive ratings. Otherwise, B's trust value is not changed in effect to the feedback from user A. Several surveys have shown, that this rating system contributes significantly to the success of eBay, even is vital to the functioning of this auction site despite its low complexity. Resnick [RZSL06] found also a monetary benefit for sellers with a high reputation, and notes in his experiments that buyers were willing to pay on average 8.1% more to a reputable seller than a new seller.

Another well-known example for centralized reputation mechanisms was implemented at the website of the online vendor Amazon[7]. As already noted in the recommendation system context, Amazon allows users to write reviews for the products offered at the Amazon website. As a second step, users reading the reviews can give feedback to the system whether a particular review was helpful to them or not. This feedback is accumulated at Amazon and used to determine the "best" or most trustworthy reviewers which are then marked as a "Top 500 reviewer", "Top 100 reviewer", etc. Trust is represented as an integer number as in the eBay case. This is mapped depending on the

[6]http://www.ebay.com/
[7]http://www.amazon.com/

score of other reviewers to the discrete steps for “Top-x”. The trust update algorithm takes into account a reviewer’s number of positive feedbacks and subtracts the negative feedbacks. In addition to this mechanism for product reviews, Amazon offers a direct rating of buyer and seller in their C2C marketplace. Whereas eBay combines ratings received as buyer and ratings received as seller into a single trust score, Amazon allows both buyer and seller to write a textual feedback for each other but restricts the trust calculation to feedback given by a buyer to a seller[8]. The rating consists of 1 to 5 stars which is also the representation of trust in the trust model. The update algorithm here simply takes the average of all received ratings. In addition to the star representation, Amazon also noted the percentage of positive ratings, which is obtained by a mapping of 5 or 4 stars to positive, 3 to neutral, and 2 or 1 stars to negative ratings.

The group of Patty Maes at the MIT media lab has been active in researching centralized reputation mechanisms. Sporas and Histos are to be named as being the most renowned systems they developed [ZMM99] so far. In Sporas, trust is modeled as a real number in the range from 0 to 3000. The trust update algorithm is a weighted average over all other users’ individual experiences with the trustee in question. Policies ensure, that user A may only be rated once by user B to prevent collusion attacks of 2 users giving each other perfect ratings repeatedly. Whereas Sporas is used to calculate global trust values for each user, Histos is designed to take the subjectivity of trust into account. In order to determine the trust of user A in user B, Histos tries to determine a rating path from A to B and – if this path is found with a certain maximum length – calculates this transitive trust. The action if multiple paths are found remains unclear from [ZMM99]. If no such path is found, the computation of trust falls back to the global Sporas trust.

Rahman et al. are working in the area of *trust development* based on *experiences* and describe in [ARH00] a trust model and algorithms about how trust can be created, distributed and combined. The representation of trust in the trust model is via a set of four defined states: *vt* (very trustworthy), *t* (trustworthy), *u* (untrustworthy), and *vu* (very untrustworthy). The trust update algorithm determine the corresponding state based on direct experiences of an agent and recommendations from other agents. The rather ad-hoc nature of some of these algorithms needs to be clarified further.

The work of Audun Jøsang et al. on his *Beta Reputation System* [JI02] and especially

[8]This is most likely due to the fact, that Amazon handles the payment processing for sellers which basically limits the risk of sellers to receiving unfair ratings. eBay, on the other hand, does not take over that responsibility.

the *subjective logic* [Jøs01, Jøs02, JG03] that it is based on has raised significant interest in the research community. Trust is represented in the trust model as a probability value with an alternative representation being described as probability density function (PDF) and a mapping being defined between both representations. The trust update algorithm of the Beta reputation system takes into account the number of positive and negative experiences and calculates trust based on belief calculus. Subjective logic defines operators to combine beliefs or take into account dependencies etc. We performed joint work with Prof. Jøsang on the computation of transitive trust [JGK03, JGK06] that is based on subjective logic. Therefore we refer the reader to Section 7.6.1 for more information on this topic.

Distibuted Reputation Systems

There is still a comparably small amount of *commercial work* in *distributed reputation systems*. Noteworthy is the Poblano project (see [CY01]), Sun's work on reputation in their JXTA peer-to-peer architecture. Poblano introduces a decentralized trust model with trust relationships not only between peers but also between peers and content (what they refer to as "codat", code or data). "Trust" in a peer is calculated here based on the content this peer has stored in addition to its performance and reliability.

Scientific work in *distributed reputation systems* has taken on momentum during the last few years. The trust modeling work of Rahman et al. highlighted in the previous subsection can be implemented in a distributed fashion as Karl Aberer and Zoran Despotovic mention in [AD01]. They are proposing a model where they focus completely on negative recommendations (complaints) to derive trust in an agent and describe distributed storage issues and trust calculation algorithms for their model. The trust model and trust update algorithms rely therefore on the number of complaints that have been reported in the system regarding the behavior of a certain agent.

The ReGreT system developed by Jordi Sabater et al. [SS01, Sab03] represents a reputation system which according to the authors uses direct experiences, witness reputation and analysis of the social network where the subject is embedded to calculate trust. The trust model represents trust as real numbers in the range of $[-1,1]$. The trust update algorithm computes trust based on the weighted average of made experiences that are in the same range.

Lik Mui et al. have developed a computational model of trust [MMH02] based on Bayes' theory. Also the Travos system developed by Patel et al. [PTJL05] relies on a probabilistic approach, much similar to the one of Beta reputation described in the centralized reputation system section. Travos is aimed at addressing the needs of agents as members of virtual organizations in a GRID context. Each agent maintains the level of trust in each of the system's other agents whereas this trust is represented as a beta probability density function.

Christoph Sorge and Martina Zitterbart describe the design for the Rebcon system [SZ06] that uses the concept of trust to determine, which users should be allowed to access certain information. They cite our EC-Web paper [KP03] and – among other things – correctly state, that our representation of trust specified as a real number in the interval of $[0..1]$ has no immediately visible real world meaning, as for instance would have "fully trusted" or "marginally trusted". The trust model they propose represents trust as a real number in the interval of $[0..1]$. The question of trust update algorithms was not addressed in [SZ06].

2.3 Summary

In this chapter we gave an overview on trust coming from two different angles: *social sciences* as the foundation of trust research, and *information technology*.

We started digging into the social aspects of trust by introducing the insights gained from the prisoner's dilemma, a game conducted by rational choice or game theorists, and *tit for tat*, an effective solution strategy for the iterated prisoner's dilemma. The main conclusion is, that rational individuals will not necessarily act cooperatively, even if cooperative actions benefit all group members. The following characteristics or aspects of trust were identified: Trust is directed *towards the future* and involves a certain *risk* that trust is betrayed. More clearly, trust is related to a *critical alternative* where the *potential damage* in case of betrayal may be higher than the *benefit* of honored trust. Trust is not a conscious strategy in most cases, but is *latent* in its nature. Trust is *subjective* and *asymmetric*, although *reciprocity* too is a key element for trust.

We discussed the reasons why people trust and the motivation why people want to be trusted. Major functions of trust are the *reduction* of (social) *complexity*, the *reduction* of (perceived) *risk*, and the *reduction* of *cost*. The *personal action space* is increased and *cooperation*, even *live* in general, are *improved*. The motivation for being trustworthy ranges from financial reasons, as a trustworthy merchant is expected to strike more deals, to pure altruistic reasons like the discussed goodwill view.

The ability to trust has to be *learned* already as an infant. Trust becomes in the following the result of a *learning process* based on *past experiences* in an *interaction history*. Important preconditions are the *durability* of a relationship with numerous *iterations* of positive experiences. Finally, *generalization* is an essential process for building trust: single points of experiences are generalized into a feeling of trust, existing knowledge in one context is generalized to other contexts, and experiences with a set of known persons are generalized into a feeling of trust in strangers.

The topic of trust appeared in information technology first in the area of *trust in cryptographic keys*. Trust in this context refers to being sure of the identity of the owner of a certain key. This challenge of binding identities to keys can be solved with digital certificates, whereas the question of trust is replaced by the question, whether a certain certificate authority is trusted to vouch for the claim that the link between identity and key is correct. We discussed the hierarchical PKI trust model as opposed to the web of trust of PGP.

The next notion of trust was introduced in the context of *trusted platforms*. Trust here refers to the expectation, that a (computing) platform will behave in a particular manner for a specific purpose.

Finally, we presented the area of *reputation systems*, which has the strongest ties to the social science view on trust. These systems collect, distribute, and aggregate feedback about participants past behavior and try to provide their user's with data allowing them to decide whom to trust. A reputation system contains the building blocks of *trust model* specifying the representation of trust, *trust update algorithm* specifying how trust shall be updated, *storage* for persisting inputs and outputs of the algorithms, and finally the *protocols* for communicating the data. Reputation systems can be categorized in *centralized* and *distributed* reputation systems, depending on whether the reputation information is stored in a centralized repository for all participants, or whether it is distributed over agents that reside on each participant's computer.

Chapter 3

UniTEC: An Architecture for a Distributed Reputation System

As mentioned before, the research results presented in this dissertation were developed by the author in the context of the UniTEC research project at the Distributed Systems Department of the Institute of Parallel and Distributed Systems of Universität Stuttgart, Germany. In addition to the scientific publications at various international journals and conferences, a prototype of the UniTEC system was developed to experiment and validate the theoretical concepts.

Several students were also involved in the realization of the prototype. Ralf Terdic contributed with his Diploma Thesis [Ter03] mainly to the data management and anonymous communication part. Jochen Widmaier worked with his Diploma Thesis [Wid03] on the identity management component. Fabian Aichele laid in his Student Thesis [Aic04] the first steps for the trust management component, which was in the following reworked in more depth by Ernesto Baschny during the course of his Diploma Thesis [Bas04].

In this chapter, we first describe several usage scenarios of UniTEC to illustrate the capabilities of and requirements for our combined reputation and recommendation system. We then move on to introduce the required models: trust model, knowledge model, system model, and our representation for recommendations. This is followed by a presentation of the interactions between the system components and models at the different participants. After that, we cover the architecture of our UniTEC agent and describe the functionality of the different components: the data management component, the identity management component, the anonymous peer-to-peer communication component, the peer-to-peer overlay component, and the trust management component. This is followed by an evaluation of the architecture described so far towards the posed requirements. Finally we summarize the findings of this chapter.

3.1 Use Cases and Requirement Analysis

In the following we describe the basic usage scenarios of a combined reputation and recommendation system. Starting from these we point out possible threats and the resulting requirements of the involved parties and their implications for UniTEC.

3.1.1 Usage Scenarios

Two basic usage scenarios can be immediately identified for a recommendation system, namely *publishing recommendations* and *requesting recommendations*.

An entity that is about to create a recommendation has made an experience with a second entity and intends to publish the gained knowledge. Therefore it has to be able to identify the target of the recommendation clearly and include this information in the recommendation so that requesters know without doubt what target is meant.

An entity A that needs advice about a target T will use a recommendation system to formulate a query for recommendations about T. In order to do that, A needs to be able to identify T uniquely and include this target identity in the query. The query is sent to the system and hopefully responses matching the query will be received. The system should be able to condense the data, accumulate the recommendations (if possible, e.g. in case of numerical ratings) and present the results to A.

3.1.2 Possible Threats

The following are a number of possible threats regarding the two basic usage scenarios:

T1 – Low acceptance. As for all online communities, the *startup problem* is a significant issue also for reputation systems. If the acceptance of the system is low, few people will participate in the scheme and invest to contribute high-quality content in form of recommendations. This obviously leads to requesters not finding suitable information, which again decreases their motivation to contribute as recommenders themselves. Despite the fact, that this issue is out of scope of this thesis, we want to note the importance of participant motivation for building a reputation system and establishing a well-functioning community.

T2 – Low quality. Recommendations are retrieved that are potentially unrelated to the query and/or that might be of low quality.

T3 – Malicious recommenders. Anyone can join the reputation system to provide consciously malicious content.

T4 – Recommendation tampering. Recommendations can be modified en route to a requester.

T5 – Reputation agent tampering. If the software component handling the request processing and recommendation collection can be tampered with it might fall under the control of an attacker. If this is the case, the reputation data itself has become unreliable and the attacker might present arbitrary content to the requester with forged trust statements.

T6 – Unauthorized access to recommendations. Not all recommendations should be accessible to all requesters.

T7 – Unclear legal situation. There is no legal redress if the system allows false recommendations to be provided and using these causes business loss.

T8 – Uncertainty. People are too worried about their requests and recommendations being attributed to them personally to want to engage in the system. Cynical persons could hesitate calling this issue a threat to reputation systems, especially when considering the willingness of many people to participate in payback card schemes. Nevertheless, we can hope, that the privacy awareness will increase and this uncertainty has to be addressed.

3.1.3 Requirements of the Participants

We now list the participant's requirements towards a peer-to-peer reputation system, based on our analysis above. For each requirement we denote in brackets the main threats that this requirement addresses.

Requester's Requirements

The requester's requirements are as follows:

R1 – Recommendations need to fit the query (T1,T2). As mentioned before, the target of a recommendation has to be uniquely identifiable in order to match a query.

R2 – High quality recommendations (T1,T2). This requirement seems self-explanatory, however it leads directly to several consequence requirements (R3 – R6)

R3 – Feedback mechanisms (T1–T3). It is imperative to be able to *provide feedback* to the system about the quality of the recommendation. In case the experience of the requester does not match the one of the recommender, this negative feedback can be used to prevent this requester from receiving further recommendations from the unsuitable recommender. Therefore, wrongful recommendations can be detected.

R4 – Recommender identification (T1–T3). Recommendations need to be *linked to their recommenders* in order to identify the recommender when processing the feedback of the requester.

R5 – Protection of reputation data (T1,T2,T5). The reputation information describing the reliability of recommenders must be protected against unauthorized modification.

R6 – Up-to-date information (T2). Recommendations themselves must contain a creation timestamp to identify their timeliness. Furthermore, especially in highly dynamic areas, for instance recommendations about stock options, it is important to receive new recommendations as quickly as possible after they are published.

R7 – Fine-grained trust modeling (T2). An entity might choose to trust another entity only in certain areas whereas doing not so in others. Therefore instead of using just one trust value per entity it is necessary to model a fine-grained set of categories, where this entity can be trusted or not. To our knowledge this requirement is not fulfilled in the related work to the extent presented in this dissertation.

R8 – Integration of real-world trust (T2,T3). As a starting point for the system, the requester needs the possibility to include existing real-world trust towards colleagues, friends and acquaintances.

R9 – Controlled disclosure of a recommender's real identity (T7). A technological method for finding a recommenders' identity must be provided given sufficient legal justification.

R10 – Privacy protection (T8). The system must protect the requesters privacy during the recommendation request process. If the privacy were not protected, other participants might collect information about the posed recommendation requests, which could be accumulated over time into a detailed user profile.

Recommender's Requirements

The recommender's requirements are as follows:

R11 – Ease of use (T1). This requirement is aimed at limiting the recommender's disturbance by the recommendation system as much as possible. From the requester's point of view it certainly would be interesting to be able to contact good recommenders with inquiries whereas those popular recommenders most likely would be overwhelmed with the number of requests. Therefore we store the recommendations and make them accessible for requesters but remove the possibility for requesters to contact recommenders directly.

R12 – Ensure the authenticity of recommendations (T4). The system has to prevent giving out recommendations in another recommender's name. Additionally, it must not be possible to secretly alter existing recommendations without the creator's authorization. To solve this issue, we add a digital signature to the recommendation. In order not to compromise the recommender's privacy, we need a digital certificate that is not bound to the real identity but instead to the pseudonym that the recommender is using.

R13 – Control over recommendations (T5,T6). Recommendations may contain sensitive information. Therefore, the recommender needs to be able to limit access to these sensitive recommendations to a certain group of requesters. Identity management is needed to handle authentication and authorization tasks.

R14 – Privacy protection (T8). *Control over collection of personal data* is significant also for the recommender. To counter the danger of abusing the system for detecting buying habits and generating detailed user profiles (e.g. for direct marketing purposes), we could anonymize the recommendations. However this conflicts with the aforementioned requirement to bind a recommendation to its recommender. Since the identity is the only place where to attach a reputation, we write recommendations using one or more pseudonyms that allow the recommender to build a reputation, but are not linkable to the real-world identity. We suggest to use one pseudonym for each area of interest of the recommender to increase the effort of establishing a link between real identity and pseudonym.

3.2 Modeling

In this section we identify the models necessary for implementing a distributed reputation system, namely the trust model, the knowledge model, and the system model, in addition to a brief description of the potential content of the communicated data items.

3.2.1 Trust Model

In order to allow the reader to understand the UniTEC architecture, we provide an overview of the trust model. The detailed description of the trust model follows in Chapter 6.

Definition 1 (Trust context area) *In general, peoples' trust is not all-encompassing. Instead, people trust in a fine-grained manner depending on the decision they are about to take. This area that person A might trust person B in is what we term trust context area. Other words in the literature for trust context area are* trust scope *or* trust purpose. *As an illustrating example, Alice might trust Bob to repair her car, but not to babysit her child.*

Due to the diverse nature of trust, our model consists of a set of interdependent *context areas*. Each context area has one *trust value* and one corresponding *confidence vector* for each recommender. Thus I trust Alice with a certain trust value and with a certain confidence in a context area, which is most likely different from my trust and confidence in Bob in that area.

We define the trust values in the range from 0..1 with 0 indicating either no previous experiences or just bad experiences with that entity in the context area and 1 indicating the maximum trust. We assume that having no previous experiences is similar to only bad experiences for the reason that it is relatively simple to switch to a new pseudonym as soon as the old one got a bad reputation. Therefore we set the initial trust in each context area to 0 although this can be set manually to a different value for known trusted entities to transfer the real world trust into the system.

The confidence vector stores meta-information used to judge the quality of the trust value and contains the following entries:

- number of direct experiences in that context area
- number of indirect experiences (from context areas influencing the one in question)
- trail of the last n direct experiences (n depends on the storage capacities of the system the agent is running on) with the associated recommender confidences
- black list flag

20 good experiences have more impact than just one good experience, therefore the total number of experiences in the context area and the derived information from other categories are valuable confidence indicators. Keeping track of a certain number of previous experiences with an entity enables the user to better understand a certain trust level and make on-the-fly trust calculations (e.g. by adjusting parameters of the trust update algorithms). The black list flag is set by the user to prevent recommendations from the selected identity in the context area in question to be taken into account at all.

3.2.2 Knowledge Model

What we refer to as knowledge model is a way of creating a local profile at each participant about “who knows what” including each participant’s own knowledge. Hence it is used for advertising one’s expertise areas to other participating entities of the system and keeping track of the expertise of others.

The *own expertise* in a participant’s knowledge model consists of a set of the number of own recommendations stored in each context area, as depicted in Fig. 3.1. This advertisement information is communicated (see Subsection 3.3.1). However, for privacy reasons not all context areas are communicated by the recommender as such. Instead the content is published in *knowledge parts* with each part being covered by

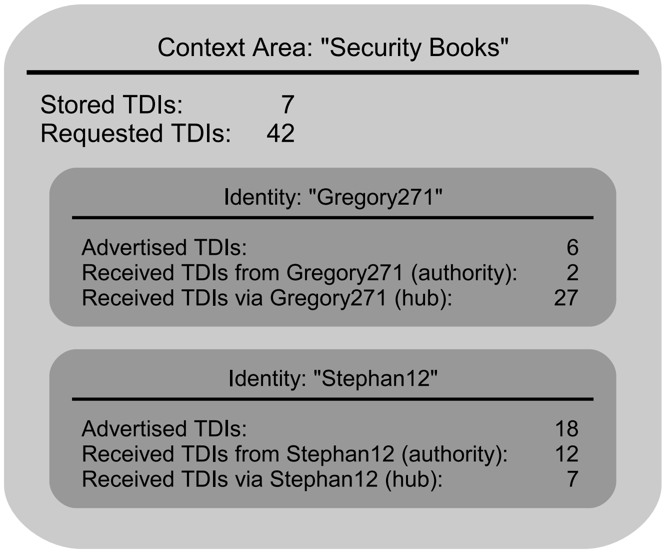

Context Area: "Security Books"
Stored TDIs: 7
Requested TDIs: 42
Identity: "Gregory271"
Advertised TDIs: 6
Received TDIs from Gregory271 (authority): 2
Received TDIs via Gregory271 (hub): 27
Identity: "Stephan12"
Advertised TDIs: 18
Received TDIs from Stephan12 (authority): 12
Received TDIs via Stephan12 (hub): 7

Figure 3.1: Sample entry in the *knowledge model* for context area “security books”

one pseudonym. Ideally, the knowledge parts should not overlap to protect the strength of the pseudonyms and prevent pseudonym linking.

In order to evaluate the *expertise of others*, the number of own recommendation requests is stored for each context area. In addition to that, each recipient of a knowledge part stores the provided information under its context areas and identity in his or her knowledge model and includes a confidence vector (different from the one in the trust model) to each (potential) recommending identity that contains the total number of recommendation responses received via this identity (from other recommenders) and the number of responses with the recommendation being published by this identity (as recommender) in the specified context area. This information allows to keep track of the value of the identity as a *hub* as well as an *authority*.

Definition 2 (Authority) *An authority is an entity that is expected to have a deep knowledge in the context area in question and therefore should be able to provide helpful recommendations.*

Definition 3 (Hub) *A hub is an entity that has not necessarily a high expertise in a certain field, but knows many experts and potential recommenders in that context area. A hub therefore is very well connected in the network.*

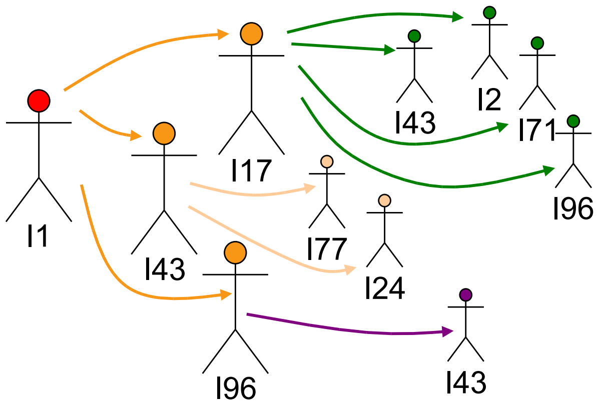


Figure 3.2: Neighborhood view of identity *I1* for a certain context area *C*

3.2.3 System Model

The system model we base our work on is that of a *peer-to-peer* system with distributed *stationary* or *mobile nodes* that have communication capabilities with other nodes. On each of those nodes resides one or more *agents*, each in context of a certain *entity*, which will be in most cases the end-user of the system. Each user employs different *identities* which might range from the user's real identity to various *pseudonyms* in order to give or request recommendations up to complete anonymity for requesting recommendations. Each agent stores a separate instance of the *trust-* and *knowledge model*.

As will be described in the following section in more detail we introduce a *neighborhood* for each entity as overlay over the real network topology for each context area that takes into account among other things the developed trust in the various identities (*I17, I43,* etc. in Fig. 3.2) in the context area in question. There is not necessarily connectivity to all nodes of the neighborhood at every given point in time. The reachable members of the corresponding neighborhood are queried for each recommendation request.

Definition 4 (Neighborhood) *For an entity E and a trust context area C, the neighborhood* ONet(context area C, entity E, level L) *is the set of identities that can be reached from* E *in* L *hops. The set* ONet(C, E, 1) *is the set of identities directly connected to entity E. The membership of an identity* I *in this local view or level 1 view of the neighborhood of an entity* E *is determined by an algorithm described in Section 3.3.4 that relies on two inputs:* E's *trust in* I *(stored in its trust model) and* I's *advertised expertise (from* E's *knowledge model entry about* I*). Generally:*

$$ONet(C,E,L) = \bigcup_{\forall E' \in ONet(C,E,1)} ONet(C,E',L-1) \tag{3.1}$$

Example (see Fig. 3.2): Let entity E act as identity *I1* in the context area in question. The level 1 view of entity E respectively identity *I1* on the neighborhood is initially a tree with height one with root *I1* and leaves *I17*, *I43*, and *I96*, thus *ONet(C, I1, 1) = {I17,I43,I96}*. The three entities in the level 1 neighborhood of *I1* however also build each a height-one tree and therefore form the level 2 neighborhood *ONet(C, I1, 2)* for *I1*. Obviously, loops are possible in this structure, therefore we talk about a *directed graph* instead of a tree.

Observation: Due to the asymmetry of trust, the neighborhood observed by *I1* might contain or be contained in the neighborhood observed by *I2* or might be a completely different "island of trust".

3.2.4 Recommendations or Trusted Data Items (TDI)

Recommendations in UniTEC consist of three main components: *recommendation data*, *recommender data* and a *metadata*.

Within the recommendation data component, the specification of the context area that this recommendation belongs to is mandatory. In addition to that, the component is flexible in that it may hold an arbitrary number of different fields at the discretion of the recommender to specify the recommendation target and content. Each field is defined by the field name, the field type and the field content. The currently supported field types include Boolean (e.g. "we recommend the target, or we do not"), Integer and Float Numbers (e.g. "the product has a quality of 70%"), Binary Objects (e.g. images of the recommendation target) and Strings of arbitrary length (e.g. to specify the target or a textual review).

The recommender data component contains the recommender's virtual identity and his or her confidence in the statement. When discussing the system with colleagues researching in this field we found it to be important to add this *confidence value* to the rating, specifying the recommender's own confidence in the given statement. The confidence influences the impact of this recommendation for the trust update when processing the requester's feedback (see Subsection 3.3.4).

The metadata component contains a recommendation identifier and the time of the recommendation creation to facilitate recommendation handling and timeliness checks. Additionally, an access control list (ACL) allows to control access to the recommendation data.

In order to prove the authenticity of the recommendation, a *digital signature* is added under the appropriate pseudonym that the recommender uses for creating recommendations of the context area in question.

The structure for our recommendation is generic in the sense that we can issue arbitrary signed statements which can be published and queried by the system. The result is a general reputation system that handles signed *trusted data items* and provides support for judging their quality. The terms recommendation and trusted data item (TDI) are in the following used interchangeably.

3.3 Interactions

In this section we focus on the interactions between the various components and models of the system at the different participants necessary for publishing and locating fitting recommendations. The recommendation location process is divided into the five subtasks of *disseminating* the request, *collecting* responses, *processing* the responses, *organizing* the results and *providing* and *processing feedback* for the system.

3.3.1 Publishing Recommendations and Advertising Knowledge

An entity may publish information as a trusted data item. This information can refer to experiences with a product or a service, but it may in general contain arbitrary statements that it wants to make available either publicly or solely to selected identities. We described the structure of the TDI in Subsection 3.2.4.

The recommender creates the TDI on the own UniTEC agent by creating the appropriate fields and entering the content information. Finally, the TDI is assigned to a certain context area with an associated pseudonym. When this is done, the agent adds the timestamp to the TDI and creates a digital signature on the TDI with the corresponding pseudonym's private key. Finally, the TDI is stored within the agent's local database (see Section 3.4) and the knowledge entry representing the number of TDIs for a certain context area is increased. Access to the recommendation can be protected by an authorization concept relying on access control lists (see Section 3.4).

Before an entity is able to request a recommendation it needs information regarding the knowledge of other identities with respect to stored recommendations per context area, as pointed out in Subsection 3.2.2. This issue can be addressed via two different approaches: *advertisement messages* and *knowledge providers*.

The knowledge model is updated when a user creates a new recommendation under a certain pseudonym in a certain context area. When the delta of the knowledge model communicated for a certain pseudonym in the last advertisement message has reached a specified threshold, a new advertisement message is created. This message is signed with the private key of the corresponding pseudonym and contains the virtual identity together with the number of recommendations stored in each context area that this pseudonym is responsible for. Possible recipients of this message include interested identities who have subscribed for knowledge updates but also the aforementioned knowledge providers.

Knowledge providers are UniTEC agents in a dedicated role for storing the expertise areas of pseudonyms. From a logical point of view, these knowledge providers can be seen as super nodes responsible to overcome the startup problem of not knowing any pseudonyms nor their areas of expertise. When a new UniTEC agent is set up, it can contact one or more knowledge provider and download the pseudonym and knowledge data available. A UniTEC agent that is already in heavy use can subscribe to reputable pseudonyms for knowledge updates directly.

One limitation of the current UniTEC prototype is that we have not implemented automatic support for advertising knowledge. Instead, the knowledge areas can be added manually to foreign pseudonyms within the identity management component.

3.3.2 Disseminating Requests

In order to understand the dissemination of the request it is important to keep in mind the recursive structure of the neighborhood introduced in the previous section. The request message contains the *identity*[1] of the requester, a *request identifier*[2], the *rec-*

[1]Once recommendations fitting the target and filter are found, they are directly sent back to the requester.

[2]This identifier allows to efficiently check, whether a request has been processed already. The reader should note that a single recommender can be reached via completely different paths due to the request

ommendation target[3], the *trust chain* and a *hop counter*. The neighborhood size and the hop counter control the number of identities reached by a certain request.

The *trust chain* is used to compute the trust of the requester in the recommender and is built during the request dissemination. At *intermediate identity* I_i, it contains a list of intermediate identities I_j ($1 \leq j \leq i$) including the trust T_j of I_{j-1} in I_j with I_0 being the requester.

Once the request message is built by the requesting agent, it is sent to the agents of all reachable members of the level 1 neighborhood for the corresponding context area. Each recipient I_i then processes the *algorithm* described in Fig. 3.3. A new copy of the request is created for each of the identities in the neighborhood for the request's context area. Each copy differs from the others through a new signed chain link that contains the trust of I_i in that neighborhood member in addition to the identity of I_i and its public key.

Note: We should stress that this presented approach for building the trust chain, which we implemented in the UniTEC prototype, has limitations which we point out in more detail in Chapter 7. The focus of this algorithm is to find the strongest trust path from the requester to the recommender, without taking into account other available information, especially confidence or negative trust.

3.3.3 Collecting Recommendation Responses

Each identity *I's* agent capable of fulfilling the recommendation request (therefore with one or more fitting recommendations in store) creates a *recommendation response message* that contains a request identifier, the stored and digitally signed recommendation and the trust chain whose last link is the trust of I in the recommender. This recommendation response is sent directly back to the requester.

A recommendation is generally not only available at the recommender's agent but can instead be stored at any interested participant's agent. Furthermore those recommendations are not necessarily all up-to-date e.g. due to network partitioning. The reader should note that the same recommendation (same recommender, identifier and version)

dissemination algorithm.

[3]The recommendation target consists of the context area of the target plus a textual filter specifying the target more closely. The filter in the current prototype, as described in [Ter03], supports full-text queries that can contain multiple elements combined via boolean operators. The elements may refer to either the whole TDI or a single field of the TDI (if field names are known or agreed upon beforehand).

```
calculate trust of Req. in Ii via the current trust chain
if not((request already processed) AND (with higher trust)) {
   if (suitable recommendation available) {
      create recommendation response
      send recommendation response back to Req.
   }
   decrease hopcounter
   if (hopcounter > 0) {
      foreach (member Ij of ONet-1 of Ii) {
         (* create set of new requests *)
         create copy of received request
         insert digitally signed trust statement:
         {
            trust of Ii in Ij
            (pseudonymous) identity of Ii
         }
         send request to Ij
      }
   }
}
```

Figure 3.3: Directed dissemination algorithm (simplified)

can be received via completely different paths through the neighborhoods of the intermediate recommenders leading to different trust chains. What needs to be done here is to *evaluate* the different trust chains from requester to recommender and produce *one* trust value that can be attached to each received recommendation.

After the trust in the recommender has been computed, all outdated recommendations (same recommender and recommendation identifier, but older version) are discarded and just the most current version of each recommendation is kept.

The organization of the results depends on the type of rating information in the recommendation in question. Recommendations with textual reviews or multiple rated attributes are organized by their trust value and highly trusted recommendations are presented first to the requester. For the same trust value, recommendations with a higher recommender confidence are presented first. Binary ratings of the same value can be accumulated in discrete groups depending on the trust value. For percentage ratings not only the recommendations but also the ratings themselves are categorized in that manner and presented to the requester as a matrix.

3.3.4 Handling Feedback

The step of handling feedback can be divided in the three sub-steps of *collecting the user feedback*, *updating trust* and *updating the neighborhood*.

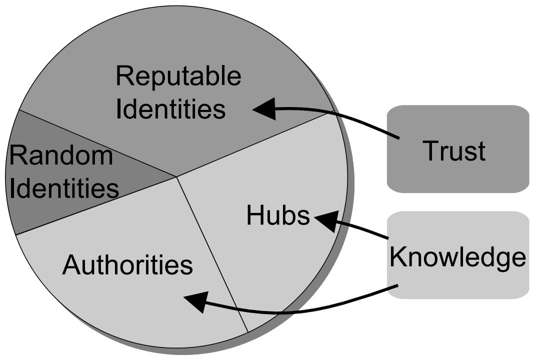


Figure 3.4: The neighborhood update selects identities from the described four categories.

The requester has to state whether a certain recommendation was useful in his or her view and which one was not. For binary ratings it is sufficient to specify the “right” answer whereas for percentage values the “correct” range has to be defined. For textual reviews or multiple attribute ratings the process of collecting user feedback cannot be automated and every single recommendation has to be rated individually. With this rating the requester has made an *experience* with a recommending identity which is either a *good* or *bad* one. For each identity the experience and the recommender confidence are added to the trust model and the number of direct experiences is increased in the appropriate context area.

This feedback gained in the previous step is used in the following to update the trust in the recommenders accordingly. UniTEC supports various trust update algorithms, as described in Section 6.5. In general, trust update algorithms specify, how to compute a certain trust value from a series of inputs. In the UniTEC context, these inputs include the experiences made with this recommending identity and further parameters to control the update process.

After the trust values have been updated it is necessary to update the identities in the level 1 neighborhood as well. According to Claus Offe [Off01], it would be irrational to solely focus on already trusted entities and thereby abandoning the possibility for gaining further experiences from other sources. As we pointed out in [KR03], we choose our neighborhood by selecting a certain number of identities of each of the three types (not disjunctive) as depicted in Fig. 3.4: *reputable identities* (high trust), *confirmedly experienced identities* (high knowledge) and *random* identities.

The selection of reputable recommenders is done by taking the *n* recommenders with the highest trust and a certain minimum number of experiences. From the remaining recommenders which have not been chosen as reputable recommenders, the confirmedly experienced identities are selected. Half of these confirmedly experienced identities are chosen from the *hubs* by taking those identities that appeared most frequently in the trust chains. The second half is chosen from the *authorities* from these recommenders that most recommendations originated from. A certain proportion of identities are randomly chosen from all the remaining ones with advertised knowledge in the context area. This is necessary to allow identities not yet popular (since trust is initialized at the minimum level) to be queried and gain a reputation.

This strategy of updating the neighborhood ensures that recommendations can be received with a sufficiently high transitive trust (via the reputable recommenders), while at the same time being open for getting to know new potential recommenders. As Ziegler and Lausen discuss in [ZL04], such an approach finds groups of users with a similar mindset over time.

3.4 Architecture

UniTEC is a completely decentralized reputation system that consists of a peer-to-peer network of agents residing on nodes with communication capabilities. We argue in Chapter 4 that the system is greatly strengthened if the computing platform that the agent resides on is a *trusted platform*, but this is not a prerequisite as such. The UniTEC agent does not follow a strictly layered approach, but instead consists of several components as described in Fig. 3.5. We describe in this section the basic functionality offered by each of the agent's components and come back to a detailed description in later chapters.

3.4.1 Data Management Component (DMC)

The *data management component* provides a secure storage framework to the other UniTEC components. It primarily consists of two parts: *local storage* and *remote storage* that each have the option to store data either in memory or use a database persistence[4]. The local storage is used directly by the agent's components to store

[4]However, currently we use database persistence solely for the local storage and not for remote storage. This is the case due to the fact that the data contained in the remote storage is updated in regular time intervals and is most likely outdated after an agent has been offline for a while.

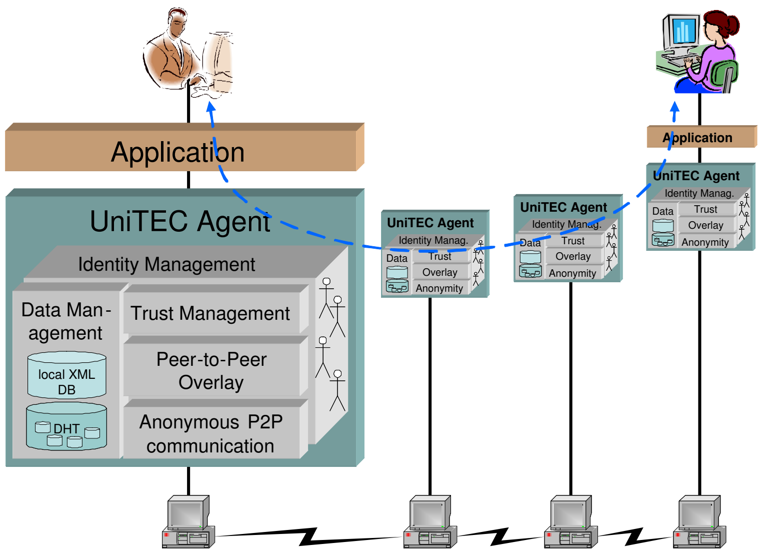


Figure 3.5: Architecture of the UniTEC system. Nodes with communication capabilities host UniTEC agents that provide trust management to applications respectively their users.

information for its user. The remote storage, on the other hand, is required due to the fact that the agent takes part in a distributed hash table (Subsection 5.3.2 provides more information on our usage of DHTs). It is used to provide storage capabilities for the overall UniTEC system.

The DMC's local storage stores primarily information from the trust and knowledge models (see Section 3.2) as well as the trusted data items that have been created by the user of the agent. Additionally, a limited number of recommendations is stored that have been received as results from recommendation requests[5]. This increases the availability of high-quality recommendations in the system.

Access to trusted data items can be controlled via an *Access Control List* (ACL). The ACL consists of two parts: an Allow-ACL and a Deny-ACL (as described in [Ter03]). If both lists are empty, no access control is performed for this TDI. If the Allow-ACL contains entries, access to the TDI is restricted to these pseudonyms and the Deny-ACL is ignored. If the Allow-ACL is empty, access to the TDI is granted to all pseudonyms but these contained in the Deny-ACL. The ACL is part of the TDI and also covered by the TDI's signature to prevent unauthorized changes. Obviously, this mechanism can offer only limited security against attackers changing the agent's code or copying the TDI content to a new TDI without ACL protection. Our approach presented in

[5]Note that the trust in the recommender of these recommendations is available from the trust model.

Chapter 4 provides enhanced security features to overcome these issues.

During runtime, the trusted data items are represented as *JavaBeans*. For *transport*, the Beans are serialized via *Java Object Serialization* for efficiency reasons. For *persistence*, the Beans are serialized to XML via the JOX serialization framework[6]. The XML documents containing the TDIs are persisted in the XML database Apache XIndice[7].

XIndice allows queries to the database via the query language XPath. XPath, however, requires knowledge about the structure of the TDIs, which is not necessarily available at a remote requester. In order to facilitate queries for recommendations, a full text search on the TDIs was implemented based on the open source framework Lucene[8]. Lucene supports indexing documents and performing full text queries over the generated index. As pointed out in Subsection 3.3.2, the filter to query TDIs allows to combine multiple query elements combined via boolean operators. The elements may refer to either the whole TDI or a single field of the TDI (if field names are known or agreed upon beforehand).

3.4.2 Identity Management Component (IMC)

For privacy reasons (see Chapter 5 for more details), each user employs multiple virtual identities or pseudonyms instead of his or her real identity when interacting with the UniTEC system. In general, a single pseudonym is responsible for a certain part of the context areas in the trust respectively knowledge model, e.g. pseudonym A is responsible for vehicle-related recommendations whereas pseudonym B handles book recommendations. The *identity management component* provides support for creating and removing pseudonyms and assigning context area responsibilities.

In general, the IMC stores data about two different kinds of pseudonyms in the DMC: *own pseudonym*s and *foreign pseudonyms*. Each own pseudonym is created on behalf of the UniTEC agent's user and has an public and private key pair (1024 bit RSA) to secure communication with that pseudonym. Foreign pseudonyms are made known to the IMC through advertisement of expertise[9] and recommendation responses. Obviously, the private key is unknown for foreign pseudonyms.

[6]`http://www.wutka.com/jox.html`
[7]`http://xml.apache.org/xindice`
[8]`http://jakarta.apache.org/lucene`
[9]This is a possible enhancement for the UniTEC prototype which supports currently manual expertise advertisements.

Both kinds of pseudonyms have an associated globally unique identifier (see Subsection 5.3.1) and a locally unique name for improved ease of use. For own pseudonyms, this local name can be specified by the user upon pseudonym creation. For foreign pseudonyms, the local name is selected randomly from a list of names when the pseudonym is made known for the first time. Additionally, the IMC allows the secure intentional disclosure of the link between a pseudonym and a real identity (represented by a X.509 certificate) for selected highly trusted entities.

Protecting the strength of the issued pseudonyms is a crucial and challenging task. Several dangers to pseudonymity are identified for recommenders (rec.), requesters (req.) and intermediaries (int.) and are addressed in the following:

Globally unique identifiers for communication (all): We use multiple pseudonyms to limit profile building to an acceptable level for the recommender. Despite this fact, these pseudonyms can be related to each other depending on the used communication mechanism (e.g. via the IP address). This issue is addressed by the anonymous peer-to-peer communication component described in the following subsection and in depth in Chapter 5.

Same request filters for different context areas (req.): If the context area for a certain request is not clear, a requester may specify various requests in multiple context areas that are covered by different pseudonyms. These pseudonyms can be linked, if the request filter is similar. To cope with this problem, a history of past requests needs to be persisted. Whenever a new request is posed, similarity checks are performed to compare the request with the history. In case the request is too similar to the history but resides in a different context area, it cannot be fulfilled[10]. A local requester can be notified accordingly.

Similar trust in trust chain (int.,rec.): The association between the pseudonyms and their context areas is performed by each user himself or herself. Therefore, it is possible that an intermediary I_{n+1} might have associated categories to a single pseudonym whereas intermediary I_n has chosen different pseudonyms. If several requests are posed in these context areas, intermediary I_{n+1} could recognize that the last link in the trust chain of each of the requests contains exactly the same trust in I_{n+1} but from different pseudonyms. Therefore, the pseudonyms from I_n are exposed to I_{n+1} and to the intermediaries of the neighborhoods of I_{n+1} in the context areas in question.

[10]This is a possible enhancement for the UniTEC prototype.

Similarities in recommendations (rec.): Our model for recommendations is flexible and allows various textual, binary and numerical fields. The recommendation content might directly contain sensitive data (e.g. addresses, phone numbers, etc.) that allow to tie a pseudonym to a real-world identity. In addition to this obvious threat, the content of different recommendations in different categories handled by different pseudonyms of the same user might have similarities (e.g. common text blocks or images, but also typical mistakes of a certain user) that could lead to the linking of the pseudonyms. In the diploma thesis [Wid03] of Jochen Widmaier, experiments have been performed to measure the textual similarities, e.g. with the longest common subsequence and the Greedy-String-Tiling algorithm. The results show that Greedy-String-Tiling allows to identify possibly incriminating text blocks to limit the danger of linking pseudonyms through textual similarities.

3.4.3 Anonymous Peer-to-Peer Communication Component (ACC)

The *anonymous peer-to-peer communication component* provides basic anonymous messaging capabilities between digital pseudonyms to the other UniTEC components while hiding the link between pseudonyms and their owner's real identity.

At the core, this is achieved by extending David Chaum's Mix approach with concepts for securely publishing and retrieving routing headers for pseudonyms in a distributed hash table, extending the Mix cascade contained in the routing headers before usage, and discovering available Mixes. In the first implementation, the communication managed by the ACC used to be based on Apache XML RPC between the agents. This was changed for performance reasons to Java object serialization via plain TCP in the second version of the ACC.

A detailed description of the functionality of the ACC can be found in Chapter 5.

3.4.4 Peer-to-Peer Overlay Component (POC)

The *peer-to-peer overlay component* allows the construction of application-specific overlays on top of the generic UniTEC overlay formed by the existence of the agents. The application-specific overlays are formed by the set of context areas used by each application, whereas each context area is associated to a specific neighborhood. Therefore the POC is responsible for *pseudonym-to-pseudonym communication* whereas communication managed by ACC internally occurs on an agent-to-agent level.

The POC performs two main interrelated tasks: neighborhood maintenance and recommendation request processing. The context-area-specific set of targets that forms the neighborhood is updated as described in Subsection 3.3.4 upon certain discrete events: after knowledge advertisements have been received, after TDIs have been received, and again after user feedback has been received on the quality of TDIs respectively their recommenders. The trust management component (TMC) responsible for maintaining the trust and knowledge models notifies the POC upon each of these events, such that the POC can update the neighborhoods accordingly. Once the agent has received a request for a certain recommendation, as pointed out in Section 3.3, it performs several checks to see whether the request shall be processed further:

Check 1: Access the trust chain in the request message and check whether the last link in the chain points to a local pseudonym that is managed by the IMC and responsible for the context area of the request. If this is not the case, no local responsible pseudonym can be found and the request is discarded.

Check 2: Validate the digital signatures of all the links of the trust chain. If one or more invalid signatures are found, the trust chain is broken and the request is discarded.

Check 3: Contact the TMC to calculate the transitive trust of the requester in the current local pseudonym. If the request has not been processed before (tested via the request identifier) or if it has been processed but with lower trust, the new trust is stored and processing continues. Otherwise, the request is discarded.

After these checks on the request message are passed, the POC determines, whether suitable recommendations are stored in the DMC. This is done by evaluating the query contained in the request on the TDI index created with Lucene (see Subsection 3.4.1).

Found TDIs that originate from the queried local pseudonym are sent back directly to the requester in a single response message (see Subsection 3.3.3). For this message, the trust chain is finalized already. Found TDIs that originate from other (non-local) pseudonyms are sent back in several response messages, one for each non-local pseudonym, whereas the trust chain is expanded with a link containing the trust of the local pseudonym in the respective non-local pseudonym. Sending multiple messages becomes necessary because of the structure of the response message that contains a single dedicated trust chain.

Finally, if the hop counter in the request message has not yet reached 0, the request is disseminated to the neighborhood as described in Subsection 3.3.2. Each request message sent to a neighborhood member contains a trust chain that is expanded with a link containing the trust of the local user in that particular member.

3.4.5 Trust Management Component (TMC)

The *trust management component* handles all trust management related tasks in the UniTEC agent. The three main activities are *evaluating the user's transitive trust* in the TDI issuers after receipt of the TDI responses, *updating the user's trust* in the TDI issuers after the feedback step, and finally handling and *updating expertise* information.

A successful evaluation of the transitive trust includes – similarly to the process described in Subsection 3.4.4 – verifying each of the digital signatures on the trust chain links from the requester to the final TDI issuer. The trust statements inside the trust chain are evaluated only if each signature is valid to compute the transitive trust in each TDI-issuer. Whereas previously, only a single trust chain had to be evaluated up to that intermediary, the situation here is different such that multiple trust chains from the requester to the recommender may exist and are evaluated here. A prerequisite for this is a generic representation of trust (see Chapter 6) since only with that it is possible to combine all the individual trust statements sensibly and independently of the local trust models used at each intermediary.

Furthermore, the TMC keeps track of the user's trust in each pseudonym that she or he has been in contact with. More concretely, it stores trust in the DMC's database according to the specified trust model. The TMC also updates the trust in these pseudonyms upon receipt of user feedback regarding the quality of the received TDIs according to the trust update algorithm specified in the user's preferences. This trust update (see Section 6.5) influences the neighborhood selection the next time a query is received for the trust context in question.

Updating the expertise of pseudonyms in the knowledge model is independent of the active user feedback step. The authority rating of each received TDI's recommender is increased by 1. The hub rating of all intermediaries contained in the trust chain(s) of each TDI is increased by an amount corresponding to the distance of the intermediary to the recommender. Consider the trust chain $Requester \rightarrow I_1 \rightarrow I_2 \rightarrow I_3 \rightarrow Recommender$. In the current implementation, the hub rating increase is halved with each step away from the recommender. Thus, the hub rating of intermediary I_3 is increased by 1, the hub rating of intermediary I_2 in increased by 0.5, and the rating of I_1 is increased by 0.25. Although an intermediary can appear in multiple trust chains of a single TDI, his or her hub rating is only updated once per TDI with the maximum of the individually computed increments. This approach ensures that intermediaries connected to many recommenders eventually appear in the immediate level-1 neighborhood of an entity for a certain context area.

3.5 Evaluation

We evaluate in this section the architecture described up to now against the requirements posed in Section 3.1.3.

The *directed dissemination algorithm* controls how the POC forwards a recommendation request towards potential recommenders. This approach in general fulfills the ease of use requirement (R11) by not contacting the recommenders directly but only retrieving TDIs from their agents. During the dissemination, the usage of *Lucene* combined with the *query filters* at each agent ensure that TDIs fitting the query are found if they exist at a recommender (R1).

The *access control mechanisms* at the DMC allow to explicitly grant access to the TDIs to certain pseudonyms respectively in case of the exclude list to deny this access to certain pseudonyms (R13). They cannot protect against an entity that has already access to the recommendation stripping the ACL from it or publishing its content itself.

Via the usage of *digitally signed TDIs*, the *integrity* and *authenticity* of the TDIs are protected (R12). Furthermore, the TDIs contain a timestamp which denotes their time of creation. Therefore, a requester can judge the *up-to-dateness* of a received data item (R6). The *quality of TDIs* is ensured via the the *feedback* and *trust update* mechanisms offered by the TMC (R2,R3). Through the use of interdependent trust context areas, the TMC supports fine-grained trust modeling and furthermore allows to initialize trust for known entities manually (R7,R8).

The IMC manages the identities of own and foreign pseudonyms whereas digital signatures on the recommendations link them to their recommenders (R4). Through the use of extended destination routing, the ACC protects the privacy of both requesters and recommenders and hides the link between their pseudonyms and the real identity (R10, R14).

On the downside, we see that in the architecture presented up to now exists no specific protection of the reputation data respectively the UniTEC agent on a certain user's platform as such (R5). Therefore, were the platform to be compromised by a virus or hacker, the results presented to the user could be entirely forged. The agent would be no longer trustworthy.

The privacy protection mechanisms of the ACC do not allow a recommender's real identity to be disclosed (R9). Even with legal justification due to financial loss caused by wrongful TDIs, the real address of a recommender lies in its pseudonym's routing header[11]. There it is hidden in layered encryption through a chain of mixes, whereas each of the mixes would need to be forced to disclose the next layer. Even then, it is not necessarily clear which of the mix owners is also the owner of the pseudonym.

3.6 Summary

In this chapter, we presented the architecture of the UniTEC distributed reputation system. We highlighted the two major usage scenarios of requesting recommendations and publishing recommendations, and derived in the following the requirements of the participants requester and recommender towards the reputation system.

These requirements served as substantial input for the modeling of trust and knowledge and the system model. Both, trust in a certain participant and knowledge of a certain participant in UniTEC are modeled in a fine-granular manner in different context areas. Whereas trust, however, is only updated based on user feedback, knowledge update occurs automatically upon receipt of advertisement statements from recommenders. The system model of UniTEC consists of a fully distributed peer-to-peer model of UniTEC agents running in the context of their users on nodes with communication capabilities. Each user may use one or more virtual identities when interacting with the system through the agent.

The typical interactions of users with the UniTEC system consist of publishing recommendations and advertising the knowledge to interested parties accordingly, of disseminating own and foreign requests to parties in the neighborhood of the request's

[11]This argument refers to extended destination routing, an approach described in depth in Chapter 5.

context area, of collecting the received responses and evaluating the user's trust in the recommenders and presenting the recommendations to the user, and finally of receiving user feedback on the quality of the recommendations and evaluating this feedback to update trust.

We presented the architecture of the UniTEC agent with its five components and their functionality. The data management component (DMC) provides secure storage functionality for the local agent and remote agents via the DHT mechanisms. The identity management component (IMC) handles virtual identities of the agent's owner but also of other users obtained via knowledge advertisements or received recommendations. The anonymous peer-to-peer communication component (ACC) protects the strength of the virtual identities by using an approach based on Chaum Mixes for communication between pseudonyms. The peer-to-peer overlay component (POC) builds on the ACC to offer overlays containing a particularly well-suited set of recipients for requests for each context area. Finally, the trust management component (TMC) is responsible for computing transitive trust but also updating trust and knowledge according to user feedback respectively received knowledge advertisement messages.

When evaluating the architecture against the earlier posed requirements, we saw that the majority of requester's and recommender's requirements are fulfilled already. Two issues have not yet been adequately addressed: The first is the area of protecting the reputation data and the UniTEC agent itself against tampering. The second is the control over TDIs which relies yet much on the goodwill and conformance of the agents themselves. These issues are what we are going to focus on in the following chapter.

Chapter 4

Trust Foundation – Trusted Platforms

The features of the UniTEC agent described in the previous chapter support users in judging the quality of TDIs and their recommenders. However, these features are based one some assumptions, most notably the assumption, that the code of a user's UniTEC agent has not been tampered with. In this chapter, we investigate how to provide a secure foundation for our reputation system by placing the agent on a trusted computing platform. We published the majority of this chapter as a full paper [KP03] at EC-Web 2003.

We start this chapter by motivating our approach and then cover the background information regarding trusted computing with its building blocks and key functionalities. In the following we present the approach itself and how UniTEC agents can make use of trusted platform technology. We conclude this chapter with an discussion of our proposed concept and summarize our findings.

4.1 Motivation

The interested reader may ask why any additional support over the already provided features of the UniTEC agent might be necessary, since the trust mechanisms ensure already that recommendations are received from trustworthy recommenders. However, we saw during the evaluation in Section 3.5 that the UniTEC agent has no specific protection against attacks outside the regular UniTEC protocols, attacks therefore on the agent itself.

Were the platform that the agent resides upon to be compromised, may it be due to any sort of malicious code or hackers both on the operating system or Java runtime level, then the *trust mechanisms* of the agent *could be circumvented*. Even worse, the compromised agent could not only be used to present wrong content to its owner. Additionally, the agent could *damage its owner's reputation* by offering false recommendations and/or by disturbing the request forwarding. Finally, the agent could try to *damage the reputation of other participants* by forging the trust statements inserted in the trust chain when forwarding recommendation requests.

Further issues arise with the *access control mechanisms* that handle access to TDIs. We recall: access control mechanisms at the DMC allow to explicitly grant access to the TDIs to certain pseudonyms respectively in case of the exclude list to deny this access to certain pseudonyms. The problem here is, that his behavior relies on the agent to adhere to it, which a corrupted agent would not necessarily do any longer.

Finally, the corrupted agent could try to disturb the basic UniTEC mechanisms, like the *DHT data storage* or the *Mix service*. We should now be very motivated to ensure, that a UniTEC agent cannot be corrupted that easily.

4.2 Trusted Computing

We present a brief introduction to the concept of trusted computing, starting with background information regarding its origins. We then move on to the major building blocks and finally point out the selected key functionalities that our approach uses.

4.2.1 Background

The idea of trusted computing has been pushed by the *Trusted Computing Platform Alliance* (TCPA), a consortium founded in 1999 by the companies Hewlett-Packard[1], Compaq, IBM, Intel, and Microsoft. The goal of the TCPA was to develop industry standards for enhancing the security of a computing environment by among other things incorporating "roots of trust" into computer platforms. TCPA raised much interest in the community, such that the number of members grew to around 200 in 2003.

[1]Siani Pearson, co-author of our paper [KP03], works in the Trusted Computing Platforms Group of the Trusted E-Services laboratory of HP research labs in Bristol, which took part in the TCPA founding and which contributed some of the original ideas on which TCPA technology is based.

However, due to the fact that each member had the power of veto, the consortium's capability of acting became limited.

This led to the foundation of the successor of TCPA in April 2003. The *Trusted Computing Group*[2] (TCG) is a non-profit organization that took over the specifications created by TCPA. The TCG consists of members of Promoter, Contributor and Adopter level and is governed by a board of directors (the Promoter members), which currently consists of the companies AMD, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft, and Sun Microsystems.

Note: The work of the TCG is a matter of controversial discussions with almost religious intensity. Consumers are afraid that they might loose control over their own computers if trusted computing takes on momentum (And the prediction of IDC[3] analysts, that by 2010, essentially all notebooks and the majority of desktops will include a TPM chip, certainly seems to hint for that momentum). However, even renowned researchers such as Ross Anderson[4] or Bruce Schneier[5] express their doubts regarding where widespread TC technology will lead. Finally, the Free Software Foundation[6] even speaks of “treacherous computing” when talking about trusted computing. We do not want to comment on or take part in this discussion, but focus on the beneficial use that we can make of TC technology while acknowledging that malicious or at least non-beneficial usage might occur.

4.2.2 Building Blocks

A *Trusted Platform* (TP) - sometimes also called a *Trusted Computing Platform* - provides most of the basic features of a secure computer, but does so using the smallest possible changes to standard platform architectures. The three major building blocks for trusted computing are the *Trusted Platform Module* (TPM) as one of the core TCG specifications, a trusted processor architecture making use of the basic TPM features, and finally a trusted operating system supporting the underlying architecture's features.

[2] https://www.trustedcomputinggroup.org
[3] http://www.idc.com/
[4] http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html
[5] http://www.schneier.com/crypto-gram-0208.html
[6] http://www.fsf.org/campaigns/drm.html

Trusted Platform Core

The *Trusted Platform Module* (TPM) is a security hardware (roughly equivalent to a smart card chip). As described in [Tru01], it is physical to prevent forgery, tamper-resistant to prevent counterfeiting, and has cryptographic functions to provide authenticity, integrity, confidentiality, guard against replay attacks, make digital signatures, and use digital certificates as required (further explanation of these terms is given in [Sch96]). The TPM itself is a passive element whose functionality can be used by other components like the processor architecture or a trusted operating system. The specification requests that the user must be able to deactivate the TPM via a BIOS call.

The core of a trusted platform contains three roots of trust, as pointed out in [Tru04]: a *Root of Trust for Measurement* (RTM), a *Root of Trust for Storage* (RTS) and finally a *Root of Trust for Reporting* (RTR). The RTM is capable of making integrity measurements of the platform's hardware and software components and stores them in the TPM. The RTM functionality is used by the *Core Root of Trust for Measurement* (CRTM), a BIOS extension that is the first step of a secure boot process as described in the following. The RTS is responsible for securely storing symmetric and asymmetric keys, passwords, and opaque data entrusted to the TPM. The RTR provides support for reliably reporting information stored in the RTS.

Trusted Processor Architecture

The features provided by the TPM are essential but not sufficient for a trusted platform as such. The processor vendors Intel and AMD have therefore decided to add a set of additional hardware components to processors and chipsets that offer additional security support. Due to the fact that currently only little information can be found on AMD's approach called *Presidio*, we focus here on a brief description of the capabilities of the Intel solution.

The Intel approach called *Trusted Execution Technology* (TET), formerly *LaGrande*, has been announced for the second half of 2007. According to [Int03, Int06], changes are required for the microprocessor, chipset, I/O subsystems, and other platform components. TET requires the availability of a TPM and provides support for:

Protected Execution. Applications run in an isolated execution environment with dedicated resources, which cannot be accessed in an unauthorized manner by other applications.

Protected Input. The communication between input devices and applications in an protected execution environment is encrypted so that only authorized applications with the corresponding key have access to the input data.

Protected Graphics. The display information sent by applications in the protected execution environment to the graphics frame buffer is protected.

Sealed Storage, (Remote) Attestation, Protected Launch. These features are provided directly by the included TPM.

Trusted Operating Systems

The capabilities provided by the TPM and the trusted processor architecture need to be supported by the operating system. Although experiences are available regarding (provable) security of small-sized and highly specified operating systems, e.g. smartcard operating systems, the challenges for building secure operating systems for PCs are orders of magnitude higher. This might be the reason why no OS vendor has yet offered an operating system that fully supports the trusted computing capabilities and fulfills its requirements.

The *Next Generation Secure Computing Base* (NGSCB), formerly called Palladium, is Microsoft's initiative in the trusted computing space. Although NGSCB is not included in Microsoft's new operating system Vista, Vista uses selected features of a TPM via the so-called *BitLocker Drive Encryption*. According to [Mic07], BitLocker combines volume encryption with integrity checks of components at system startup. The key necessary to decrypt the volume upon system startup is released by the TPM only if the startup integrity checks passed successfully. This in turn is only the case if the components have not been altered or compromised and the volume is still in the computer that it was in when the measures were stored.

In addition to the OS adoptions implemented by Microsoft, work has been done for other operating systems as well to make use of trusted hardware. The idea of a *Trusted Linux* has raised much attention during the last years. As stated by Dolle and Wegener in [DW06], IBM laid the foundation in 2003 by publishing packages for the Linux kernel supporting TPMs. Based on this, many individual contributions for trusted Linux have been made, e.g. securing the Linux boot process has been addressed by members of the research group for applied data security at the University of Bochum in the context of the Trusted Grub[7] project.

The individual trusted computing efforts of 23 research institutions from industry and academia were bundled in *Open Trusted Computing* [KLR+06], an IST research project funded by the European Commission during the 6th framework program. The goal of this ongoing project is to build an architecture for a secure computing system based on open source software. According to the website[8], their work is based on TC hardware used by low level operating system layers with the addition of isolation properties. A specific focus is put on the Linux operating system.

4.2.3 Key Functionalities

The following key functionalities of a TP are necessary for our approach:

Protected storage. This functionality allows protection against theft and misuse of secrets held on the platform. Such secrets are rendered unintelligible unless the correct access information is presented and the correct programs are running and unchanged.

Integrity checking. This functionality provides a mechanism for a platform to show that it is executing the expected software in the expected state. As pointed out before, the CRTM is the first step of a secure boot process. Starting from there, the integrity of each of the following steps (CRTM, rest of the BIOS, boot routine, OS-loader, OS-kernel, device drivers, up to application programs) can be measured and stored securely in the TPM. This integrity of a TP can be checked by both local users and remote entities. This mechanism is used to provide the information needed to deduce the level of trust in the platform. The trust decision itself can only be made by the entity that desires to use the platform, and will change according to the intended use of the platform, even if the platform remains unchanged. The entity needs to rely on statements by trusted individuals or organizations about the proper behavior of a platform.

TCPA/TCG pseudonymous identities. These identities enable a mechanism for the platform to prove that it is a TP while maintaining anonymity. Proof that a platform is a genuine TP is provided by cryptographic *attestation identities*. Each

[7] http://www.prosecco.rub.de/trusted_grub.html
[8] http://www.opentc.net

identity is created on the individual TP, with attestation from a PKI Certification Authority (CA). Key features (further discussion in [Pea02]) are the following:

- The TPM has control over multiple pseudonymous attestation identities; the platform owner may choose different CAs to certify each TPM identity in order to prevent correlation.
- A TPM attestation identity does not contain any owner/user related information: It is a platform identity to attest to platform properties.
- No unique TPM "identity" is ever divulged to arbitrary third parties or used to digitally sign data – in order to give privacy protection, a TPM only uses attestation identities to prove to a third party that it is a genuine (TCPA/TCG-conformant) TPM.

4.3 UniTEC Agents on Top of Trusted Platforms

In this section, we show how to protect our software agents against misuse and fraud. We highlight the differences compared to Chapter 3 with regards to recommender and requester, point out the changes to the trusted data items and their storage in the TDM and finally cover the changes in the system interactions for requesting and publishing recommendations.

4.3.1 Recommender and Requester Role

As presented in the previous Chapter, the system model of our reputation system consists of *trusted agents* running in a specific *entity's* context on a particular computing platform. The trusted agents have connectivity to other agents on other entities' platforms in a peer-to-peer manner. The system is greatly strengthened if these platforms are TPs, as argued below. The entities can act in the roles of *recommender* or *requester* and may use pseudonymous attestation identities created via trusted platform mechanisms instead of pseudonyms created directly by the UniTEC agent.

Recommender: Upon having made own experiences, the recommender creates a recommendation (Subsection 3.2.4), publishes it and announces his expertise to interested parties (see 3.3.1). If the recommender's platform were compromised, wrongful recommendations could be created or existing recommendations could be altered or released inappropriately in the recommender's name. To counter this, a recommendation should only be sent out if the recommender platform's software environment is in the expected state (e.g. has not been hacked or compromised); this is possible because TP technology provides *protected storage functionality* for sealing data to a platform and software environment.

Requester: When uncertain whether to buy a product or to use a service, a user formulates a query with his trusted agent which queries a set of reliable sources and presents the received recommendations back to the user (see 3.3.2 and 3.3.3). Feedback (see 3.3.4) is given to the agent about what recommender gave a fitting recommendation and which one should not be queried on further occasions. If the requester's agent is not reliable, then the feedback given to the system by the requester about the information received might have been tampered with, and this can completely change the trust decisions based on those recommendations. It would even be possible to change the reputation information about already known recommenders or add strong trust in new (malicious) recommenders, with potentially disastrous results.

4.3.2 Recommendations or Trusted Data Items (TDI)

The structure and contents of recommendations or trusted data items is described in Subsection 3.2.4.

As mentioned before, we do add the recommender's identity to the recommendation, however not the real world identity but instead the *pseudonymous attestation identity* that the recommender is using for recommendations of the category in question. In order to prove the authenticity of the recommendation, the *digital signature* is added under the appropriate pseudonym.

The recommender's privacy is protected via protection of the stored recommendation using encryption and hardware-based storage of the decryption key(s) using the TP's protected storage functionality. *Authorization data* is needed in order to gain access to data stored via the TPM, and this cannot be overridden even by the platform owner or administrator, so the recommendation will only be accessible with the say-so of the recommender (or, more practically, the agent acting on behalf of the recommender). This approach, as opposed of the pure ACL based approach described in Subsection 3.4.1, provides increased protection of sensitive TDI content.

4.3.3 Interactions within this System

Requesting Recommendations

An entity seeking advice about a recommendation target uses the agent on its platform to formulate a *query for recommendations* about that target; this agent returns recommendations that are *locally* available and accessible and on his or her behalf requests recommendations from *other entities* with expertise in the area in question (each acting under pseudonymous identities) as discussed in Chapter 3. For each category in the trust model, this group of experts (or *neighborhood*) is being identified over time by adapting the trust values according to successful or not so successful previous encounters. Upon receipt of the query the receiving entity checks whether suitable recommendations are available and decides whether or not to forward the query further to its own group of experts for the category in question.

The received recommendations are weighted according to the *transitive trust* in their recommenders. This result is *presented* to the requester, who then decides whether or not to strike the deal/use the service. In case of a decision for the deal or service, own experiences are made. Then, the user decides whether it was a good or bad deal and delivers *feedback* to the system regarding which recommenders were giving out a fitting and which ones a non-fitting recommendation. Their *reputation is updated* accordingly.

If the agent is placed on a TP, the TPM can protect this trusted mechanism. Each agent may be *integrity checked* by the user of the platform or a remote party to ensure that the agent is operating as expected and has not been modified or substituted in an unauthorized manner. This process would involve a trusted third party (usually the vendor of the agent software) publishing or otherwise making available a signed version of the integrity measurements that should correspond to a genuine agent. Upon boot, the integrity of each agent can be measured as an extension to the platform boot integrity checking process [Tru01]; a challenger may then check the software state of the platform by comparing the measured integrity metrics with the certified correct metrics and, based on this information, decide whether to trust the agent. The agents themselves can be protected further by running within a protected environment such as a suitably isolated compartment.

Publishing Recommendations

An entity that is about to publish knowledge gained from interacting with a second entity creates a recommendation via an agent on its platform as described before. This recommendation is associated with the pseudonymous attestation identity that corresponds to the category of the recommendation in question.

The recommendations are protected via the TPM (exploiting protected storage mechanisms binding data to a TP and sealing it to its software environment) so that for example, unauthorized people could not see them. It is advantageous to allow recommendations from one recommender to be stored on multiple hosts, for instance for load balancing (for reputable recommenders) or availability reasons. This is achieved by storing *authorization information* with the recommendation, so that the owner of the platform on which the recommendation were stored would not necessarily be able to access that recommendation (in the sense of reading an unencrypted version of it), although he or she could delete it.

4.4 Discussion

The approach of placing a UniTEC agent on top of a trusted platform has all the advantages of the solution evaluated in Section 3.5. Additionally, the security requirements posed in Subsection 3.1.3 are addressed as follows:

S1: Protection against false recommendations (R2,R12). The recommender's platform can be integrity checked and trusted identities can be used to link recommendations. Nobody can make a recommendation in another person's name since the recommendations are protected by the digital signature of their recommender. The trust mechanisms ensure that wrongful recommenders are detected and prevent them from being queried in future transactions.

S2: Reputation protection (R2,R12,R13). If the recommender's platform were to be compromised, bogus recommendations could be created or existing recommendations could be altered or released inappropriately in the recommender's name. To counter this, it is possible for a recommendation to only be sent out or forwarded on if the recommender platform's software environment is in an unaltered state (e.g. has not been hacked); this is possible because TPs provides functionality for sealing data to a platform and software environment.

S3, S4: Recommendation protection (R13). The TPM protects against unauthorized access using TCPA protected storage mechanisms. Furthermore, recommenders are protected against malicious requesters through integrity checking of the requester's platform (if this platform is a TP), possibly coupled with other policy-level checks on the corresponding enquirer, before the recommender's platform releases recommendation information.

S5: Participant's privacy ensured (R10,R14). All parties may engage in the system without having to say who they really are via the use of trustworthy pseudonyms. This requirement, however, is only partially fulfilled, since multiple pseudonyms of a single user can be still linked together (possibly also to the user's identity) via the GUID used for communication. Our approach presented in Chapter 5 addresses this requirement fully.

S6: Redress for unreliable recommendations (R9). Potentially, if entity A receives a recommendation from entity B that proves to be false and results in A making a financial loss, could that entity go to entity B's privacy-CA to find out their real identity? The answer will depend upon the circumstances: Whether your privacy-CA reveals real identity in such a situation will depend upon the policy of that CA as well as legal reasons, such as whether you are suspected of breaking the law.

4.5 Summary

Trusted platform technology and in particular the work of the trusted computing group have been the center of many debates which often focus around the question of whether or not a user is still in control of the own computer. We do not want to participate in this religious discussion.

We have presented in this chapter, how to place a reputation system agent on top of a trusted platform, and shown, which of the platform's features can be applied by the agent. From the evaluation, we see that several of the limitations that were still open in Chapter 3 have been successfully addressed by placing the agent on this trusted foundation.

Due to the fact, that the reputation agent holds highly sensitive data, it seems realistic that the added protection of trusted hardware is worth overcoming the initial hesitation of activating a trusted platform module in the BIOS. Whether the public acceptance of trusted computing will increase, only time will tell.

Chapter 5

Trusted Entities – Privacy Considerations

In this chapter we introduce our approach how to represent the trusted entities in a distributed reputation system while protecting the privacy of all involved participants. The majority of this content has been published in two full papers, the first one at ACM SAC 2005 [KTR05a], the second as journal paper in a special issue of the International Journal for Infonomics [KTR05b].

In the following, we motivate our approach by presenting the problem statement and derived requirements from a distributed reputation system perspective towards an identity management infrastructure. We then discuss the related work in this area and their applicability to our case. After covering anonymization techniques as one of our main building blocks, we present our approach in depth. A detailed evaluation follows and a summary concludes this chapter.

5.1 Motivation

With the advent of the Internet and the ever increasing demand for world-wide communication, more and more information with uncertain quality is *available to people* and at the same time more and more *data about people* is gathered and stored electronically for a variety of reasons.

The first problem is addressed by research in the trust management and reputation system area, which strives to help users estimate the quality of products, services etc. and assess the subjective trustworthiness of other users, e.g. as reviewers/recommenders for the aforementioned recommendation targets. For the task of trust assessment, reputation systems collect, process and sometimes distribute sensitive user data. This user profiling as a basic building block of each reputation system is an example of the latter problem and holds massive *privacy risks*. In our view, the main danger lies in matching these *profiles* to *real user identities*.

One well-known approach for protecting the privacy of the users lies in employing anonymous communication techniques to prevent profiling. However, this is no viable option for reputation systems as such, since in order to build trust in an entity, a way of representing that entity and attaching information to it is necessary. The alternative of using *digital pseudonyms* instead of real identities has raised much interest lately. Some persons would argue that trust in a pseudonym cannot be built. However, during the course of this chapter we will point out our arguments why we think that we can very well trust pseudonyms, and actually do so already in many cases today.

So assume that trust in pseudonyms is possible and the concept of pseudonyms can be applied to the reputation system area. Are we there yet and all problems are solved adequately? We think not. The aforementioned contributions are essential but not yet sufficient if the employed pseudonyms can be linked to a globally unique identifier like the IP address. The users' *privacy* has to be *protected at various layers*, ranging from the application down to the communication medium.

Regarding the *communication aspects* in a *client-server scenario*, users can employ web anonymizers like the Java Anonymous Proxy (JAP) project[1] of the Technical University Dresden to protect their privacy. JAP and similar approaches build on the concept of Chaum mixes (discussed in Section 5.2) that hide the communication between two entities in the communication of large groups of senders and recipients. However, the mix approach is *not directly suitable for peer-to-peer* (P2P) type communication since clients need to know the IP address of the servers in order to communicate. Mixes therefore mainly protect clients but not servers. In spite of the increasing popularity of peer-to-peer based applications, there is still a need for anonymization techniques efficiently supporting this communication paradigm. Our approach of extended destination routing meets this need and *provides untraceable P2P communication between pseudonyms* with strong sender and recipient anonymity. This serves as a base for the higher-level communication of the application built on top, for instance the UniTEC recommendation queries, trust assessment queries and the corresponding replies.

[1] `http://anon.inf.tu-dresden.de/index_en.html`

5.1.1 Problem Statement

We require an approach for creating and using digital pseudonyms as representations for the trusted entities in our system. This approach needs to protect the privacy of all involved participants.

5.1.2 Derived Requirements

From the before mentioned problem statement and the UniTEC background and requirements identified in Chapter 3, we can derive the following requirements that an identity management solution suitable for our distributed reputation system has to fulfill:

- **Fault tolerance:** The failure or takeover of any node may not lead to the failure of the whole system.
- **Peer-to-peer:** The system must not depend on centralized concepts or entities in order to be applicable in a peer-to-peer context.
- **Connectivity:** The system must enable pseudonyms communicating with each other.
- **Ensure strength of pseudonyms:** The communication layer must not enable an attacker to link the pseudonym to a real identity through the use of a globally unique identifier (e.g. the IP address). Furthermore, the concepts must not allow an attacker to link different pseudonyms of an entity to each other.
- **Unlinkability of sender and recipient:** Even if it may be possible to discern that a sender is communicating and a recipient is receiving information, it must not be possible to notice that they are communicating with *each other*.
- **Sender and recipient anonymity:** The initiator respectively the recipient of the communication should remain hidden from other UniTEC nodes.

5.2 Related Work

In the following, we investigate whether approaches from the related work can address our requirements. We first provide an overview of traditional identity management solutions. Then, we introduce the major existing approaches for anonymization techniques. Finally, we examine the work that has been done in our research area of reputation systems with respect to identity provisioning under privacy protection constraints.

5.2.1 Traditional Identity Management Solutions

Owning various digital identities in many different systems has become a reality for the human being of the 21st century. In the private realm for instance, email providers, online-shops, auction sites etc. all require user authentication, with generally different account data. There are users who address the issue of having to remember many different account names and passwords by placing neat yellow password post-its under their keyboards, others store all passwords in a file on their local PC. While this situation is quite an inconvenience for private individuals, the situation for business users is significantly worse. This is not to say that business users necessarily need access to *more* systems. Instead, we assume that business users in general have access to *sensitive* or even *critical* IT systems, where the potential for loss (may it be financial loss, loss of reputation, etc.) for a company is higher were the account data compromised.

Companies are well aware of the security risks involved in users administering their own identities manually, but also of the management burden and cost of keeping user account data up-to-date. Identity management solutions have been developed for managing virtual identities which address the aforementioned issues for both, private and business realms. Two prominent examples are the logically centralized solution provided by Microsoft Passport and the decentralized approach of the Liberty Alliance. Two further solutions have recently received high attention in the security community, namely Microsoft InfoCards as the successor of Passport and the Security Assertion Markup Language (SAML) 2.0, whereas findings of Liberty and several other standards have been fed into.

The Beginnings: Microsoft Passport and the Liberty Alliance

Passport is a centralized identity management approach required for several of the products shipped with Microsoft Windows, e.g. the MSN Messenger, but also for the Microsoft Email service Hotmail and other third party services.

Websites participating in the Passport scheme outsource their user authentication to the Passport server. Whereas previously users had to remember many different usernames and passwords for the various websites, this task is now simplified by using a single account at the Passport server. Whenever a user wants to access a Passport-enabled application, this application expects to receive a certain cookie named *MSPAuth*. If this cookie does not exist yet, the user is redirected to the Passport site, which again checks for a MSPAuth cookie. If it does not exist in the Passport domain either, the Passport server queries the user's credentials. If the credentials match the information stored on the server, an MSPAuth cookie is set in the Passport domain which serves for future single sign-on purposes. The user is then redirected to the target application URL and a cookie for this target application domain is set as a URL parameter. Depending on the validity of these cookies, the user can access a second application without re-authenticating himself.

Whereas the Microsoft approach relies on the Passport server as centralized source of trust, the *Liberty Alliance* supports a decentralized paradigm. Parties may participate in the scheme in an *identity provider* respectively *service provider* role. Identity providers validate user identities in a variety of ways, e.g. using traditional username-password-pairs or client certificate authentication. They may issue generally short-lived *assertions* regarding the authenticity of users. These assertions are consumed by service providers which rely completely on identity providers for providing user authentication. The details of the assertion handling are outside the scope of this discussion. More information can be found at the Liberty Alliance website[2].

Today's Distributed Approaches: InfoCards and SAML2.0

The slow adoption of Passport as opposed to the massive interest in the Liberty approach leads to the assumption that users were not that enthusiastic about entrusting all their digital identities solely to a single party. Therefore, the *InfoCards* system was developed as a successor of Passport with distributed features in mind and relying on the web service standards WS-SecurityPolicy, WS-MetadataExchange and WS-Trust. Here, the InfoCards serve as a visual representation of all the user's digital identities, which can be chosen by the user in the so-called identity selector for each transaction that requires user authentication. The identity selector can acquire a security token from the Security Token Service (STS) of the identity provider that issued the InfoCard and present this token to the STS of the target service that the user wants to consume. If the trust relationship between service and identity provider is configured properly, the user will be granted access.

[2] `http://www.projectliberty.org/`

The Security Assertion Markup Language (SAML) in its current version 2.0 was defined by integrating results from five earlier standards: SAML 1.0, SAML 1.1, Liberty Alliance ID-FF 1.1 and 1.2, and lastly Shibboleth. In its current version 2.0, it defines among other things these four major building blocks: the token format, protocols, bindings and profiles. The roles of identity provider issuing assertions and service provider consuming assertions are adopted from the Liberty approach. The assertion or token format is used to specify authentication, authorization and general attribute statements regarding a certain principal. The protocols define how to retrieve existing and how to request new assertions from an identity provider but also how to change a principal's name identifier, map it to another SAML name or terminate the mapping. These defined protocols can be bound to different underlying transport protocols, e.g. SOAP, as described in the binding specifications. The profiles are used to combine information about assertions, protocols, and bindings to address a particular use-case, e.g. browser-based single sign-on or single logout. One of the strengths of SAML lies in the ability to support identity federation and in basic privacy support via the use of pseudonymous identifiers in the assertions.

Discussion

While the Microsoft Passport approach provides single sign-on capabilities to its participants, it does not meet our requirements for an identity management solution. Most prominently, it is a centralized approach and therefore does not fulfill the peer-to-peer requirement. Furthermore, there is no privacy support. Whereas the other presented approaches take on a distributed paradigm, the requirement of P2P support is still not fulfilled, the notion of trust does not fit to the one of our reputation system, and finally there is a massive lack in support of privacy features.

The three approaches have in common that they require a trust relationship to be configured manually between the service provider and the identity provider, respectively the STS in the corresponding role. In a fully decentralised system like UniTEC, each agent would be identity provider and service provider at the same time. The manual configuration of the trust relationships would be un-manageable. Furthermore, the trust establishment required by Liberty, SAML2.0 and InfoCards is binary in the sense that either trust exists between a service provider and an identity provider, or it does not. The fine-granular trust relationships, which we introduce in Chapter 6, conflict with this binary trust representation.

Although Liberty and SAML 2.0 provide a limited privacy support by using opaque user identifiers within the assertions, neither is the strength of these pseudonyms protected nor is the unlinkability of client and service provider guaranteed. An outside attacker cannot access the pseudonym since the communication of an assertion to a service provider is generally encrypted (to protect the assertion). However, the attacker can notice that communication between client and service provider took place and therefore link client to service provider. Taking into account the fact that the IP addresses of the parties are known to the attacker as well, it is not unlikely that the link can be extended to real world entities as well. If the attacker is the service provider himself, additional threats exist due to the fact that obviously the content of the communication is known as well as the used pseudonym and the client's IP address. If the same client uses a different pseudonym for the next communication with this malicious service provider, the second pseudonym can be linked to the first one if the IP address has not changed. Thus, the strength of the pseudonyms is endangered. Obviously, these dangers of linking the identities of users to their real-world identities also apply to the WS-Trust-based communication of clients with their STS in the InfoCard case.

We conclude that the traditional identity management solutions do not support the requirements we pose towards the identity management component of our distributed reputation system. Due to the fact that our main criticisms lay in the area of privacy protection, we investigate the existing approaches for enabling anonymous communication in the following.

5.2.2 Anonymous Communication Techniques

In this subsection, we introduce the terminology that is then used to describe the properties of existing anonymous communication techniques.

Terms and Definitions

The following terminology draws on the work of Pfitzmann and Waidner in [PW87].

There are three different *types of anonymous communication* properties, namely sender anonymity (the initiator of the communication remains hidden), recipient anonymity (the target of the communication remains hidden, e.g. using broadcast) and the unlinkability of sender and recipient (it might be visible that sender and recipient are communicating, but it is unclear that they are communicating with each other).

A second term, the *degree of anonymity*, was introduced by Reiter and Rubin. This degree ranges from *absolute privacy*, where the attacker does not notice any communication, to *provably exposed*, where the attacker can identify the sender and even prove to others that this sender indeed sent the message. There are several steps in between these two extremes, e.g. *beyond suspicion*, where the attacker notices communication, however the correct sender is no more likely to be the originator than any other sender.

In addition to different types and degrees of anonymity we have to consider different *attacker types*: the *global attacker* can listen to each and every communication whereas the *local attacker* might only collaborate with a number of senders, recipients and intermediaries.

Existing Approaches

We present a selection of the related work in the area of anonymous communication, namely the basic concepts of onion routing, crowds and DC networks and the related projects Tarzan and Freenet, which partly build on top of these basic concepts.

Chaum introduced in [Cha81] the idea of *mix networks* that describes how nodes called *mixes* can be used to enable anonymous email messaging. Later on, Reed, Syverson and Goldschlag have taken on this idea in the concept of *onion routing* described first in [GRS96] for allowing general anonymous communication. Mixes are nodes used by senders as intermediaries to pass on messages either to other mixes (thereby forming a so-called *mix cascade*) or to the intended recipient. A mix accumulates a certain number of messages, reorders them and passes them on. If the necessary number of messages is not received within a certain time the mix creates dummy messages to fill up the queue and sends the messages on. Instead of transmitting in plain text over a cascade of mixes $M_1, \ldots, M_x$ the sender wraps multiple layers of encryption around data d thereby forming an onion-like structure.

The approach of mixes provides *unlinkability* between sender and recipient even in the case of a global attacker. This attacker can observe groups of senders communicating with groups of recipients, but it cannot observe which individual from the sender group communicates with which individual from the recipient group. The sender anonymity property holds as long as there is at least a single non-corrupt mix in the cascade. On the downside, the sender has to know the address of the recipient in order to form the onion, therefore this approach does not provide recipient anonymity. Our solution presented in this chapter builds on mixes while ensuring the property of recipient anonymity.

Freedman and Morris proposed in [FM02] the Tarzan system that builds on mix networks and establishes a peer-to-peer anonymizing IP network overlay. It guarantees high resistance against traffic analysis through the use of layered encryption, multi-hop routing, cover traffic and a mix selection protocol. Tarzan provides a high level of sender and recipient anonymity; however, the sender still has to know the address of the recipient in order to communicate.

A theoretical concept named *DC Networks* was developed by Chaum [Cha88] and enhanced by Waidner and Pfitzmann in [WP89]. The basic idea behind the dining cryptographers (DC) lies in having a shared communication channel between all participants, whereas each participant shares secret keys with each of his or her neighbors. Each participant applies the XOR function to all known keys and communicates the result. If a participant wants to send a message M, he instead combines M with all keys with XOR. If all the communicated results are combined with XOR again, message M can be read without anybody (except the sender) knowing who has sent it. This approach has interesting anonymity properties by providing *sender and recipient anonymity* plus *unlinkability* of sender and recipient. However, implementations are still rare due to the massive communication overhead.

Reiter and Rubin introduced the *Crowds* system [RR99] that provides users with the possibility to hide their transactions with a specific web server within those transactions of all the other users in the crowd. This is achieved by placing a proxy called Jondo on the nodes of all participating users. The Jondo intercepts all requests from the client's webbrowser to a webserver and instead transmits these requests to another Jondo in the Crowd. Each Jondo either transmits requests to another Jondo or directly to the specified webserver. The Crowds concept does not provide unlinkability between sender and recipient in case of a global attacker. This is the case, because the global attacker can monitor the path a message takes from the originator through the set of Jondos to the target despite the message between the Jondos being encrypted. For non-global attackers Crowds provide sender anonymity, since they allow the user to deny having sent a request to the webserver.

A project that is related to the Crowds approach is the Freenet[3] project [CSWH00]. Freenet is a peer-to-peer network for distributed data storage that provides sender anonymity against collaborating nodes. Clarke et al. argue that the anonymity properties can again be strengthened by employing mixes for pre-routing messages. However,

[3] `http://freenet.sourceforge.net`

since the focus of Freenet lies in anonymous distributed data storage, it cannot be easily applied to P2P communication.

5.2.3 Privacy Protection in the Area of Reputation Systems

Friedman and Resnick [FR01] describe the effect of using pseudonyms in the context of trust and reputation systems and point out the social cost involved in allowing users to change their pseudonyms freely.

In [KP03], respectively Chapter 4, we propose to use TCPA attestation identities to create pseudonyms. Besides providing a way to reveal the real identity of a pseudonym holder in a legal dispute this also addresses the risk of Sybil attacks defined by Douceur in [Dou02].

Seigneur and Jensen argue in [SJ04, SGJ05] that the inherent conflict of privacy protection on the one hand and trust-establishment on the other can be solved by fine-grained negotiation mechanisms.

Daniele Quercia et al. recently proposed TATA[QHC06], an approach based on existing algorithms for blind (t,n) threshold signatures. It consists of two phases: The induction phase requires the cooperation of t among the n participants to generate a pseudonym and corresponding public and private key pair for a user. The authentication phase is required when two pseudonyms want to communicate to prove that each holds the respective private key.

None of these approaches discuss how the strength of the pseudonyms can be protected from attacks against their globally unique id used during communication.

5.3 Anonymous Communication between Pseudonyms

In this section, we introduce our approach that allows pseudonyms to communicate with each other without endangering their anonymity. First, we lay the base by highlighting several properties of our pseudonyms, most notably the pseudonyms' identifier. After discussing where to publish the pseudonyms' public keys, we introduce the Mix functionality incorporated in the UniTEC agent. This is followed by an in-depth presentation of our core solution called *extended destination routing*. The notation used for describing the structure of the protocol messages can be found in Appendix A.

5.3.1 Properties of Digital Pseudonyms

The digital pseudonyms are managed by the UniTEC agent's IMC as presented in Chapter 3. Upon creation of a pseudonym P, the IMC creates a public and private key pair (PuK_P, PrK_P) and an identifier (Id_P). The key pair is used among other things for ensuring confidentiality of the communication with P, whereas Id_P identifies P. We consider two options for identifying the pseudonyms: firstly the public key of the pseudonym and secondly the hash of this public key.

The public key could be used as an identifier since the likelihood of two pseudonyms having the same public key is infinitesimally small. However, the length of PuK_P (1024 Bit RSA) makes this solution infeasible as an address for communication. Therefore, our approach uses the second option, namely the *SHA-1 hash* (160 Bit) of the public key:

$$Id_P = Hash(PuK_P) \tag{5.1}$$

The fact that cryptographic hash-functions produce an equipartition of the output space ensures that the probability of two pseudonyms having the same identity is again minuscule. SHA-1 is a *secure hash-function*. Therefore finding a public key that produces a given hash is computationally infeasible. It is important to note that this connection between identity and public key relieves us from the necessity to employ digital certificates. Obviously, when a key gets compromised, the pseudonym cannot be used any longer. One issue still has to be solved: how can we make these keys available to other pseudonyms without compromising the anonymity.

5.3.2 Publishing Public Keys

Three different entities could publish the public keys: the agent of the key owner, a centralized or a distributed repository. Storing a public key on the key owner's agent however would enable inquirers to see the link between the IP address of the agent and the identity of the pseudonym, the very thing we need to keep hidden. A centralized solution well-known in the area of public key infrastructures is to employ a directory service. Yet this contradicts the requirement of no centralized components in the UniTEC system.

Therefore we chose the distributed approach, namely we built our solution on Chord (see [SMK+01]), a redundant representative of distributed hash tables (DHTs). Chord organizes its participating nodes in a logical ring. Each UniTEC agent participates in this Chord ring and stores part of the content of the whole database. Upon startup, the agent registers all of its managed pseudonyms in the DHT as we explain in more detail in Section 5.3.5.

5.3.3 Mixing

As base for anonymity UniTEC agents provide mix functionality in a similar way to David Chaum's mix network approach described in the related work section. When enabled, the mixing service creates a public and private key pair that is used for encrypting messages to be sent via this mix. It is the user's choice whether or not to enable the mix service at their agent, however, we argue later that not enabling this service results in weaker anonymity properties for the pseudonyms registered at that agent.

Mixes in traditional onion routing strip off the messages' layers of encryption and pass the messages on. Compared to that, UniTEC mixes perform different operations as described in the following section. All the mixes need to send *dummy traffic* in order to cope with message sparsity[4].

5.3.4 Extended Destination Routing (EDR)

We first cover a basic version of EDR which still allows two different kinds of attacks which we address in the following. The adaptations to basic EDR which prevent the aforementioned attacks form our suggested approach of full EDR.

Basic EDR

Conventional onion routing assumes the availability of the recipient's address information. In order to ensure recipient anonymity, this information cannot be made available in plain text. To solve this problem, pseudonyms store their *addresses* encrypted as *onions* in a lookup service, in our case the DHT Chord. The resulting onion is obtained by communication initiators and used as a *routing header* for routing the message to

[4]Without dummy traffic, it is possible that the messages are delayed by the mixes for a long time, until a sufficient number of other messages has arrived at the mix so that the actual re-ordering and send-process can be performed. A current limitation of the UniTEC prototype's ACC is that it does not yet send dummy traffic.

the intended recipient. As an example, a pseudonym R chooses a mix cascade containing three mixes M_1, M_2 and M_3. R encrypts its address A_R successively with the public keys of the chosen mixes PuK_{M_1}, PuK_{M_2} and PuK_{M_3}. The resulting routing header RH is:

$$\begin{aligned} RH = \{\, & A_{M_1}, \\ & Enc(\, PuK_{M_1}, \\ & \quad \{\, A_{M_2}, \\ & \quad\; Enc(\, PuK_{M_2}, \\ & \quad\quad \{\, A_{M_3}, \\ & \quad\quad\; Enc(\, PuK_{M_3}, \\ & \quad\quad\quad A_R\,)\}\,)\}\,)\} \end{aligned} \tag{5.2}$$

As the pseudonyms' routing headers and public keys are stored in the DHT, they are available to all potential senders. If a pseudonym S wants to send a data item D to pseudonym R, it encrypts D with R's public key and appends R's routing header after stripping off the address of the first mix A_{M_1}. The resulting message M is sent to M_1:

$$\begin{aligned} M = \{\, & Enc(\, PuK_{M_1}, \\ & \quad \{\, A_{M_2}, \\ & \quad\; Enc(\, PuK_{M_2}, \\ & \quad\quad \{\, A_{M_3}, \\ & \quad\quad\; Enc(\, PuK_{M_3}, \\ & \quad\quad\quad A_R\,)\}\,)\}\,), \\ & Enc(\, PuK_R,\, D\,)\} \end{aligned} \tag{5.3}$$

The receiving mix decrypts the remaining part of the routing header, strips off the address of the next mix A_{M_2} and uses this address to forward the data. This is recursively repeated until R receives $Enc(PuK_R, D)$.

The basic EDR described until now permits two attack possibilities, *message tracing* and the *own mix attack* which we both address in the following by the real EDR.

Message Tracing

The encrypted payload $Enc(PuK_R, D)$ remains the same during the communication, allowing message route tracing and thus endangering the unlinkability between S and

R. The basic defense against this form of attack is *hop-by-hop encryption*: At each node, before forwarding a message, the payload part of the message is encrypted with the public key of the next hop (e.g. $Enc(PuK_{M_1}, Enc(PuK_R, D))$ for the first mix). After receiving a message, its payload is decrypted and then again encrypted with the public key of the next hop (PuK_{M_2}), and finally the message is sent to this hop. A random-length padding is applied before the encryption in order to vary the payload's length and protect against statistical attacks.

Own Mix Attack

A malicious pseudonym R can manipulate its routing header by choosing itself as the only "mix" of the cascade:

$$\begin{aligned} RH = \{\, & A_R, \\ & Enc(\, PuK_R, \\ & \quad \{\, A_R, \\ & \quad\; Enc(\, PuK_R, \\ & \quad\quad A_R\,)\}\,)\} \end{aligned} \tag{5.4}$$

Since routing headers are encrypted, this manipulation cannot be detected. When the sender communicates directly with the first "mix" which is in fact R, its anonymity is compromised.

Full EDR

The solution that defends against this form of attack is *extended destination routing*. After looking up the routing header, the sender chooses part of the cascade itself. A simple way to accomplish this without major changes to the described approach is to extend the routing header with a cascade controlled by the sender. This construction ensures, that a recipient cannot gain access to the sender's real identity by manipulating the routing header.

For example, the manipulated header mentioned above could be extended by the sender with a cascade part containing two mixes M_1 and M_2. The extended routing header (ERH) looks as follows:

$$
\begin{aligned}
ERH = \{\, & A_{M_1}, \\
& Enc(\, PuK_{M_1}, \\
& \quad \{\, A_{M_2}, \\
& \quad\; Enc(\, PuK_{M_2}, \\
& \qquad \{\, A_R, \\
& \qquad\; Enc(\, PuK_R, \\
& \qquad\quad \{\, A_R, \\
& \qquad\quad\; Enc(\, PuK_R, \\
& \qquad\qquad A_R\,)\}\,)\}\,)\}
\end{aligned}
\tag{5.5}
$$

5.3.5 Storing Public Keys and Routing Headers

As introduced in the previous section we store public keys as well as routing headers in the distributed hash table Chord. However, if the pseudonym information was stored directly in the lookup service, the link between the sending agent's IP address and the pseudonym identity would become known to nodes in the lookup service. Therefore, as illustrated in Fig. 5.1, the updating pseudonym sends its information through a mix cascade to an *update proxy*. The update proxy can be any UniTEC agent with enabled update service. Again, the sender's anonymity is protected.

The information to be stored consists of a signed package containing several routing headers, a timestamp, the pseudonym ID and its public key. We include several routing headers to increase the pseudonym's availability because the mixes needed to reach the pseudonym may become unavailable at any time. The timestamp is used to avoid replay attacks, since the responsible node in the DHT may determine the most recent update package for a certain pseudonym. Without the timestamp, an attacker may send an outdated update package to an update proxy, when at the same time the mixes in the included routing headers have gone offline, thereby efficiently cutting of the pseudonym from the overlay.

After receiving this package, the update proxy checks its integrity and authenticity by verifying the signature and comparing the hash of the included public key with the pseudonym's ID. If these checks are successful, the update proxy determines the nodes responsible for storing information for this pseudonym in the DHT, which is based on the pseudonym ID. It then forwards the package to these nodes, which store it in memory. There is no need to store this data on persistent media, since the information

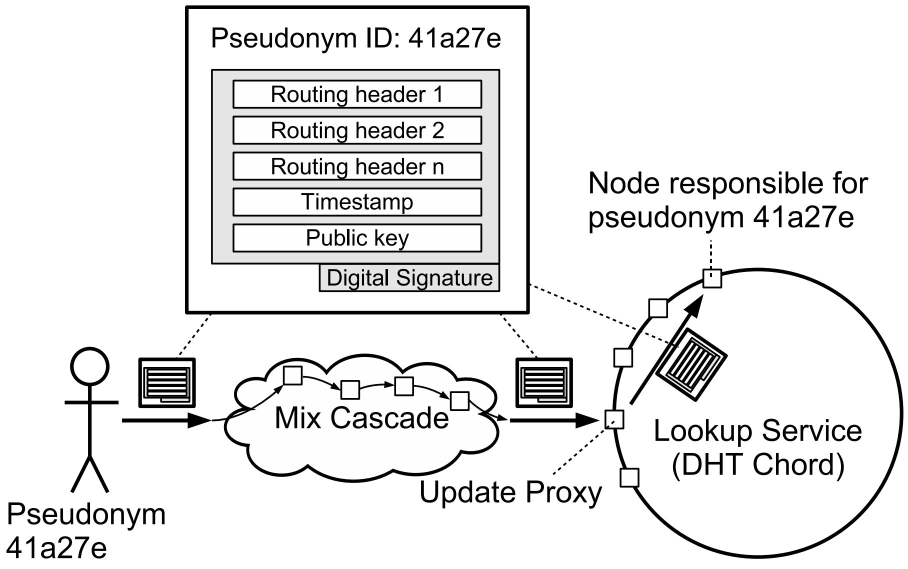


Figure 5.1: Routing update. A pseudonym with ID `41a27e` constructs the update package, containing several routing headers, a timestamp, the pseudonym's public key and its digital signature over the previously mentioned entries. The package is sent via a Mix cascade to the update proxy which forwards it for storage to the responsible node(s) in the DHT.

contained in the routing headers is based on the availability of the used mixes and is updated at regular intervals.

One might ask why the updating pseudonym does not send the data directly to the responsible nodes instead of going through an update proxy. At the time of this writing, none of the existing DHT implementations allow the determination of the responsible nodes while preserving anonymity for the querying entity. Furthermore, using an update proxy permits a separation between the DHT and the anonymity layer, allowing us to replace the DHT implementation if necessary.

The routing information updater service ensures that the routing information published in the DHT is up-to-date. Most notably, this includes taking into account the information from mix discovery and making sure that the mixes contained in the routing headers are alive.

5.3.6 Mix Discovery

A prerequisite for the presented approach is the availability of mix information, i.e. addresses and public keys, as they are used to create routing headers and to forward messages. For this purpose, every agent manages a list containing information about known available mixes.

Due to the use of Chord, the participating agents are organized in a logical ring topology. When activating the mix service at an agent, the local mix information is pushed to the first *n* successors in the ring, which store it in their list. Periodically, every agent starts a mix discovery, asking a random agent for known mixes, thus pulling the information contained in that agent's list. The pulled information is merged with the local list.

In regular intervals, the mix discoverer checks whether the mixes in the resulting list are alive and prunes entries containing non-responsive mixes. Secondly, the mix discoverer ensures that the list has a specified length and starts a mix discovery if the current length is not sufficient.

5.4 Evaluation

We investigate in this section how our approach of extended destination routing fulfills the posed requirements. First, we cover the security-related requirements, followed by a presentation of the performance results of our prototype. Finally we discuss why trusting pseudonyms is possible and, in fact, being done already.

5.4.1 Security Considerations

The overall approach described in this chapter provides connectivity between pseudonyms. Our peer-to-peer based solution can tolerate node failures or takeovers through the use of multiple routing headers while the data is securely encrypted. Therefore, the requirements regarding connectivity, fault-tolerance, and peer-to-peer are fulfilled.

We provide sender and recipient anonymity through the use of extended destination routing, especially through the creation of routing headers and their storage in the DHT. By applying the mix concept, we ensure unlinkability of sender and recipient as in the original onion routing as well as the protection of the employed pseudonyms.

On the downside, the usage of hop-by-hop encryption might motivate an attack on the mixes, since the payload is decrypted at each mix (but still encrypted with the recipient's public key). This leads to two relevant attack scenarios: *cooperation of corrupt mixes* and the *last mix attack*.

Cooperation of Corrupt Mixes

Corrupt mixes could exchange information about the forwarded payload, thereby being able to trace at least part of the message route. It is unlikely that, while using extended destination routing with a high number of available mixes, the resulting cascades consist entirely of corrupt mixes. Nevertheless, cooperating mixes can diminish the anonymity degree provided by a cascade.

This effect can be reduced (though not eliminated) by using a different approach for the cascade extension. Instead of just extending the routing header, the sender can use traditional onion routing for the sender-controlled part of the cascade. For example, using

$$\begin{aligned} RH = \{\, & A_{M_1}, \\ & Enc(\, PuK_{M_1}, \\ & \quad \{\, A_{M_2}, \\ & \quad\; Enc(\, PuK_{M_2}, \\ & \qquad A_R\,)\}\,)\} \end{aligned} \tag{5.6}$$

as published routing header, and

$$Enc(PuK_R, D) \tag{5.7}$$

as encrypted data to be sent to recipient R, the sender extends the recipient-controlled part of the cascade M_1, M_2 with two more mixes M_3, M_4 of its own, generating the following message M:

$$
\begin{aligned}
M = Enc(\ & PuK_{M_3}, \\
& \{\ A_{M_4}, \\
& \quad Enc(\ PuK_{M_4}, \\
& \quad\quad \{\ A_{M_1}, \\
& \quad\quad\quad Enc(\ PuK_{M_1}, \\
& \quad\quad\quad\quad \{\ Enc(\ PuK_{M_1}, \\
& \quad\quad\quad\quad\quad \{\ A_{M_2}, \\
& \quad\quad\quad\quad\quad\quad Enc(\ PuK_{M_2}, \\
& \quad\quad\quad\quad\quad\quad\quad A_R\ )\}\ ), \\
& \quad\quad\quad\quad Enc(PuK_R,\ D\ )\})\})\})
\end{aligned}
\tag{5.8}
$$

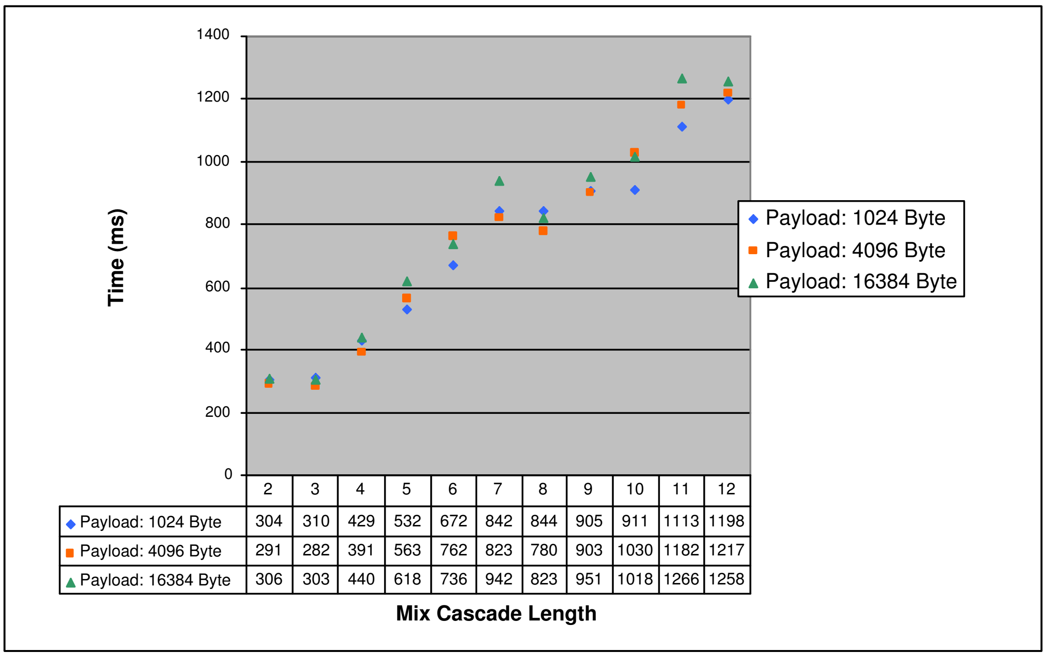
Obviously no mix in the sender cascade can see the payload, as it resides safely inside the onion. This still does not protect from attacks on mixes in the recipient's part where the sender has no control over the cascade. However, one can argue that the recipient is responsible for selecting mixes carefully. Dingledine et al. [DFHM01] describe an approach that uses reputation for selecting "good" mixes, which might be an option worth investigating in future works building on this dissertation, since UniTEC inherently provides reputation mechanisms.

Last Mix Attack

If the sender of a message happens to be the last mix in the cascade and it is aware of this by recognizing the payload and by analyzing the routing header – which is empty after stripping off the address of the last hop – then the recipient's address is exposed to the sender.

To avoid this, it is necessary to prevent mixes from identifying empty routing headers. This can be realized by including not only the address into the core of the recipient routing header, but also random size padding encrypted with R's public key. This prevents the last mix to notice that it actually sends to the recipient's agent since there might be hidden still more mixes of the recipient's part of the cascade in the padding, whose contents the last mix cannot access. This is the reason why we recommend to activate the mix service at each node, so that the last mix cannot discern from a non-activated recipient mix service that it communicates directly with the recipient.

Another alternative is for the recipient to embed the own mix service in the recipient cascade. This would enable the recipient in case of legal dispute to deny being the



	2	3	4	5	6	7	8	9	10	11	12
◆ Payload: 1024 Byte	304	310	429	532	672	842	844	905	911	1113	1198
■ Payload: 4096 Byte	291	282	391	563	762	823	780	903	1030	1182	1217
▲ Payload: 16384 Byte	306	303	440	618	736	942	823	951	1018	1266	1258

Figure 5.2: Message transfer time in case of a single sender and 20 Mixes, with a message payload size of 1024, 4096 and 16384 Byte, varying Mix cascade length and communication over Java object serialization over TCP sockets.

owner of the own pseudonym while being still able to receive all communication directed to that pseudonym. For the UniTEC prototype, we implemented only the former padding-based approach.

5.4.2 Performance Considerations

The UniTEC prototype was developed using Java 2 SDK Standard Edition Version 1.4.1 and the Bouncy Castle Crypto API[5] as JCE provider. Originally, as presented in [KTR05a], we used XML-RPC as communication protocol. In our first evaluations, the performance results were not completely satisfactory due to the high overhead of the XML-RPC communication. Therefore we did a re-implementation of the communication component and used Java object serialization directly on TCP sockets. The implementation was evaluated through multiple scenarios on our 64 node network emulation cluster NET. NET relies on standard Pentium IV 2,4 GHz blades connected via a 1000 MBit switch. More information on NET can be found in [HR02, Her05].

[5] `http://www.bouncycastle.org`

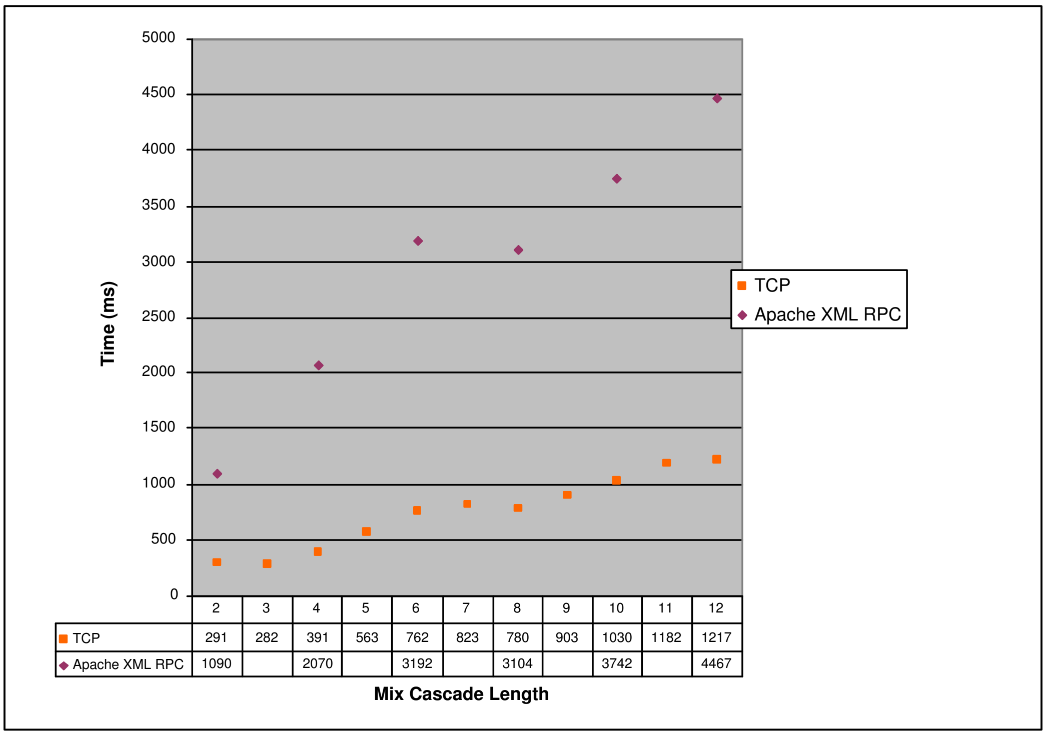
The results depicted in Fig. 5.2 represent the performance measurements when executing the UniTEC agent on 21 cluster nodes, 20 of which with enabled mix service in addition to one dedicated sender. The agent on the dedicated sender node sends a message from one of its pseudonyms to another one of its pseudonyms and records the *message transfer time* (MTT). We vary the message payload size and total mix cascade length while keeping the average mix queue delay (50 ms) and RSA key size (1024 Bits) fixed. This allows us to focus mainly on the cryptographic operations and the communication time as well as lookups in the DHT. The total mix cascade length is the sum of mixes in the sender and recipient cascades for message and acknowledgment. Each point in the figure represents the average MTT of 20 sent messages.

The MTT depends linearly on the length of the mix cascade. This is reasonable since for every additional hop, a fixed amount of time is necessary to provide the decryption and re-encryption of the message. Furthermore, the gradient depends on the message size due to the fact that the time for performing the aforementioned actions increases linearly with the message size. As an example, a 16 KB message takes on average about 1 s through a total cascade of length 10, which leads to an average time of 100 ms per hop.

Fig. 5.3 depicts the performance improvements gained through the use of Java object serialization directly over TCP sockets as opposed to the results presented in [KTR05a] which are based on Apache XML RPC. Clearly, the improvement is massive with an average transfer time in the TCP case of less than one third of the time necessary for the XML RPC case.

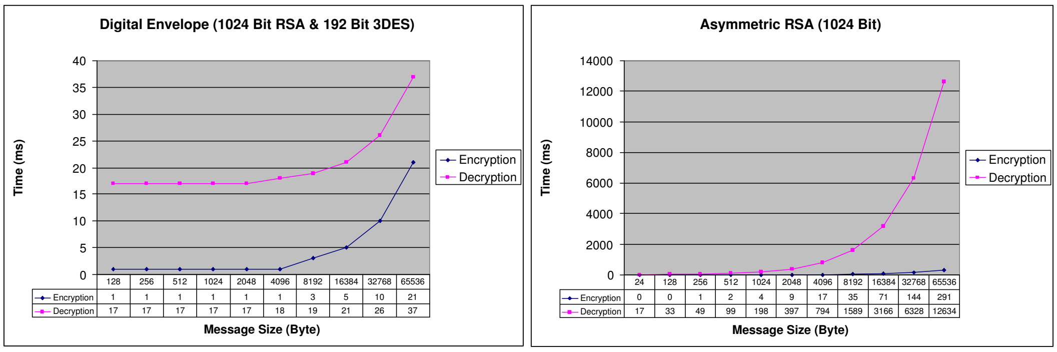
In order to better understand the part of the transfer time taken up for the crypto operations, we further investigated the crypto performance of the cluster nodes. Fig. 5.4 presents our measurements for encryption and decryption time of a message of the specified size. Each dot in the figure represents the average of 50 measurements. The left part of the figure depicts the results from a digital envelope approach, which we use in UniTEC and that consists of the data being encrypted by a symmetric 192 Bit 3DES key to which this 3DES key is appended, but encrypted with the 1024 Bit RSA key. The right part of the figure displays the results when using plain RSA.

The digital envelope results show an average difference between encryption and decryption time of 16ms, the reason of which is not immediately clear. However, when taking a look at the results of decrypting 24 Byte or 192 Bit with RSA, we see that the difference resulted from decrypting the 3DES key with RSA. The reason for this asymmetric time lies in the fact that in most RSA implementations, the public expo-



	2	3	4	5	6	7	8	9	10	11	12
TCP	291	282	391	563	762	823	780	903	1030	1182	1217
Apache XML RPC	1090		2070		3192		3104		3742		4467

Figure 5.3: Message transfer time in case of a single sender and 20 Mixes, with a message payload size of 4096 Byte, varying Mix cascade length, and communication over XML RPC respectively Java object serialization over TCP sockets.



	128	256	512	1024	2048	4096	8192	16384	32768	65536
Encryption	1	1	1	1	1	1	3	5	10	21
Decryption	17	17	17	17	17	18	19	21	26	37

	24	128	256	512	1024	2048	4096	8192	16384	32768	65536
Encryption	0	0	1	2	4	9	17	35	71	144	291
Decryption	17	33	49	99	198	397	794	1589	3166	6328	12634

Figure 5.4: Comparative analysis of the time for encrypting and decrypting a message of specified size, firstly as shown on the left hand side with a digital envelope approach, combining 1024 Bit RSA with 192 Bit 3DES, and secondly as depicted on the right hand side solely with 1024 Bit RSA.

nent e is *chosen* to be either 3 or 65537 ($2^{16}+1$), which allows a fast computation of x^e ($y = x^e \bmod n$). The private exponent d contains about as many Bits as the modulus n, which makes the computation of $x = y^d \bmod n$ slow in comparison.

The crypto operations performed for each message at every mix are the decryption of the routing header, the decryption of the message payload size and the re-encryption of the payload. For a 4096 Byte message with a cascade length of 10, the crypto operation takes less than 40 ms, which is also less than 40% of the whole message transfer time. When taking into account the average mix message queue delay of 50 ms, the pure communication time in our emulation is negligible. This, however, is not necessarily the case in an open environment where no Gigabit connection is available.

To summarize, the performance of 100 ms per hop is acceptable for the intended application scenario of a combined recommendation and reputation system. This is especially the case in the light of the offered anonymity properties and the fact that we evaluated a first prototype, which can still be optimized for performance.

5.4.3 Can We Trust Pseudonyms?

In face-to-face interactions, a multitude of inputs is available to a communication or interaction partner, like mimics, gestures, voice modulation etc., that on a subconscious level influence the initial feeling or disposition to trust. Therefore, we tend to agree that a feeling of trust can be built faster during face-to-face interactions than online, as we described in Section 2.1. On the other hand, according to psychologist Christiane Eichenberg [Eic05], the absence of these additional inputs in online communication can also lead to overcoming distance and a feeling of closeness. She states that interestingly, people using chat tend to act more open and honest during that time since they feel secure behind their pseudonymity.

From the point of view of trusting, the pseudonyms offered by our approach are not so much different from the identities used in most Internet-based services nowadays. Switching pseudonyms in our approach is as easy as creating a new public-private key pair, creating a number of routing headers and publishing these routing headers in the DHT. The cost of switching pseudonyms, as mentioned in [FR01], is starting over with zero trust and having to gain a reputation. Switching to a new email address (or even forging one) or a nickname in IRC is a matter of seconds, with the cost being the same: the reputation has to be built again.

Despite these facts, *virtual communities do exist* nowadays and *people are making friends* via IRC, Instant Messenger clients or multi player online games, with people that they may have never seen in real life nor do they know, whether these peoples' names, age, gender, and possibly personality etc. resemble reality at all. The actions that these entities perform lead to positive or negative experiences and finally to trust being built. This trust and interest in the person behind the pseudonym often leads to real face-to-face meetings (possibly with some surprises) and to friendships outside the purely online boundary.

In the light of these facts, we see that *trusting pseudonyms is possible* and already done today.

5.5 Summary

In this chapter, we focused on the protection of the privacy of all participants of the UniTEC system via the use of pseudonyms. We identified the requirements towards the identity management solution, most notably providing connectivity while ensuring the strength of the pseudonyms. After investigating the related work in traditional identity management solutions, anonymous communication techniques, and reputation systems, we found that none could fulfill all posed requirements adequately.

Our approach of extended destination routing is based on Chaum mixes. Routing headers are formed for each pseudonym containing multiple layers of encryption. In the middle of this structure lies the IP address of the pseudonym owner's UniTEC agent. Around the IP address are layers of encryption for each Mix that is to be used protecting the pseudonym's link to the IP address. Both, a pseudonym's public key and each routing header are stored under the key of the pseudonym's identity via a protected mechanisms in the DHT. To send a message from an own pseudonym to a target pseudonym, the routing header of the target pseudonym is retrieved from the DHT. The recipient-controlled part of the cascade in the routing header is extended with a sender-controlled part. Finally, the message is sent on to the first mix and will after passing through the chain of mixes eventually arrive at the recipient.

Our security evaluation of the approach shows, that we were able to meet the security requirements posed towards the identity management solution. The performance of the prototype's ACC, based on Java object serialization over TCP sockets, is more than adequate for the reputation system scenario.

Chapter 6

Trust Representation and Dynamics – Modeling and Updating Trust

Researchers in the area of trust and reputation systems have put a lot of effort in developing various trust models and associated trust update algorithms that support users or their agents with different behavioral profiles. While each work on its own is particularly well suited for a certain user group, it is crucial for users employing different trust representations to have a *common understanding* about the meaning of a given trust statement. Our results presented in this chapter provide such a common understanding by introducing our approach for a *generic trust model*. Additionally, we show how a selection of several well-known trust-update algorithms can be plugged easily into the UniTEC system. These results were published in two full papers at iTrust2003 [KR03] respectively iTrust2005 [KBR05].

We structure this chapter as follows: In the next section we motivate our work on a general trust model and point out briefly the benefits for a distributed reputation system. We then cover some definitions of trust, reputation and experiences that are used in the following to define the dimensions of trust relationships. After that, we present in detail the components of the generic trust model with respect to the used trust and certainty measures, the realization of the trust context and the area dependencies, and support for the dynamics of trust. We then introduce a selection of trust update algorithms from the related work, which have raised significant attention in the trust research community, and describe how the mappings between the generic trust model and the algorithm-specific trust models are performed. Finally, we present the results of our comparative evaluation of these algorithms under a selection of test scenarios.

6.1 Motivation

One aspect that research in trust and reputation systems strives to determine, is a suitable computational representation of trust, commonly referred to as a *trust model*. Tightly interwoven with trust models are the algorithms used to determine, how this trust is updated according to different usually discrete events. Such events might be a new experience with the person in question, or new information from other trusted sources regarding the reputation of this person etc.

Numerous different models and trust update algorithms have been proposed in the literature and each approach is particularly well suited for a certain user group or application area, as we pointed out in Subsection 2.1.3. However, these trust models are not interoperable since there is a lack of a generic representation of trust. In the words of Ruohomaa et al. [RK05] if "[...]a reputation statement says that a user is trustworthy by '3 on a scale from 1 to 5', what does it mean in the receiver's context?" A generic trust model would allow users intending to use different models to translate their local representation to the generic one in order to understand each other's trust statements.

Our contribution is built on the observation that, although the algorithms used to compute a certain trust value are quite different from each other, the data that the algorithms are working upon and the outcome of the algorithms are not that different and can thus be mapped onto a generic model. We suggest one approach for such a generic representation which we implemented in the context of UniTEC. This generic trust model allows us to easily integrate various existing trust update algorithms as presented in this chapter.

6.2 Terms and Definitions

In this section, we briefly define our understanding of the concepts of *trust* in the sense of local or inter-personal trust, *reputation* in the sense of overall or global trust, and *experience* as the event that the trust update is based on.

Since Chapter 2 deals in depth with the scientific background on trust, with a particular focus on the social science view, we want to highlight here only briefly the definition of *trust* that our trust model relies upon. One definition commonly found in the area of reputation systems is of Diego Gambetta [Gam00] which we pointed out in Subsection 2.1.3: "... trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action...". Lik Mui [MMH02] adapts this definition slightly in emphasizing the importance of an interaction history: "Trust: a subjective expectation an agent has about another's future behavior based on the history of their encounters". In the following we adopt Lik Mui's definition.

The *reputation* of entity A in our view is the *average trust* (whatever average means in that context) of *all other entities* towards A. Therefore reputation clearly has a global aspect whereas trust is viewed from a local and subjective angle.

As highlighted in Chapter 3, trust of entity A in an entity B in a certain context area is built by making *experiences* with that entity in the context area in question. An experience in this context is therefore an observation of A about some behavior of entity B. It is necessary to be able to judge whether an experience has been positive or negative in order to update A's trust in B.

6.3 Dimensions of Trust Relationships

In Subsection 2.1.3 we pointed out various aspects of trust which influence our understanding and modeling of a trust relationship. The subjectivity of trust recognized there leads consequently to asymmetric trust relationships between *trustor*, the entity who is trusting, and *trustee*, the entity who is trusted. In the following, we identify various *dimensions of trust relationships* in addition to trustor and trustee:

Trust measure refers to the quality of the trust relationship, which ranges from complete distrust over a neutral trust measure to full trust. The more a trustee is trusted, the higher the trust measure is supposed to be.

Trust certainty specifies the confidence of the trustor in his or her estimation of the trustee. If this estimation is gained via only few personal experiences or just via word of mouth, the certainty is supposed to be low.

Trust context People trust in a fine-grained manner depending on the area and goal in question, as we pointed out in the discussion of generalization as concept for learning trust on page 21. Person A, for instance, might trust person B to babysit her child whereas she might not trust person B to repair a computer. In [KR03] we use the term category to specify a trust context. Other terms for trust context commonly found in the literature are *trust scope* or *trust purpose*.

Trust directness This aspect distinguishes between *functional trust* and *referral trust*. Functional trust means that the trustee can directly cooperate with the trustor. With referral trust, the trustor does not trust the trustee functionally but believes in the ability of the trustee to forward the cooperation request to one or more other good experts. Consider for instance person A knowing that person B has many friends working in the computer business, although B is not schooled in this context herself. A will not trust B functionally with a repair task but might very well trust recommendations received via B from one of B's expert friends. This concept is often expressed as direct versus indirect trust [JGK03, ARH00]. However, in our view, direct trust (both, functional and referral direct trust) is based on experiences, whereas indirect trust is based on computational trust combination, as we point out in Chapter 7 respectively [JGK06].

Trust dynamics A trust relationship is not static, but changes dynamically on various different incidents, e.g. on *own direct experiences* as we discussed in Subsection 2.1.5). If for instance the babysitting of A's child by B went well, the trust of A in B will increase. In addition to own experiences, *trust estimations received from others* influence the own trust assessment as well (as pointed out on page 19). If A's good friends C, D, and E warn A about the unreliable nature of B, A might refrain from relying on B's babysitting capabilities. Lastly, quite interestingly, *trust relationships may also change over time* when no experiences have been made, a fact that is up to our knowledge not yet covered in the related work.

6.4 Towards a Generic Trust Model

Having presented the general concepts of trust relationships in the previous section, we describe in the following how these concepts are mapped on the components of our generic trust model. The key components of our model result from an analysis of the characteristics of various existing trust models and the inputs gained from trust research in the social sciences (see Section 2.1).

6.4.1 Trust Measure and Certainty

Various different representations of trust values exist in the related work. Trust values can be depicted as real numbers in certain intervals like for instance $[-1,+1]$, as done by Jonker and Treur [JT99] and Sabater [Sab03] or probabilities in $[0,1]$, as proposed among others by Jøsang and Ismail [Jøs02], Yu and Singh [YS02], and Kinateder and Rothermel [KR03]. Others propose discrete values, like the binary representation by Blaze and Feigenbaum [BFL96] or four discrete values introduced by Abdul-Rahman and Hailes [ARH00].

The metric used for the *trust measure* in our proposed generic trust model is a real number in the interval of $[0,1]$. Complete distrust is represented by 0 whereas 1 corresponds to full trust. This representation allows an easy transformation of any previously described measures in the generic measure as we will see in more detail in Section 6.5.

Not all investigated algorithms support the computation of a certainty value, which states the quality of the trust assessment represented in the trust measure. If uncertainty is mentioned [YS02, Jøs02, Sab03] it is specified in the interval $[0,1]$.

The *trust certainty* in the generic trust model is represented similarly to the trust measure as a number in the interval of $[0,1]$, whereas 0 describes complete uncertainty and 1 the total certainty.

6.4.2 Trust Context

As pointed out in the previous section, applications can define various context areas in which entities can be trusted. It is important to note that these areas are not necessarily independent from each other. Different kinds of dependencies can exist among the context areas: *instance-of* relationships are one-level relationships for *classification*, *is-a* relationships provide *generalization*, *part-of* relationships enable *aggregation* and surely many other – potentially application-specific – forms of dependencies between context areas can be imagined.

Chen and Singh are considering this effect up to a certain degree in their work [CS01], as they are organizing trust categories in a hierarchy by using *instance of* relationships with the leaf categories holding the comments (or experiences as we call them) and the trust of the non-leaf categories being calculated from them. We found two drawbacks

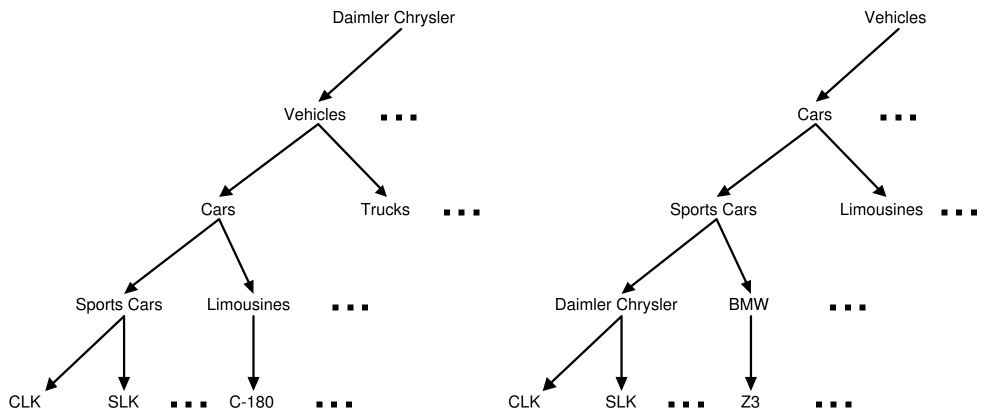


Figure 6.1: Context area dependencies modeled via a hierarchy. This figure shows the limitations of the hierarchical approach. Both ways of modeling the dependencies make sense, but only one can be used.

of this solution: Firstly – as illustrated in the example in Fig. 6.1 – the hierarchy is not a suitable approach for all sets of categories. The left arrangement in the figure makes perfect sense, since persons with expertise in Daimler Chrysler sports cars will most certainly also possess a certain knowledge about other cars of this company. Yet the right side models the fact that experts in Daimler Chrysler sports cars know something about sports cars in general. Thus a single hierarchy is obviously not enough for grasping the dependencies. Secondly, relationships between categories do not necessarily need to be of the type *instance of*, thus modeling trust in such a way limits the general usability of the model significantly.

In our approach for realizing trust context, we model the *asymmetric semantic distance* between the context areas and therefore abstract from the actual kind of dependency. The metrics chosen for the semantic distance is a real number in the interval of $[0,1[$. A distance close to 1 represents a high dependency, a distance of 0 refers to no dependency. Therefore, we organize the trust context areas as a weighted directed graph as can be seen in Fig. 6.2. This allows us to spread the impact of a trust update in one area to related areas and control this influence through the weights. The context areas and the semantic distances between the areas are specified by the applications to be supported with trust management. However, due to the subjectivity of trust, each user can locally modify the distances to suit his or her personal views[1].

[1]Supposedly, the user knows best the existing relationships between his or her areas of interest. For future work, it might be challenging to investigate whether and if yes how these relationships can be detected and if this process can be automated.

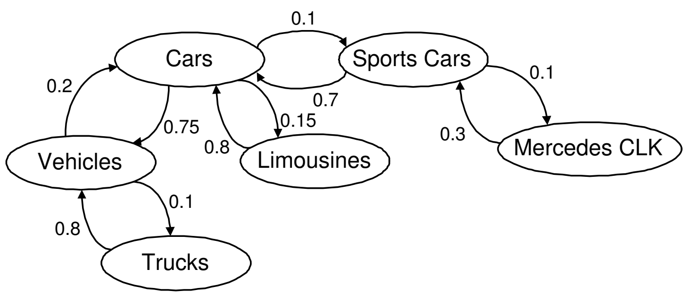


Figure 6.2: Example snippet of a weighted directed trust context graph. The weights represent sample semantic distances of the context areas.

This is an example about how part of such a *dependency graph* may look like with the focus on a context area “Mercedes CLK”. A good experience with a review on the Mercedes CLK of recommending identity *I* will for instance lead to an increase in *E’s* trust in *I* in the Mercedes CLK category and also (to a much lesser degree with 30% impact as opposed to a direct experience in sports cars) to an increase in trust in sports cars and so on. The weights presented here are example values to illustrate the idea and are not validated against reality.

6.4.3 Trust Directness

Another dimension of a trust relationship is its directness. In the current implementation of our model in the UniTEC prototype, functional and referral trust are two distinct model instances, each with a specific trust measure, certainty etc. They are stored and updated separately by the trust algorithms.

6.4.4 Trust Dynamics

As already mentioned in Chapter 3, a change in trust occurs upon receipt of feedback regarding an experience of a trustor with a trustee. Various aspects are discussed in the following that influence the trust dynamics.

Quality Feedback The trustor provides feedback about the subjective quality of a received information item. The metrics used to rate the quality is a real number in the interval of $[0, 1]$. A perfect information item is rated with 1, 0 describes a completely unsatisfactory one. The generic trust model does not dictate how this feedback is gained; e.g. for recommendations of a static attribute-value structure, this feedback can be gained automatically by collaborative filtering.

Trustor Confidence Trustors may specify a confidence in their own offered information items. This confidence is represented by a real number in the interval of $[0,1]$. Similar to the trust certainty, 0 stands for no confidence whereas 1 stands for the highest possible confidence in the offered information. This confidence influences the trust update such that a weak statement with a low confidence leads to only a slight trust update, whether positive or negative.

Transaction Utility Each information request, and the corresponding responses and feedback statements refers to a certain transaction the requester or trustor is about to take. Depending on the transaction's significance, the trustor specifies the utility as a real number in the interval of $[0,1]$. We assume that a "maximum utility" can be specified in such as utilities higher as this maximum utility will lead to the same trust update impact as with the specified maximum utility. 1 refers to the normalized maximum utility which leads to a trust update with a higher impact.

Experience Aging The quality of trustees is not necessarily constant but may change over time, for instance due to gathered experience in a certain field. In order to determine trust as a prediction of the future behavior, it is possible to specify that the latest experiences ought to weigh more than older experiences. We propose two options for experience aging: a *feedback window* and an *experience aging factor*. The feedback window limits the amount of considered experiences, either depending on a certain number of experiences or a certain maximum age. The aging factor in the interval of $[0,1]$ determines the ratio of a new experience to previous experiences in the update computation. We describe in the following section how this aging factor is used in the algorithms.

Related Trust Context Areas As mentioned before, an update in a single trust context area A may lead to an update of a lesser extent in related areas B_i according to the relationships in the context area graph. The semantic distance between two context areas that are linked via one or more intermediary areas can be computed by calculating the product of the semantic distances along the path. The proportion of the update of B_i to A is determined by the strongest semantic distance from A to B_i, in other words by calculating the maximum product of all paths from A to B_i. Context area that cannot be reached from A or where the distance is not known are not updated.

Trust Fading When no experience with a trustee is made in a long time, the old trust relationship might no longer be valid. This usually means that the trust confidence level decreases over time. There might also be situations or time frames when the trust level itself decreases without new experiences. We represent the magnitude of this fading effect with a *fading factor* with a real number $\lambda \geq 0$. A factor of 0 means no fading effect. The higher the fading factor, the faster the trust relationship drops back to the initial state specified by the trust algorithm. This state might even be a state of no trust and no confidence. Fading needs to be explicitly supported by the trust update algorithm[2].

6.5 Trust Update Algorithms

The subset of investigated algorithms discussed here are Abdul-Rahman–Hailes, Beta Reputation, ReGreT and the original UniTEC algorithm.

6.5.1 Abdul-Rahman – Hailes

The work on a trust model in [ARH00] is based on sociological studies similar to the work of Marsh [Mar94]. Here, interpersonal trust is context-dependent, subjective and based on prior experiences. A reputation information exchange amongst members of the community assists with trust decisions. All these aspects fit well in our generic trust model.

Trust is measured in a discrete metric with four values: very untrustworthy vu, untrustworthy u, trustworthy t and very trustworthy vt. Abdul-Rahman and Hailes describe three uncertainty states, which complement the four trust values: more positive experiences u^+, more negative experiences u^-, and equal amount of positive and negative experiences u^0. Ratings, in our words experiences and the corresponding feedback, are specified in a discrete metric: very bad vb, bad b, good g, very good vg.

[2]A simple fading algorithm has been implemented for the UniTEC trust update algorithm, as described in the following section.

To fit this into our generic trust model, the discrete trust values have to be mapped onto the scalar trust metric. The range $[0,1]$ is split into three equally sized ranges. The values at the borders of these ranges represent the four values from the discrete metric $(0, \frac{1}{3}, \frac{2}{3}, 1)$. Each discrete value is assigned a single position number ($pos(0) = vu$, $pos(1) = u$, $pos(2) = t$, $pos(3) = vt$). To map a value from the generic trust model (t_g) onto this discrete metric (t_d), the following calculation is applied: $t_d = pos(round(t_g \cdot 3))$. The same formulas are applied for mapping the four discrete rating values. This translation follows the reasoning that trust in this model cannot be greater than *very trustworthy*, which is represented by the value of 1 in the generic trust model.

The semantics of the uncertainty values are not defined in [ARH00], therefore the mapping into our generic trust model is difficult. For the four trust values, no uncertainty is known, which is represented as a certainty of 1 in our generic trust model. The initial trust value (when no previous experiences are known) is represented by the u^0 uncertainty value, so it makes sense to keep the generic certainty value of 0 (the generic trust value is of no importance in this case, so we also keep it at the lowest level of 0). Rahman's paper does not give an explanation on how to interpret this uncertainty value in other situations. The uncertainty values u^+ and u^- represent states where slightly more positive (or negative) previous experiences have been recorded. This is expressed in our generic trust model by a slight mistrust $(1/3)$ or a small positive trust $(2/3)$. In these cases the uncertainty component is represented by a mean generic certainty value (0.5).

6.5.2 The Beta Reputation System

The Beta Reputation System [Jøs02] is based on Bayesian probability. The *posteriori* probability of future positive experience is represented as a beta distribution based on past experiences. The trust value, in this work called "reputation rating", is determined by the expectation value of the corresponding beta distribution. This is a probability value in the scalar range $[0,1]$. A one-to-one mapping to our generic trust value is possible.

The certainty of the trust calculation is defined by mapping the beta distribution to an opinion, which describes beliefs about the truth of statements [Jøs01, JDV03]. In this mapping the certainty starts at 0 and grows continuously to 1 with more experiences being considered. This metric also can be directly mapped to our scalar generic certainty metric $[0,1]$.

Experiences in the Beta Reputation System are rated through two values: $r \geq 0$ for positive evidence and $s \geq 0$ for negative evidence. The sum $r+s$ represents the weight of the experience itself. These two weighted rating values can be mapped to the generic rating value ($0 \leq R \leq 1$) and the generic weighting metric ($0 \leq w \leq 1$) as $r = w \cdot R$ and $s = w \cdot (1-R)$.

In this trust model, the accumulation of ratings can make use of a so-called forgetting factor, which is the equivalent to the generic aging factor. In the Beta Reputation System the forgetting factor (λ_{beta}) has a reversed meaning compared to our definition: $\lambda_{beta} = 1$ is equivalent of having no forgetting factor and $\lambda_{beta} = 0$ means a total aging (only the last experience counts). Thus $\alpha = 1 - \lambda_{beta}$ represents a simple mapping to our generic aging factor.

6.5.3 The ReGreT System

The ReGreT system [Sab03] represents a reputation system which uses direct experiences, witness reputation and analysis of the social network in which the subject is embedded to calculate trust.

Direct experiences are recorded as a scalar metric in the range $[-1,+1]$. Trust is calculated as a weighted average of these experiences and uses the same value range. A mapping to the generic values can be done by transforming these ranges to $[0,1]$ (shifting and scaling appropriately).

For each trust value, a reliability is calculated based on the number of outcomes and the variation of their values. This reliability is expressed as a value in the range $[0,1]$ which directly matches the representation of our generic certainty value.

An aging factor is not used. Instead, the oldest experience is neglected ($w = 0$), the newest experience is fully weighted ($w = 1$). The weight of experiences in between grows linearly from 0 to 1.

6.5.4 The Original UniTEC Algorithm

In the first publication of the UniTEC project [KR03] we introduced a simple trust update algorithm based on geometric learning:

$$T_{new} = (1-(a \cdot c \cdot d)) \cdot T_{old} + (a \cdot c \cdot d) \cdot E \tag{6.1}$$

The *new trust* T_{new} in the range of $[0,1]$ in each *context area* is calculated from the *old trust* T_{old} (also in $[0,1]$) and the *new experience* E and is influenced by an *aging factor* a, the recommender's own *confidence* in the recommendation c and a *distance factor* d depending on the distance of the context area in question from the original context area of the recommendation in the context dependency graph.

The aging factor in the range of $]0..1[$ is specific to each context area and influences how fast new experiences of the requester change the trust compared to previous experiences. The confidence factor influences the impact of a certain recommendation on the trust update and is under the recommender's control. The distance factor handles the influence of an experience in the original context area on related context areas in the dependency graph. d is the maximum of the products of weights for all paths in the dependency graph from the original context area to the context area in question. Every trust update in a related context area leads to an increase in its number of indirect experiences in the trust model. For efficiency reasons it is necessary to impose a certain limit on the trust update of dependency categories for long distances (reflected in a small d).

Ratings in this original UniTEC paper are expressed as a binary metric of $\{0,1\}$ (either bad or good experience). However, the trust update algorithm works much the same with ratings in a scalar range of $[0,1]$. Therefore, no further mapping to the rating metrics of the generic trust model is required.

In UniTEC of [KR03] we specified the certainty of the trust assertion through a confidence vector, where the amount of direct and indirect experiences and a trail of the latest n direct experiences is recorded. We calculate the certainty as a single scalar metric as in the generic trust model. This can be accomplished in a similar manner as in the ReGreT System, where the number of experiences and the variability of its values are consolidated into a single value in the range $[0,1]$.

We created a simple fading algorithm that works with the UniTEC update algorithm and uses the fading factor λ. Time is split in discrete time units. If in an interval, one or more experiences are made, these are executed and fading does not take place. When no experiences are recorded, trust will drop linearly to the minimal trust value in $1/\lambda$ time units.

6.6 Test Scenarios

To assess the quality of the trust update algorithms presented in the previous section, a series of test scenarios was developed. Each scenario simulates a different behavior pattern of trustor and trustee as a list of ratings. This pattern is then reflected by each single trust algorithm as trust dynamics. A test scenario can bring forward a specific feature or a malfunction of a trust update algorithm.

As the test scenarios simulate the behavior of real-world people, there are certain expectations associated with the trust dynamics. A trust algorithm is expected to generate trust dynamics that satisfy these expectations. Failing to comply with the specified expectations can either be a consequence of the calculations themselves or it reflects a shortcoming of the adaptations and mappings necessary for the local trust model to work in the generic trust model.

We want to stress that due to the subjectivity of trust, as indicated in social science research (see 2.1.3), also the quality estimation of the behavior reflected in the trust dynamics is subjective. Therefore, we do not offer a ranking of trust update algorithms, but instead point out the distinctive features of the algorithms, so that each user can choose the algorithm that most closely reflects his or her own expected behavior.

OnlyMaximalRatings Starting from the initial trust state, only maximal ratings (= 1) are given. We would expect the trust to grow continuously and approach the maximal trust value (= 1).

OnlyMinimalRatings Starting from the initial trust, only minimal ratings (= 0) are given. If the initial trust is the minimal trust value (= 0), then trust should stay at this level. Otherwise, we would expect the trust value to decrease and eventually approach the minimal trust value.

MinimalThenMaximalRatings First, a series of minimal ratings is given, which is followed by a series of maximal ratings. We would expect the trust dynamics to start as described in the test scenario *OnlyMinimalRatings*. After switching to maximal ratings, trust should rise again. The expected growth rate of trust after the start of the maximal ratings should be lower than in the *OnlyMaximalRatings* test scenario.

MaximalThenMinimalRatings First, a series of maximal ratings is given, followed by a series of minimal ratings. We expect trust to rise as in the test scenario

OnlyMaximalRatings. When the series of minimal ratings starts, trust should decrease again. The trust decrease rate in the second half of the test should be slower than in the test scenario *OnlyMinimalRatings*.

SpecificRatings After the previous test scenarios, which work with extreme ratings, these four test scenarios make use of a specific set of ratings (the *SpecificRatings*) which simulate a real-world rating situation. The ratings are: 1.0, 0.8, 0.5, 0.4, 0.5, 1.0, 0.6, 0.7, 0.7, 0.8, 1.0, 0.4, 0.3, 0.2, 0.2, 0.5, 1.0, 0.3, 0.4, 0.3. These ratings are submitted in this original order (-*Normal*), in a reversed order (-*Reversed*), ordered by ascending (-*OrderedAsc*) and by descending rating value (-*OrderedDesc*). In the *Normal* order, there are more positive ratings in the first half of the rating sequence, whereas negative ratings predominate in the second half. The expectation is that the final trust value is slightly below the mean trust value ($= 0.5$). With the *Reversed* order the expectation for the final trust value is a value slightly above the mean trust value. If the ending trust values in all four scenarios are equal, it suggests that the trust algorithm uses an *indistinguishable past* (see [JT99]), which means that the order of previous experiences does not matter. This should not be the case when using an aging factor.

KeepPositive This scenario has a *dynamic* nature, in that it actively reacts on the resulting trust values after each individual rating. Maximal ratings are given until a certain level of trust is reached (> 0.8). This trust level is then "misused" in form of minimal ratings, until the trust value reaches a mistrust level (< 0.5). Then, maximal ratings are submitted to raise trust again. This process is repeated four times. Here the trust algorithm's reaction to attempts of misuse is analyzed. We would expect trust to quickly drop to a mistrust level when the minimal ratings occur. The *optimal algorithm* should quickly detect a misuse attempt and react appropriately, e.g. by reporting minimal trust or even blacklisting the user.

6.7 Evaluation

We subjected each trust update algorithm discussed in Section 6.5 with a variation of aging factors to each test scenario from the previous section. For each evaluation graph we use the representation of the algorithms presented in Fig. 6.3.

Test scenario *OnlyMaximalRatings* presented in Fig. 6.4 illustrates the different initial trust values of the algorithms: The trust dynamics start either with a trust value

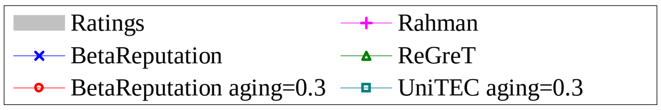


Figure 6.3: Key for the trust dynamics presented in the evaluation

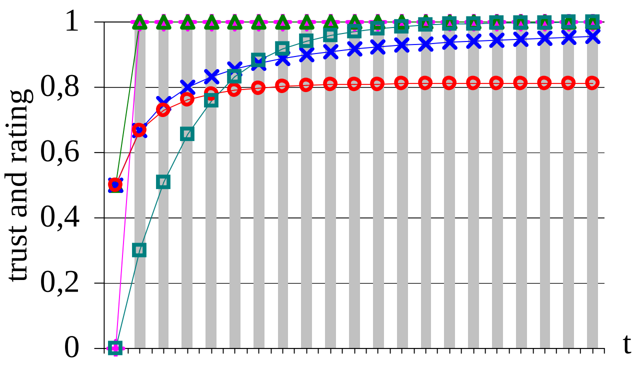


Figure 6.4: Trust update algorithms evaluation: Scenario *OnlyMaximalRatings*

of 0 (UniTEC and Abdul-Rahman) or 0.5 (Beta Reputation and ReGreT). Trust rises monotonously for all algorithms. Trust in Beta Reputation with no aging factor and UniTEC approaches asymptotically the maximum trust value. Beta Reputation with an aging factor approaches a certain level of positive trust value. ReGreT and Abdul-Rahman reach maximum trust after just one maximal rating and remain at this level.

Similar effects can be noticed in the test scenario *OnlyMinimalRatings* (see Fig. 6.5). The trust algorithms that started with the lowest trust value (UniTEC and Abdul-Rahman) stay at this minimum trust level. ReGreT that started at 0.5 drops to the lowest trust value after just one bad experience. Beta Reputation also started with a trust value of 0.5. Without aging factor, trust approaches asymptotically the minimum

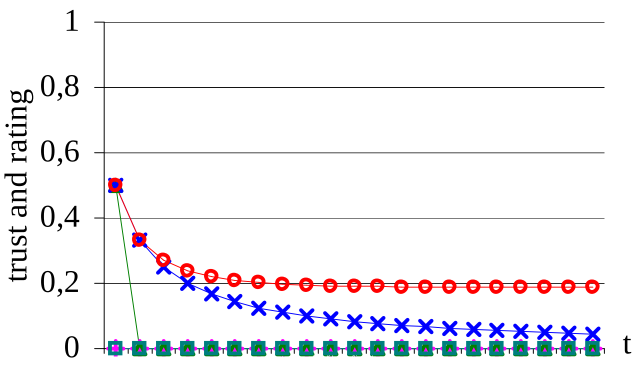


Figure 6.5: Trust update algorithms evaluation: Scenario *OnlyMinimalRatings*

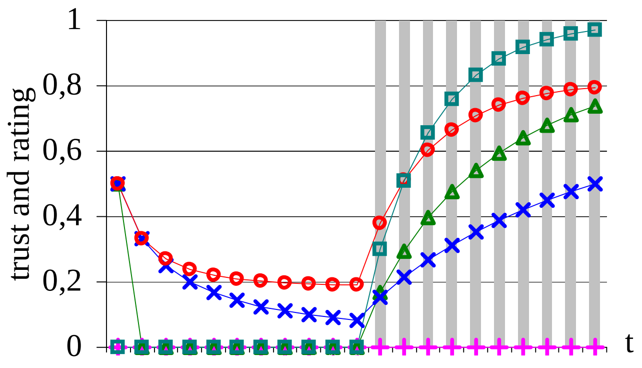


Figure 6.6: Trust update algorithms evaluation: Scenario *MinimalThenMaximal*

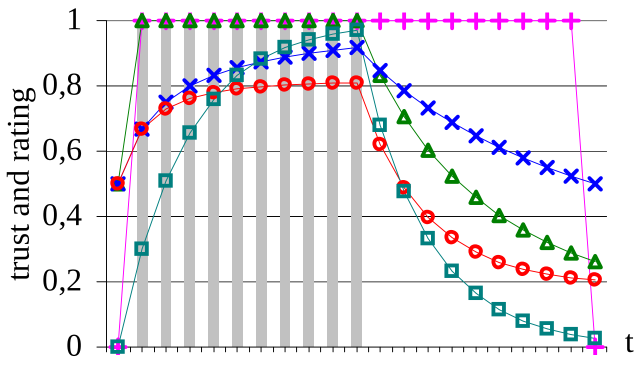


Figure 6.7: Trust update algorithms evaluation: Scenario *MaximalThenMinimal*

trust value. With an aging factor, trust never drops below a certain level of mistrust.

Beta Reputation with an aging factor uses only a limited interval of the totally available trust value scope. This happens because the accumulation of the evidence (r and s) using a positive aging factor represents a geometric series. An upper limit for r and s thus limits the possibly reachable maximum and minimum trust values. One possible solution to make use of the whole range regardless of an aging factor is to scale the possible output range to the whole generic trust value range. This can only be done if the aging factor remains constant throughout the relevant rating history.

In *MinimalThenMaximalRatings* (Fig. 6.6) when the maximal ratings start, trust starts rising again in all analyzed algorithms but Abdul-Rahman's. In this latter case, trust remains at the lowest level until as much maximal ratings as minimal ratings have been received. The discrete metrics of this trust model does not support other intermediate states. Another interesting observation is that UniTEC shows the same rise on trust as in the *OnlyMaximalRatings* test scenario (rising above 0.8 after 5 maximum ratings). The other algorithms show a slower rise of trust, as we would expect after the negative

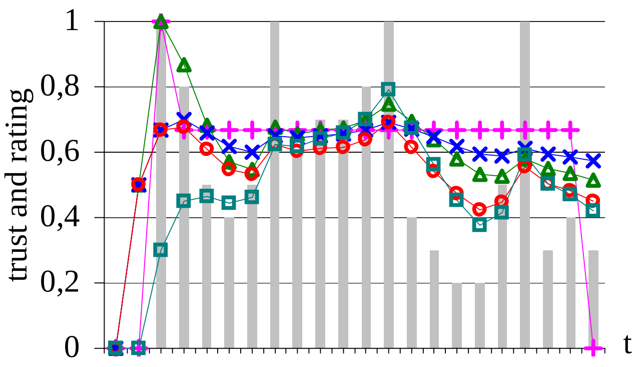


Figure 6.8: Trust update algorithms evaluation: Scenario *SpecificRatingsNormal*

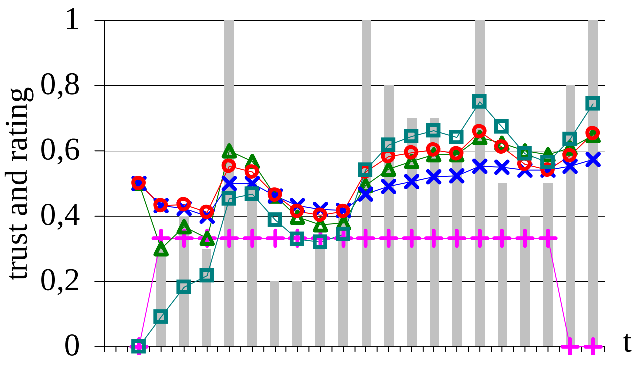


Figure 6.9: Trust update algorithms evaluation: Scenario *SpecificRatingsReverse*

impact of the negative ratings. This demonstrates one deficiency of algorithms like the original UniTEC one, which rely solely on the last trust value and the new rating for their calculations and do not consider adequately the remaining history of ratings.

The *MaximalThenMinimalRatings* test scenario (Fig. 6.7) shows similar results as the previous scenario. It starts as expected like the *OnlyMaximalRatings*. When the minimal ratings start, trust drops with all but Abdul-Rahman's algorithm. Here, trust suddenly drops from maximum to the minimal value at the end of the scenario which is the point when more minimal than maximal ratings are recorded in the history.

In both scenarios we notice that Beta Reputation without an aging factor shows a slow reaction to the pattern change in the ratings.

The *SpecificRatings* test scenarios are depicted in Figs. 6.8, 6.9, 6.10, and 6.11. In *SpecificRatingsNormal*, most algorithms follow the expected trust dynamics. The ending trust value for UniTEC and Beta Reputation with aging factor of 0.3 is just below the average trust value of 0.5. ReGreT and Beta Reputation without an aging factor are

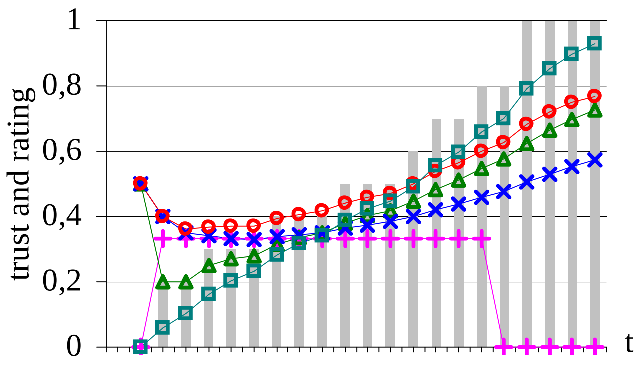


Figure 6.10: Trust update algorithms evaluation: Scenario *SpecificOrderedAsc*

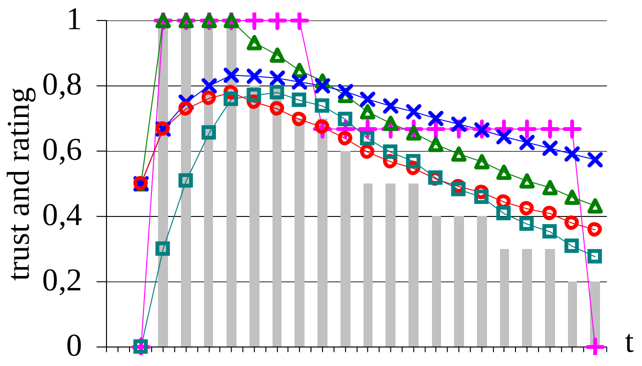


Figure 6.11: Trust update algorithms evaluation: Scenario *SpecificOrderedDesc*

a bit more optimistic, ending just above 0.5. In *SpecificRatingsReversed* the opposite can be seen: The ending trust value is just above the trust value mark of 0.5.

In those four test scenarios, Abdul-Rahman generates a trust dynamic that follows our expectations up until the end, when suddenly trust and certainty drop back to the lowest values. What happened here is that the algorithm reached the uncertainty state u^0. The most evident problem with this can be seen in Fig. 6.10. At the last couple of ratings this state of uncertainty is reached, which is not expected at all. The weakness lies in the lack of semantical meaning of the u^0 state. The only solution would be to alter the original algorithm and its underlying trust model to improve the way uncertainty is handled.

We see the characteristic of *indistinguishable past* with Abdul-Rahman and Beta Reputation without an aging factor: The ending trust values are the same regardless of the ordering of the ratings. All remaining algorithms use aging of ratings, leading to different ending values depending on the order of the ratings.

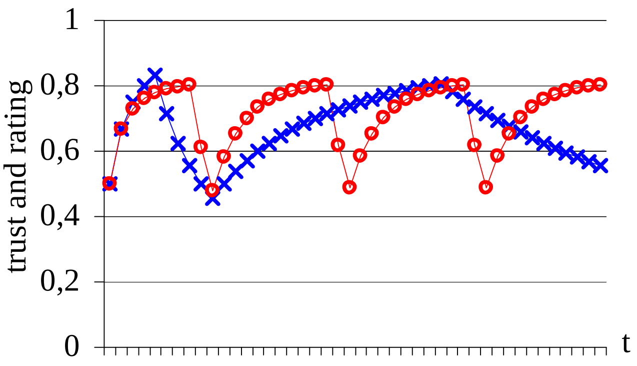


Figure 6.12: Trust update algorithms evaluation: Beta reputation in scenario *KeepPositive*

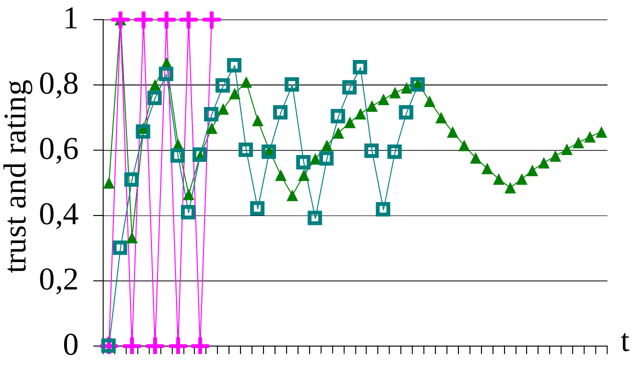


Figure 6.13: Trust update algorithms evaluation: Alfarez-Rahmann, ReGreT and UniTEC in scenario *KeepPositive*

In our last test scenario, *KeepPositive*, the rating history depends on the calculated trust values. Fig. 6.12 shows how an aging factor can improve the Beta Reputation's reaction to the sudden minimum ratings, while it also makes the reaction speed more independent of the total history size. Without an aging factor, the dynamics of trust get more steady as the history of ratings grows. The reaction of the remaining algorithms to this test scenario can be seen in Fig. 6.13. Abdul-Rahman cannot really compete due to the lack of precision: After just one maximum or minimum rating trust flips from minimum to maximum and back. ReGreT shows a deficiency similar to Beta Reputation without aging factor: As more experiences are recorded, trust dynamics react slower to rating pattern changes. UniTEC shows fast reaction to the minimum ratings while maintaining this reaction independently of the history size.

6.8 Summary

In this chapter, we investigated the various dimensions that a trust relationship can cover, and highlighted how these dimensions are reflected in our generic trust model. The trust model represents a generic representation of trust, that allows to plug in various existing trust update algorithms. Bijective mappings are provided that map the local models that each algorithm works with onto the generic model, and vice-versa.

We showed for the following four algorithms, how they can be plugged into the model and what mappings and adaptations are required: Abdul-Rahman–Hailes, Beta Reputation, ReGreT, and the algorithm presented in the first UniTEC publication [KR03]. Trying to summarize the quality of the algorithms is not an easy task. Each algorithm shows strengths in certain areas and weaknesses in others, and we cannot state that one of the algorithms shines in each of the scenarios. It is therefore true what was stated at the beginning of the chapter: trust is subjective, as is the choice of a suitable trust update algorithm.

The easy-to-understand discrete 4-step trust metric of Abdul-Rahman–Hailes may appeal to users more than the continuous metrics used in the remaining three algorithms. However, since the algorithm is based merely on counting experiences, it reacts slowly on changes in the behavior. In particular, an attacker can misbehave up to n times after having behaved properly $n+1$ times. Beta Reputation that relies on Bayesian probability operates on a sound statistical basis and shows good results. The limitation of trust range in case of aging can be overcome. The initialization of trust with a neutral value might encourage pseudonym switching and should be considered harmful. ReGreT meets the expectations, but shows deficiencies that it reacts slower on behavioral changes with increased history size. Reducing the history size by an experience window or adapting the aging function accordingly might address this issue. Finally, the geometric learning of the UniTEC algorithm performs well during most scenarios. One drawback is, however, that the old trust value before the update does not capture the history of past experiences sufficiently in all cases. This becomes especially apparent in the MinimalThenMaximal scenario.

The work presented in this chapter takes into account numerous trust dimensions and most notably allows a fine-grained modeling of trust in interdependent context areas. This generic model allows to compare the quality of trust computation of different trust update algorithms.

Chapter 7

Trust Transitivity – Evaluating Trust Networks

In Chapter 3 we presented an overview how the UniTEC prototype disseminates requests and through this dissemination forms a trust chain. Since a recommender can be reached via various different paths from a requester, the resulting structure of the combined trust chains forms a trust network. The individual trust statements of the network need to be combined to finally allow an estimation of the requester's transitive trust in the recommender. Here we focus on the concept of *trust transitivity* and describe an approach for *trust network analysis* using *subjective logic* (TNA-SL). The results presented in this chapter were gained during joint work with *Prof. Audun Jøsang* from Queensland University of Technology in Brisbane, Australia and *Elizabeth Gray* from Trinity College Dublin, Ireland and published in a first version as a full paper [JGK03] at FAST 2003 and as a heavily extended and reworked version as a WIAS journal paper [JGK06].

TNA-SL consists of the following three elements: Firstly it uses a concise *notation* with which trust transitivity and parallel combination of trust paths can be expressed. Secondly it defines a method for *simplifying complex trust networks* so that they can be expressed in this concise form. Finally it allows trust measures to be expressed as beliefs, so that derived trust can be automatically and securely computed with *subjective logic*. We compare our approach with trust derivation algorithms that are based on normalization such as PageRank and EigenTrust and with the algorithm proposed in the first UniTEC publication [KR03]. We also provide a numerical example to illustrate how TNA-SL can be applied.

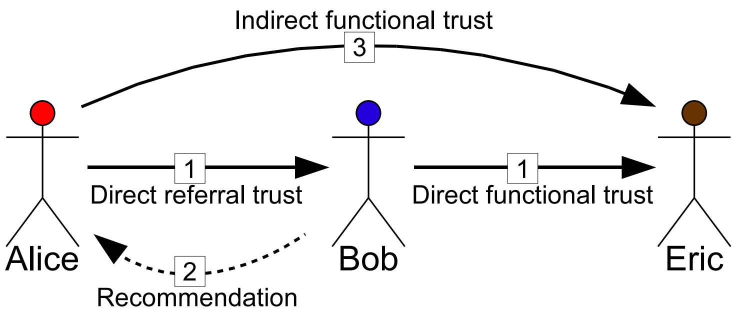


Figure 7.1: Transitive trust principle

7.1 Introduction

Trust transitivity means, e.g. that if Alice trusts Bob, who in turn trusts Eric, then Alice will also trust Eric. This assumes that Bob actually tells Alice that he trusts Eric, which can also be called a *recommendation*[1]. This is illustrated in Fig. 7.1, where the indexes indicate the order in which the trust relationships and recommendations are formed.

It can be shown that trust is not always transitive in real life [CH96]. For example the fact that Alice trusts Bob to look after her child, and Bob trusts Eric to fix his car, does neither imply that Alice trusts Eric for looking after her child, nor that she trusts him for fixing her car. However, under certain semantic constraints trust can be transitive, and a reputation system can be used to derive trust. In the example, trust transitivity collapses because the context area, that is the *trust scope*, of Alice's and Bob's trust is different.

Based on the situation of Fig. 7.1, let us assume that Alice needs to have her car serviced, so she asks Bob where to find a good car mechanic in town. Bob is thus trusted by Alice to know about a good car mechanic and to tell his honest opinion about that. Bob in turn trusts Eric to be a good car mechanic.

It is important to separate between trust in the ability to recommend a good car mechanic which represents *referral trust*, and trust in actually being a good car mechanic which represents *functional trust*. The context area or scope of the trust is nevertheless the same, namely to be a good car mechanic. Assuming that, on several occasions, Bob

[1]The reader should note that there is a difference between this recommendation and the recommendations or trusted data items that we have considered up to this point. The recommendation considered in this chapter is an explicit statement regarding the trustworthiness of another entity in a certain context. TDIs on the other hand may contain information regarding the trustworthiness of another pseudonym, but may also contain any other data item as outlined in Subsection 3.2.4.

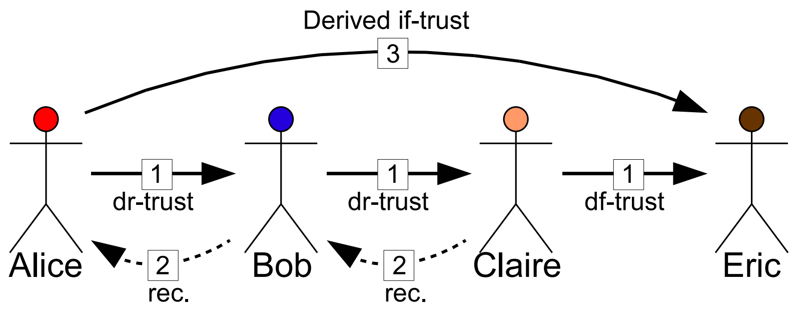


Figure 7.2: Extended transitivity example: Indirect trust is derived through transitivity.

has proven to Alice that he is knowledgeable in matters relating to car maintenance, Alice's referral trust in Bob for the purpose of recommending a good car mechanic can be considered to be *direct*. Assuming that Eric on several occasions has proven to Bob that he is a good mechanic, Bob's functional trust in Eric can also be considered to be direct. Thanks to Bob's advice, Alice also trusts Eric to actually be a good mechanic. However, this functional trust must be considered to be *indirect*, because Alice has not directly observed or experienced Eric's skills in car mechanics. This behavior is captured by the directness criterion of our trust relationship definition in Section 6.3.

Let us slightly extend the example, wherein Bob does not actually know any car mechanics himself, but he knows Claire, whom he believes knows a good car mechanic. As it happens, Claire is happy to recommend the car mechanic named Eric. As a result of transitivity, Alice is able to derive trust in Eric, as illustrated in Fig. 7.2.

Defining the exact context of Alice's trust in Bob is more complicated in the extended example. It is most obvious to say that Alice trusts Bob to recommend somebody (who can recommend somebody etc.) who can recommend a good car mechanic. The problem with this type of formulation is that the length of the trust context expression grows proportional with the length of the transitive path, so that it rapidly becomes impenetrable. It can be observed that this type of trust scope has a recursive structure that can be exploited to define a more compact expression for the trust scope. As already mentioned, trust in the ability to recommend represents referral trust, and is precisely what allows trust to become transitive. At the same time, referral trust always assumes the existence of a functional trust scope at the end of the transitive path, which in this example is about being a good car mechanic.

The "referral" variant of a trust scope can be considered to be recursive, so that any transitive trust chain, with arbitrary length, can be expressed using only one trust scope with two variants[2]. This principle is captured by the following criterion:

Definition 5 (Functional Trust Derivation Criterion) *Derivation of functional trust through referral trust, requires that the last trust arc represents functional trust, and all previous trust arcs represent referral trust.*

If trust scopes or trust context areas are built hierarchically to represent context area generalization[3], the trust chain does not need to contain links that all have the same scope. For example, knowing how to change wheels on a car is more specific than to be a good car mechanic, where the former scope is a subset of the latter. Whenever the functional trust scope is equal to, or a subset of the referral trust scopes, it is possible to form transitive paths. This can be expressed with the following consistency criterion:

Definition 6 (Trust Scope Consistency Criterion) *A valid transitive trust path requires that the trust scope of the functional/last arc in the path be a subset of all previous arcs in the path.*

Trivially, every arc can have the same trust scope, as we required for the UniTEC prototype. Transitive trust propagation is thus possible with two variants (i.e. functional and referral) of a single trust scope.

A transitive trust path stops if the first functional trust arc encountered when there are no remaining outgoing referral trust arcs. It is, of course, possible for a principal to have both functional and referral trust in another principal, but that should be expressed as two separate trust arcs. The existence of both a functional and a referral trust arc, e.g. from Claire to Eric, should be interpreted as Claire having trust in Eric not only to be a good car mechanic, but also to recommend other car mechanics.

The examples above assume some sort of absolute trust between the agents in the transitive chain. In reality trust is never absolute, and many researchers have proposed to express trust as discrete verbal statements, probabilities or other continuous measures

[2]We support this behavior in our trust model via the *trust directness* dimension.

[3]We describe in Subsection 6.4.2 the limitations of a hierarchical representation of context areas, and instead recommend a weighted directed graph.

as we outline in Chapter 6. One observation which can be made from a human perspective is that trust is weakened or diluted through transitivity. Revisiting the above example, we note that Alice's trust in the car mechanic Eric through the recommenders Bob and Claire can be at most as strong as Claire's trust in Eric. This is illustrated in Fig. 7.2.

By assuming Alice's trust in Bob and Bob's trust in Claire to be positive but not absolute, Alice's derived trust in Eric is intuitively weaker than Claire's trust in Eric.

Claire obviously recommends to Bob her opinion about Eric as a car mechanic, but Bob's recommendation to Alice is ambiguous. It can either be that Bob passes Claire's recommendation unaltered on to Alice, or that Bob derives indirect trust in Eric which he recommends to Alice. The latter way of passing recommendations can create problems, and it is better when Alice receives Claire's recommendation unaltered. This will be discussed in more detail in Section 7.5.

It could be argued that negative trust in a transitive chain can have the paradoxical effect of strengthening the derived trust. Take for example the case where Bob distrusts Claire and Claire distrusts Eric, whereas Alice trusts Bob. In this situation, Alice might actually derive positive trust in Eric, since she relies on Bob's advice and Bob says: "Claire is a cheater, do not rely on her". So the fact that Claire distrusts Eric might count as a pro-Eric argument from Alice's perspective. The question boils down to "is the enemy of my enemy my friend?", which relates to how multiple levels of distrust should be interpreted. This topic could be interesting to analyze in further research.

7.2 Parallel Trust Combination

It is common to collect advice from several sources in order to be better informed when making decisions. This can be modeled as *parallel trust combination* illustrated in Fig. 7.3.

Let us assume again that Alice needs to get her car serviced, and that she asks Bob to recommend a good car mechanic. When Bob replies that Claire, a good friend of his, recommended Eric to him, Alice would like to get a second opinion, so she asks David whether he has heard about Eric. David also knows and trusts Claire, and has heard from her that Eric is a good car mechanic. Alice who does not know Claire personally, is unable to obtain a first hand recommendation about the car mechanic Eric, i.e. she does not directly know anybody with functional trust in Eric. Intuitively, if both Bob

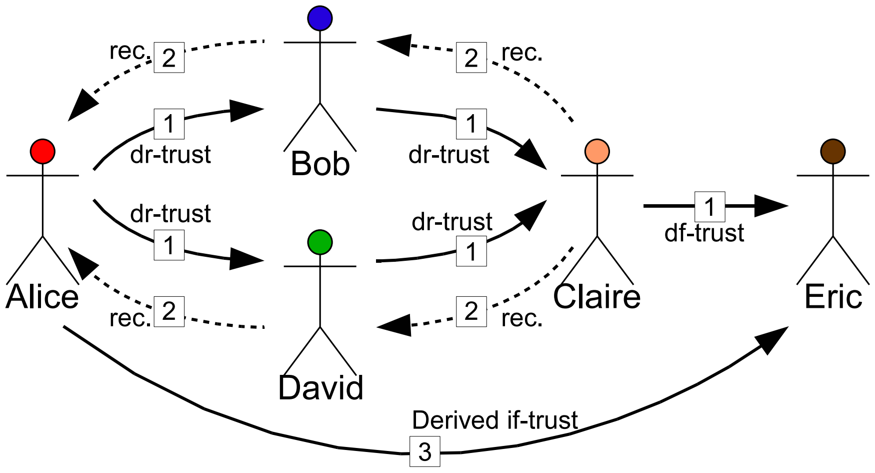


Figure 7.3: Trust derived by parallel combination of trust paths

and David recommend Claire as a good adviser regarding car mechanics, Alice's trust in Claire's advice will be stronger than if she had only asked Bob. Parallel combination of positive trust thus has the effect of strengthening the derived trust.

In the case where Alice receives conflicting recommended trust, e.g., trust and distrust at the same time, she needs some method for combining these conflicting recommendations in order to derive her trust in Eric. Our method, which is described in Section 7.6, is based on subjective logic which can easily handle such cases.

7.3 Structured Notation

In this section, we introduce a concise notation with which trust transitivity and parallel combination of trust paths can be expressed.

Transitive trust networks can involve many principals, and in the examples below, capital letters A, B, C, D, E and F are used to denote principals instead of names such as Alice and Bob. We use basic constructs of directed graphs to represent transitive trust networks and add some notation elements which allow us to express trust networks in a structured way.

A single trust relationship can be expressed as a directed arc between two nodes that represent the trust source and the trust target of that arc. For example the arc $[A, B]$ means that A trusts B.

The symbol ":" is used to denote the transitive connection of two consecutive trust arcs to form a transitive trust path. The trust relationships of Fig. 7.2 can be expressed as:

$$([A,E]) = ([A,B] : [B,C] : [C,E]) \tag{7.1}$$

where the trust scope is implicit. Let the trust scope e.g. be defined as σ: *"trusts X to be a good car mechanic"*. Let the functional variant be denoted by fσ and the referral variant by rσ. A distinction can be made between initial *direct trust* and derived *indirect trust*. Whenever relevant, the trust scope can be prefixed with "d" to indicate direct trust (dσ), and with "i" to indicate indirect trust (iσ). This can be combined with referral and functional trust, so that for example indirect functional trust can be denoted as ifσ. A reference to the trust scope can then be explicitly included in the trust arc notation as e.g. denoted by $[A,B,\mathrm{dr}\sigma]$. The trust network of Fig. 7.2 can then be explicitly expressed as:

$$([A,E,\mathrm{if}\sigma]) = ([A,B,\mathrm{dr}\sigma] : [B,C,\mathrm{dr}\sigma] : [C,E,\mathrm{df}\sigma]) \tag{7.2}$$

Let us now turn to the combination of parallel trust paths, as illustrated in Fig. 7.3. We will use the symbol "⋄" to denote the graph connector for this purpose. The "⋄" symbol visually resembles a simple graph of two parallel paths between a pair of agents, so that it is natural to use it in this context. Alice's combination of the two parallel trust paths from her to Eric in Fig. 7.3 can then be expressed as:

$$([A,E,\mathrm{if}\sigma]) = ((([A,B,\mathrm{dr}\sigma] : [B,C,\mathrm{dr}\sigma]) \diamond ([A,D,\mathrm{dr}\sigma] : [D,C,\mathrm{dr}\sigma])) : [C,E,\mathrm{df}\sigma]) \tag{7.3}$$

In short notation, the same trust graph can be expressed as:

$$([A,E]) = ((([A,B] : [B,C]) \diamond ([A,D] : [D,C])) : [C,E]) \tag{7.4}$$

It can be noted that Fig. 7.3 contains two paths. The graph consisting of the two separately expressed paths would be:

$$([A,E]) = ([A,B] : [B,C] : [C,E]) \diamond ([A,D] : [D,C] : [C,E]) \tag{7.5}$$

A problem with Eq. 7.5 is that the arc $[C,E]$ appears twice. Although Eq. 7.4 and Eq. 7.5 consists of the same two paths, their combined structures are different. Some computational models would be indifferent to Eq. 7.4 and Eq. 7.5, whereas others would produce different results depending on which expression is being used. When implementing the serial ":" as binary logic "AND", and the parallel "$\diamond$" as binary logic "OR", the results would be equal. However, when implementing “:” and “$\diamond$” as probabilistic multiplication and comultiplication respectively, the results would be different. It would also be different in the case of applying subjective logic operators for transitivity and parallel combination which we describe in Section 7.6. In general, it is therefore desirable to express graphs in a form where an arc only appears once. This will be called a *canonical expression*.

Definition 7 (Canonical Expression) *An expression of a trust graph in structured notation where every arc only appears once is called canonical.*

With this structured notation, arbitrarily large trust networks can be explicitly expressed in terms of source, target, and the additional properties depending on the utilized trust model, e.g. trust scope, measure, confidence etc..

A general directed trust graph is based on directed trust arcs between pairs of nodes. With no restrictions on the possible trust arcs, trust paths from a given source X to a given target Y can contain loops and dependencies, which could result in inconsistent calculative results. Dependencies in the trust graph must therefore be controlled when applying calculative methods to derive measures of trust between two parties. *Normalization* and *simplification* are two different approaches to dependency control.

7.4 Normalisation of Trust Measures

To allow a comparison with our model that is based on graph simplification, we briefly examine trust measure normalization in this section. The basic principle behind normalization is to retain the whole trust graph with its loops and dependencies, and normalize the computed trust measures in order to maintain consistency. Three examples are described in the following how this can be done: PageRank, EigenTrust and the original UniTEC algorithm for computing transitive trust. Other proposed models including [KSGM03, Lev04, ZL04] also allow looped and/or dependent transitive trust paths.

7.4.1 The PageRank Algorithm

The early web search engines such as Altavista simply presented every web page that matched the key words entered by the user, which often resulted in too many and irrelevant pages being listed in the search results. Altavista's proposal for handling this problem was to offer advanced ways to combine keywords based on binary logic. This was too complex for users, and therefore did not provide a good solution.

PageRank proposed by Page *et al.* (1998) [PBMW98] represents a way of ranking the best search results based on a page's reputation. Roughly speaking, PageRank ranks a page according to how many other pages are pointing at it. This can be described as a reputation system, because the collection of hyperlinks to a given page can be seen as public information that can be combined to derive a reputation score. A single hyperlink to a given web page can be seen as a positive rating of that web page. Google's search engine[4] is based on the PageRank algorithm, and the rapidly rising popularity of Google at the cost of Altavista was obviously caused by the superior search results that the PageRank algorithm delivered. The definition of PageRank from Page *et al.* (1998) [PBMW98] is given below:

Definition 8 (PageRank) *Let P be a set of hyperlinked web pages and let u and v denote web pages in P. Let $N^-(u)$ denote the set of web pages pointing to u and let $N^+(v)$ denote the set of web pages that v points to. Let E be some vector over P corresponding to a source of rank. Then, the PageRank of a web page u is:*

$$R(u) = cE(u) + c\sum_{v\in N^-(u)} \frac{R(v)}{|N^+(v)|}\,, \quad \textit{where } c \textit{ is chosen such that } \sum_{u\in P} R(u) = 1. \quad (7.6)$$

In [PBMW98] it is recommended that E be chosen such that $\sum_{u\in P} E(u) = 0.15$. The first term $cE(u)$ in Eq. 7.6 gives rank value based on initial rank. The second term $c\sum_{v\in N^-(u)} \frac{R(v)}{|N^+(v)|}$ gives rank value as a function of hyperlinks pointing at u.

[4]http://www.google.com/

According to Def. 8, $R \in [0,1]$. However, the PageRank values that Google provides to the public are scaled to the range [0,10] in increments of 0.25. We will denote the public PageRank of a page u as $PR(u)$. This public PageRank measure can be viewed for any web page using Google's toolbar which is a plug-in to the MS Internet Explorer. Although Google do not specify exactly how the public PageRank is computed, it is widely conjectured that it measured on a logarithmic scale with base close to 10. An approximate expression for computing the public PageRank could for example be:

$$PR(u) = l + \log_{10} R(u) \tag{7.7}$$

where l is a constant that defines the cut-off value for pages to have a PageRank so that only pages with $R(u) > 10^{-l}$ will be included. A typical value is $l = 11$.

It is not publicly known how the source rank vector E is defined, but it would be natural to distribute it over the root web pages of all domains weighted by the cost of buying each domain name. Assuming that the only way to improve a page's PageRank is to buy domain names, Clausen (2004) [Cla04] shows that there is a lower bound to the cost of obtaining an arbitrarily good PR.

Without specifying many details, Google states that the PageRank algorithm they are using also takes other elements into account, with the purpose of making it difficult or expensive to deliberately influence PageRank.

In order to provide a semantic interpretation of a PageRank value, a hyperlink can be seen as a positive rating of the page it points to. Negative ratings do not exist in PageRank so that it is impossible to blacklist web pages with the PageRank algorithm of Eq. 7.6 alone. Before Google with it's PageRank algorithm entered the search engine arena, some webmasters would promote web sites in a spam-like fashion by filling web pages with large amounts of commonly used search key words as invisible text or as metadata in order for the page to have a high probability of being picked up by a search engine no matter what the user searched for. Although this still can occur, PageRank seems to have reduced that problem because a high PR is also needed in addition to matching key words in order for a page to be presented to the user.

PageRank applies the principle of trust transitivity to the extreme because rank values can flow through looped and arbitrarily long hyperlink chains.

7.4.2 The EigenTrust Algorithm

The EigenTrust algorithm proposed by Kamvar *et al.* (2003) [KSGM03] is aimed at deriving global reputation scores in P2P communities with the purpose of assisting members in choosing the most reputable peers.

EigenTrust assumes that each peer i observes whether its interactions with a peer j have been positive or negative. The satisfaction score s_{ij} for peer j as seen by peer i is based on the number of satisfactory interactions $\mathrm{sat}(i,j)$ and the number of unsatisfactory interactions $\mathrm{unsat}(i,j)$, and is expressed as:

$$s_{ij} = \mathrm{sat}(i,j) - \mathrm{unsat}(i,j) \tag{7.8}$$

The *normalized* local trust score c_{ij} of peer j as seen by peer i is computed as:

$$c_{ij} = \frac{\max(s_{ij},0)}{\sum\limits_{l\in L} \max(s_{i,l},0)} \tag{7.9}$$

where L is the local set of peers with which peer i has had direct experiences. This step effectively normalizes the local trust values to the range [0,1] and thereby removes any negative trust values. A local peer with a large negative satisfaction score would thus have the same normalized local trust score as a local peer with satisfaction score 0.

In EigenTrust, trust scores of peers one hop outside peer i's local group, denoted by t_{ik}, can be computed from two connected trust arcs with the following formula:

$$t_{ik} = \sum_{j\in L} c_{ij} c_{jk} \tag{7.10}$$

This step effectively collapses functional trust and referral trust into a single trust type, and uses multiplication of normalized trust scores as the transitivity operator. While this allows for simple computation, it creates a potential vulnerability. A malicious peer can for example behave well during transactions in order to get high normalized trust scores as seen by his local peers, but can report its own local trust scores with false values (i.e. too high or too low). By combining good behaviour with reporting false local trust scores, a malicious agent can thus cause significant disturbances in global trust scores.

The computation of global trust scores takes place as follows. In the EigenTrust model, $C = [c_{ij}]$ represents the matrix of all normalized local trust values in the community, $\vec{c}_i$

represents the vector of peer i's local trust values, and $\vec{t_i}$ represents the vector containing the trust values t_{ik}, where peer i and peer k are separated by n intermediate nodes (loops included). Then $\vec{t_i}$ can expressed as:

$$\vec{t_i} = C^n \vec{c_i} \tag{7.11}$$

When n is large, the vector $\vec{t_i}$ will converge to the same vector for every peer i, which is the left principal eigenvector of C. The vector $\vec{t_i}$ is a global trust vector in the EigenTrust model, and quantifies the community's trust in peer k. To provide an interpretation of the trust measure computed by EigenTrust, it is useful to note a few properties.

- Trust measures are in the range [0,1].
- The sum of trust measures over the members of the community is not constant (as opposed to PageRank).
- Negative trust can not be expressed, only zero trust. This means that newcomers will have the same reputation as a longstanding member with the worst possible track record.
- Negative trust can not be propagated. This means that when a peer has had many negative experiences with another peer, it is as of no interaction has taken place.

7.4.3 The Original UniTEC Algorithm for Transitivity

We described in Subsection 3.3.3 how the UniTEC prototype receives recommendation responses to a posed request. Each recommendation response contains the TDI in addition to a trust chain.

To determine the resulting transitive trust T of the requester in the recommender from *a single trust chain* (with T_1 denoting the trust of the requester in identity I_1 and T_n denoting the trust of intermediate identity I_{n-1} in the recommender), our approach [KR03], published in the same month as Eigentrust [KSGM03], also uses the product of the individual trust values in the links of the chain:

$$T = \prod_{i=1}^{n} T_i \tag{7.12}$$

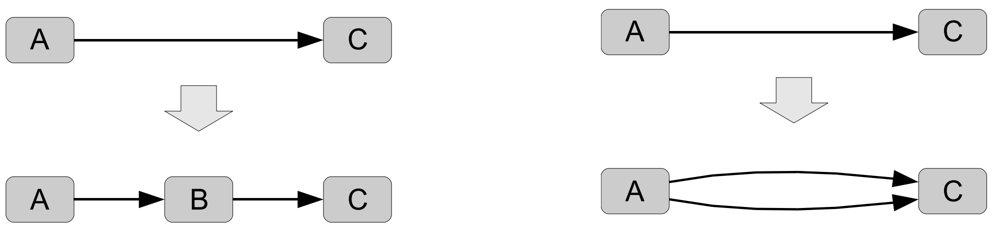
Furthermore, the algorithm follows an *optimistic approach* to combine *different trust chains*, more clearly, the strongest trust chain determines the resulting trust that is attached. In the case of direct trust of the requester in the recommender, this direct trust takes precedence over the strongest chain if the number of experiences with the recommender in the trust model is sufficiently high, that is if there is a sufficient amount of certainty in the direct trust.

Additionally, the trust chain evaluation algorithm is embedded in the request dissemination, as described in Fig. 3.3. Therefore it would fit better in the next section, since it does simplify the existing trust network through its optimistic approach. This is achieved by the decision of an intermediary to only forward a recommendation request, if either it has not been processed before, or if it has been processed before, but with *lower trust*. Thus, the intermediary will *not* forward a request that has been already processed with higher trust, which leads to the *strongest trust chain being available* at the requester, but *just a subset* of the available *weak trust chains*.

7.5 Network Simplification

This section describes our method for simplifying graphs by removing loops and dependencies from the graph between the source and the target parties, resulting in a directed series-parallel graph which eliminates the need for normalization. Firstly, we describe an algorithm for determining all possible paths from a given source to a given target, and secondly discuss how to select a subset of those paths for creating a DSPG. Finally, we explain why it is necessary to pass a trust recommendation as first hand direct trust from the recommender to the relying party, and not as indirect derived trust.

Trust network analysis based on network simplification is very different from methods based on normalization. Simplification of a trust network consists of only including certain arcs in order to allow the trust network between the source trustor and the target trustee to be formally expressed as a canonical expression. Graphs that represent this type of networks are known as *directed series-parallel graphs* (DSPG) [FL03]. A DSPG can be constructed by sequences of serial and parallel compositions that are defined as follows [FL03]:



(a) Directed series composition (b) Directed parallel composition

Figure 7.4: The principles of directed series and parallel composition

Definition 9 (Directed Series and Parallel Composition)

- *A* directed series *composition consists of replacing an arc* $[A,C]$ *with two arcs* $[A,B]$ *and* $[B,C]$ *where B is a new node.*
- *A* directed parallel *composition consists of replacing an arc* $[A,C]$ *with two arcs* $[A,C]_1$ *and* $[A,C]_2$.

The principle of directed series and parallel composition is illustrated in Fig. 7.4. The calculative analysis of a DSPG trust network does not require normalization, because a DSPG does not have loops and internal dependencies.

7.5.1 Finding Paths

Without normalization, a graph must be simplified in order to remove loops and dependencies. The first step is to determine the possible paths. The pseudo-code in Fig. 7.5 represents an algorithm for finding all directed paths between a given pair of source and target peers, where each individual path is loop free.

Transitive trust graphs can be stored and represented on a computer in the form of a list of directed trust arcs with additional attributes. Based on the list of arcs, an automated parser can establish valid DSPGs between two parties depending on the need. The initial direct trust arcs of Fig. 7.6 can for example be listed as in Table 7.1.

A parser based on the algorithm of Fig. 7.5 going through Table 7.1 will be able to determine the directed graph of Fig. 7.6 between A and E. The principal A can be called a relying party because she relies on the recommendations from B, D and C to derive her trust in E.

Pseudo-Constructor for a trust arc between two parties:

```
Arc(Node source, Node target, Scope scope, Variant variant){
    this.source = source;
    this.target = target;
    this.scope = scope;
    this.variant = variant;
}
```

Pseudo-code for a depth-first path finding algorithm:
After completion, ‘paths’ contains all possible paths between source and target.

```
void FindPaths(Node source, Node target, Scope scope) {
    SELECT arcs FROM graph WHERE (
        (arcs.source == source) AND
        (arcs.scope == scope))
    FOR EACH arc IN arcs DO {
        IF (
            (arc.target == target) AND
            (arc.variant == ‘functional’) AND
            (Confidence(path + arc) > Threshold)) {
            paths.add(path + arc);
        }
        ELSE IF (
            (arc.target != target) AND
            (arc.variant == ‘referral’) AND
            (arc NOT IN path) AND
            (Confidence(path + arc) > Threshold)) {
            path.add(arc);
            FindPaths(arc.target, target, scope)
        }
        path.remove(arc);
    }
}
```

Pseudo-code for method call:
The global variables ‘path’ and ‘paths’ are initialized.

```
Vector path = NEW Vector OF TYPE arc;
Vector paths = NEW Vector OF TYPE path;
FindPaths(StartSource, FinalTarget, scope);
```

Figure 7.5: Path finding algorithm

Source	Target	Scope	Variant
A	B	σ	referral
A	D	σ	referral
B	C	σ	referral
B	D	σ	referral
D	C	σ	referral
C	E	σ	functional

Table 7.1: Initial direct trust relationships of Fig. 7.6

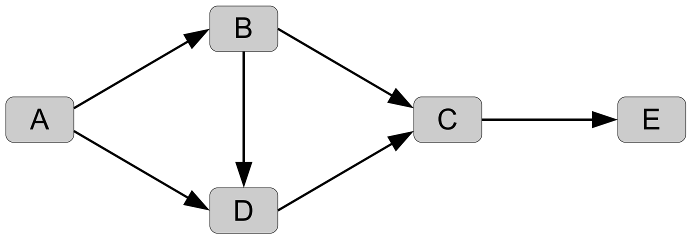


Figure 7.6: Trust network with dependent paths

7.5.2 Discovering the Optimal Directed Series-Parallel Graph

Ideally, all the possible paths discovered by the algorithm of Fig. 7.5 should be taken into account when deriving the trust value. A general directed graph will often contain loops and dependencies. This can be avoided by excluding certain paths, but this can also cause information loss. Specific selection criteria are needed in order to find the optimal subset of paths to include. With n possible paths, there are $2^n - 1$ different combinations for constructing graphs, of which not all necessarily are DSPGs. Of the graphs that are DSPGs, only one will be selected for deriving the trust measure.

Fig. 7.6 illustrates an example of a non-DSPG with dependent paths, where it is assumed that A is the source and E is the target. While there can be a large number of possible distinct paths, it is possible to use heuristic rules to discard paths, e.g. when their confidence drop below a certain threshold.

In the pseudocode of Fig. 7.5, the line “(Confidence(path + arc) > Threshold)” represents such a heuristic rule for simplifying the graph analysis. By removing paths with low confidence, the number of paths to consider is reduced while the information loss can be kept to an insignificant level.

In the graph of Fig. 7.6 there are 3 possible paths between A and E:

$$\begin{aligned}
\phi_1 &= ([A,B]:[B,C]:[C,E]) \\
\phi_2 &= ([A,D]:[D,C]:[C,E]) \\
\phi_3 &= ([A,B]:[B,D]:[D,C]:[C,E])
\end{aligned}$$

This leads to the following 7 potential combinations/graphs.

$$\begin{array}{lll}
\gamma_1 = \phi_1 & \gamma_4 = \phi_1 \diamond \phi_2 & \gamma_7 = \phi_1 \diamond \phi_2 \diamond \phi_3 \\
\gamma_2 = \phi_2 & \gamma_5 = \phi_1 \diamond \phi_3 & \\
\gamma_3 = \phi_3 & \gamma_6 = \phi_2 \diamond \phi_3 &
\end{array} \tag{7.13}$$

The expression γ_7 contains all possible paths between A and E. The problem with γ_7 is that it can not be represented in the form of a canonical expression, i.e. where an arc can only appear once. In this example, one path must be removed from the graph in order to have a canonical expression. The expressions γ_4, γ_5 and γ_6 can be canonicalised, and the expressions γ_1, γ_2 and γ_3 are already canonical, which means that all the expressions except γ_7 can be used as a basis for constructing a DSPG and for deriving A's trust in E.

The optimal DSPG is the one that results in the highest confidence level of the derived trust value. This principle focuses on maximizing certainty in the trust value, and not e.g. on deriving the strongest positive or negative trust value. The interpretation of confidence can of course have different meanings depending on the computational model. There is a trade-off between the time it takes to find the optimal DSPG, and how close to the optimal DSPG a simplified graph can can be. Below we describe an *exhaustive* method that is guaranteed to find the optimal DSPG, and a *heuristic* method that will find a DSPG close to, or equal to the optimal DSPG.

- **Exhaustive Discovery of Optimal Trust Graphs**
  The exhaustive method of finding the optimal DSPG consists of determining all possible DSPGs, then deriving the trust value for each one of them, and finally selecting the DSPG and the corresponding canonical expression that produces the trust value with the highest confidence level. As mentioned before, there are $(2^n - 1)$ different combinations for constructing graphs out of n possible paths. The computational complexity of this method is therefore $lm(2^n - 1)$, where m is the average number of paths in the DSPGs, and l is the average number of arcs in the paths.

- **Heuristic Discovery of Near-Optimal Trust Graphs**

  The heuristic method of finding a near-optimal DSPG consists of constructing the graph by including new paths one by one in decreasing order of confidence. Each new path that would turn the graph into a non-DSPG and break canonicity is excluded. This method only requires the computation of the trust value for a single DSPG and canonical expression, with computational complexity lm, where m is average number of paths in the DSPGs, and l is the average number of arcs in the paths.

The heuristic method will produce a DSPG with overall confidence level equal or close to that of the optimal DSPG. The reason why this method is not guaranteed to produce the optimal DSPG, is that it could exclude two or more paths with relatively low confidence levels because of conflict with a single path with high confidence level previously included, whereas the low confidence paths together could provide higher confidence than the previous high confidence path alone. In such cases it would have been optimal to exclude the single high confidence path, and instead include the low confidence paths. However, only the exhaustive method described above is guaranteed to find the optimal DSPG in such cases.

Figures 7.7, 7.8 and 7.9 illustrate how new paths can be included in a way that preserves graph canonicity. In the figures, the nesting level of nodes and arcs is indicated as an integer. A bifurcation is when a node has two or more incoming or outgoing arcs, and is indicated by brackets in the shaded node boxes. The opening bracket "(" increments the nesting level by 1, and the closing bracket ")" decrements the nesting level by 1. A sub-path is a section of a path without bifurcations. The equal sign "=" means that the node is part of a sub-path, in which case the nesting level of the arc on the side of the = sign is equal to the nesting level of the node. Each time a new path is added to the old graph, some sub-path sections may already exist in the old graph which does not require any additions, whereas other sub-path sections that do not already exist, must be added by bifurcations to the old graph.

The criteria needed for preserving a DSPG when adding new sub-paths are detailed in Def. 10. The source and target nodes refer to the source and target nodes of the new sub-path that is to be added to the old graph by bifurcation.

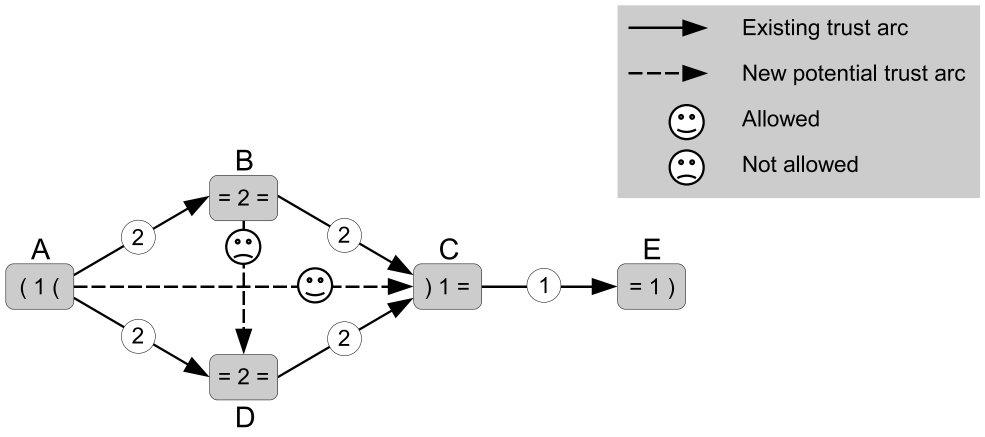


Figure 7.7: Visualization of the DSPG preservation requirement 1: Target node must be reachable from the source node.

Definition 10 (Criteria for Preserving a DSPG when Adding New Sub-Paths)

1. *The target node must be reachable from the source node in the old graph.*
2. *The source and the target nodes must have equal nesting levels in the old graph.*
3. *The nesting level of the source and target nodes must be equal to, or less than the nesting level of all intermediate nodes in the old graph.*

These principles are illustrated with examples below. Requirement 1 is illustrated in Fig. 7.7. The new arc $[B,D]$ is not allowed because D is not reachable from B in the old graph, whereas the new arc $[A,C]$ is allowed because C is reachable from A.

The new allowed arc can be included under the same nesting level as the sub-paths $([A,B]:[B:C])$ and $([A,D]:[D:C])$ in this example. The old and new graph of Fig. 7.7 are expressed below. Note that the brackets around sub-paths, e.g. $([A,B]:[B,C])$, are not reflected in Fig. 7.7 because they do not represent nesting, but simply grouping of arcs belonging to the same sub-path.

$$\begin{aligned} &\text{Old graph:} \quad ((([A,B]:[B,C])\diamond([A,D]:[D,C])):[C,E]) \\ &\text{New graph:} \quad ((([A,B]:[B,C])\diamond([A,D]:[D,C])\diamond[A,C]):[C,E]) \end{aligned} \tag{7.14}$$

Requirement 2 is illustrated in Fig. 7.8. The new arc $[B,D]$ is not allowed because B and D have different nesting levels, whereas the new arc $[A,D]$ is allowed because A and D have equal nesting levels.

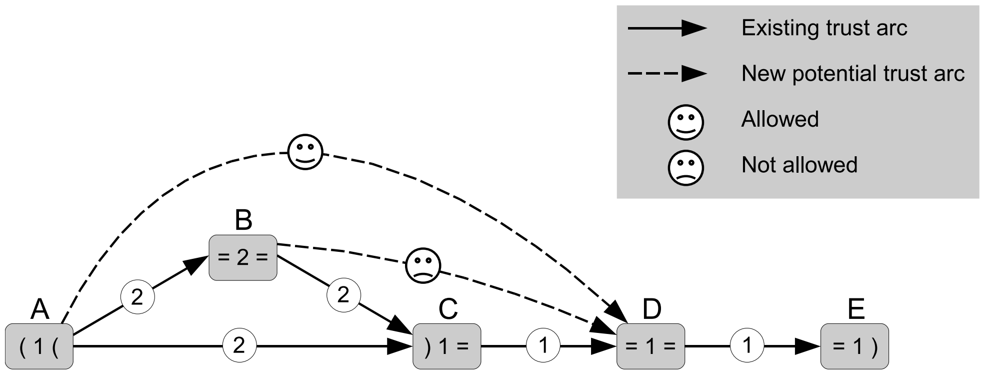


Figure 7.8: Visualization of the DSPG preservation requirement 2: Source and target nodes must have equal nesting levels.

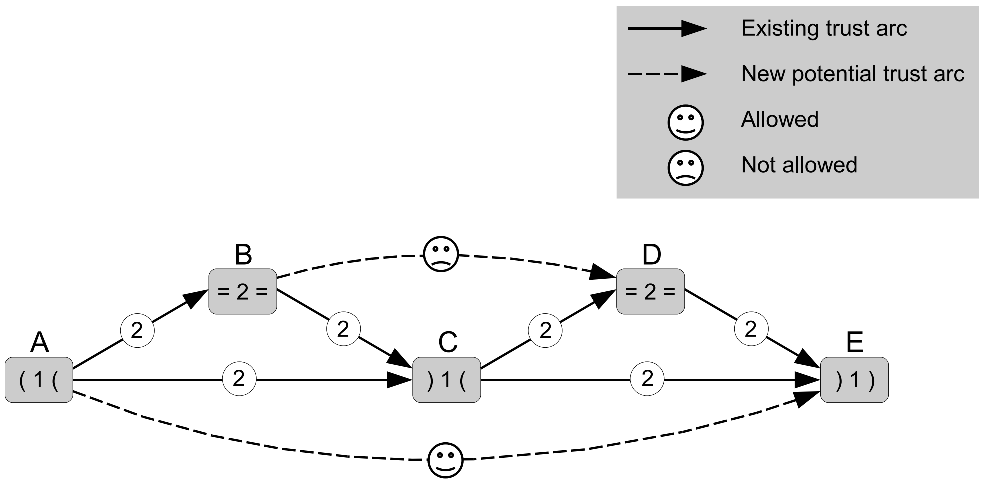


Figure 7.9: Visualization of the DSPG preservation requirement 3: Intermediate nodes can not have a nesting level less than the source and target nodes.

Adding the new allowed arc results in an additional nesting level being created, which also causes the nesting levels of the sub-paths $[A,B]:[B,C]$ and $[A,C]$ to increment. The old and new graph of Fig. 7.8 can then be expressed as:

$$\begin{aligned} &\text{Old graph:} \quad ((([A,B]:[B,C])\diamond[A,C]):[C,D]:[D,E]) \\ &\text{New graph:} \quad ((((([A,B]:[B,C])\diamond[A,C]):[C,D])\diamond[A,D]):[D,E]) \end{aligned} \tag{7.15}$$

Requirement 3 is illustrated in Fig. 7.9. The new arc $[B,D]$ is not allowed because the node C has a nesting level that is less that the nesting level of B and D, whereas the new arc $[A,E]$ is allowed because the nesting level of C is equal to that of A and E.

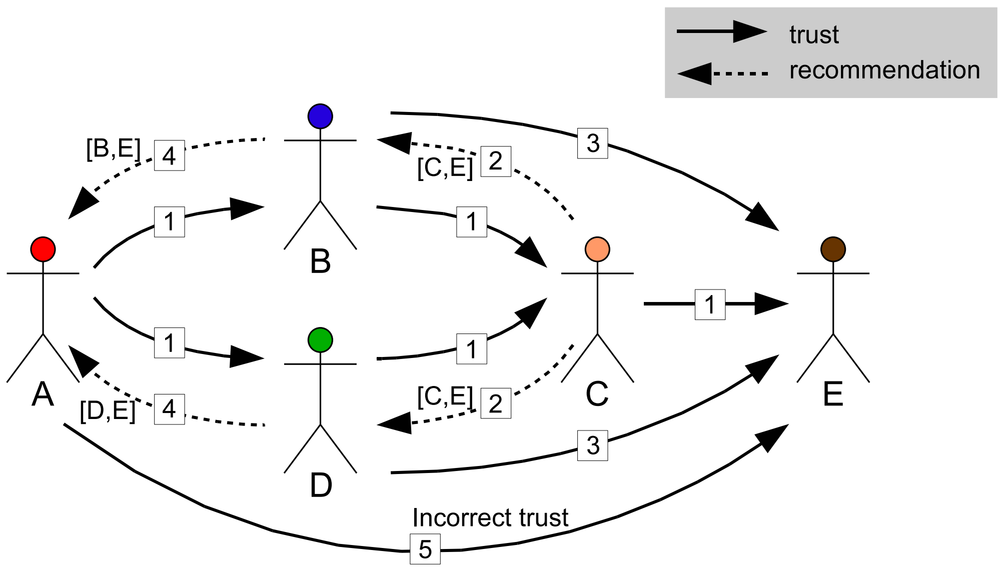


Figure 7.10: Incorrect way of passing recommendation leads to hidden trust expressions.

Adding the new allowed arc results in an additional nesting level being created, which also causes the nesting levels of the existing sub-paths to increment. The old and new graph of Fig. 7.9 can then be expressed as:

$$\text{Old graph:} \quad ((([A,B]:[B,C])\diamond[A,C]):(([C,D]:[D,E])\diamond[C,E]))$$

$$\text{New graph:} \quad (((([A,B]:[B,C])\diamond[A,C]):(([C,D]:[D,E])\diamond[C,E]))\diamond[A,E]) \tag{7.16}$$

7.5.3 First Hand Trust

Even if relying parties are aware of the need to base computations on canonical expressions, it is possible that the recommendations they receive prevent them from following that principle. Fig. 7.10 shows an example of how a certain type of recommendation can lead to non-canonical hidden trust expressions.

Here the trust and recommendation arrows are labeled in the order in which they are formed. In the scenario of Fig. 7.10, C passes her recommendation about E to B and D (index 2), so that B and D are able to derive indirect trust in E (index 3). Now B and D pass their derived indirect trust in E to A (index 4), so that she can derive indirect trust in E (index 5). The problem with this scenario is that A is ignorant about C so that A in fact analyzes a hidden graph with a non-canonical expression that is different from expression of the perceived graph:

$$\begin{array}{ccc} \text{Perceived graph:} & & \text{Hidden graph:} \\ (([A,B]:[B,E])\diamond([A,D]:[D,E])) & \neq & (([A,B]:[B,C]:[C,E])\diamond([A,D]:[D,C]:[C,E])) \end{array} \tag{7.17}$$

The reason for this is that when B and D recommend $[B,E]$ and $[D,E]$, they implicitly recommend $([B,C]:[C,E])$ and $([D,C]:[C,E])$, but A is not aware of this [Jøs99]. It can easily be seen that neither the perceived nor the hidden graph is equal to the real graph, which shows that this way of passing recommendations can lead to incorrect analysis.

We argue that B and D should pass the recommendations explicitly as $([B,C]:[C,E])$ and $([D,C]:[C,E])$ respectively so that the relying party A knows their origin. The recommenders B and D are normally able to do this, but then A also needs to be convinced that B and D have not altered the recommendation from C. If B and D are unreliable, they might for example try to change the recommended trust measures from C. Not only that, any party that is able to intercept the recommendations from B, D, or C to A might want to alter the trust values or any other parameters, and A therefore needs to receive evidence of the authenticity and integrity of the recommendations. An example of a correct way of passing recommendations is indicated in Fig. 7.11

In the scenario of Fig. 7.11, A receives all the recommendations directly, resulting in a perceived graph that is equal to the real graph that can be expressed as:

$$([A,E]) = ((([A,B]:[B,C])\diamond([A,D]:[D,C])):[C,E]) \tag{7.18}$$

The lesson to be learned from the scenarios in Fig. 7.10 and Fig. 7.11 is that there is a crucial difference between recommending trust in a principal, resulting from your own experience with that principal, and recommending trust in a principal which has been derived as a result of recommendations from others. As already mentioned, the term *direct trust* represents the former, and *indirect trust* the latter. Fig. 7.10 illustrated how problems can occur when indirect trust is recommended, so the rule is to only

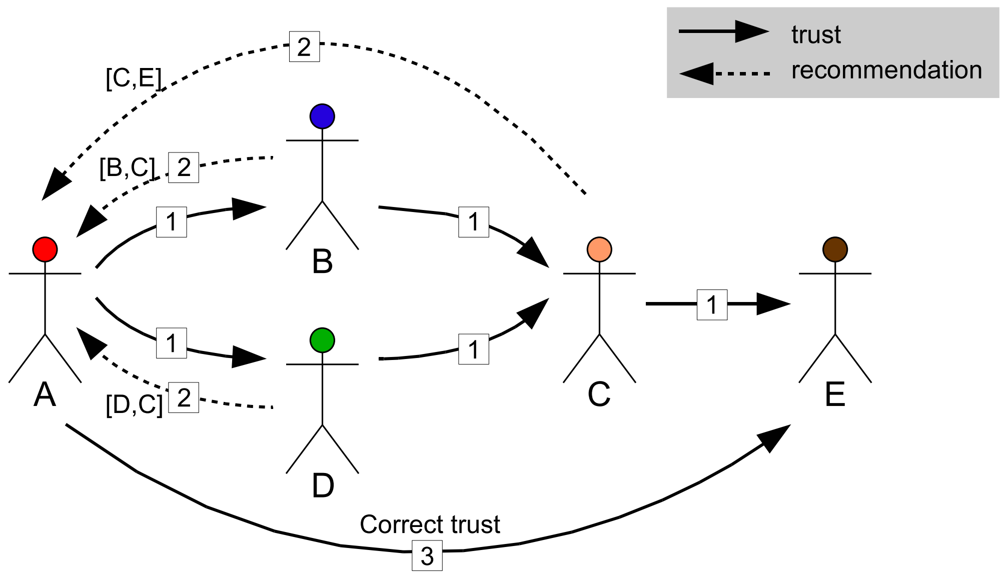


Figure 7.11: Correct way of passing recommendations

recommend direct trust [Jøs99]. For example, A's derived indirect trust in E in Fig. 7.11 should not be recommended to others.

7.6 Trust Network Analysis with Subjective Logic

We have presented mechanisms for network simplification which result in near optimal trust graphs. In order to derive trust measures from these graphs, calculative methods are needed for *serial combination of trust measures* along the transitive paths illustrated in Fig. 7.2 as well as for *fusion of trust measures* from the parallel paths illustrated in Fig. 7.3. *Subjective logic* represents a practical belief calculus that can be used for calculative analysis trust networks.

TNA-SL requires trust relationships to be expressed as beliefs, and trust networks to be expressed as DSPGs in the form of canonical expressions. In this section we describe how trust can be derived with the belief calculus of subjective logic, and subsequently give a numerical example.

7.6.1 Subjective Logic Fundamentals

Belief theory is a framework related to probability theory, but where the probabilities over the set of possible outcomes do not necessarily add up to 1, and the remaining probability is assigned to the union of possible outcomes. Belief calculus is suitable for approximate reasoning in situations of partial ignorance regarding the truth of a given proposition.

Subjective logic[Jøs01] represents a specific belief calculus that uses a belief metric called *opinion* to express beliefs. An opinion denoted by $\omega_x^A = (b,d,u,a)$ expresses the relying party A's belief in the truth of statement x. When a statement for example says *"Party X is honest and reliable regarding* σ*"*, then the opinion about the truth of that statement can be interpreted as trust in X within the scope of σ. Here b, d, and u represent belief, disbelief and uncertainty respectively where $b,d,u \in [0,1]$ and $b+d+u=1$. The parameter $a \in [0,1]$ is called the base rate, and is used for computing an opinion's probability expectation value that can be determined as $\mathrm{E}(\omega_x^A) = b + au$. More precisely, a determines how uncertainty shall contribute to the probability expectation value $\mathrm{E}(\omega_x^A)$. In the absence of any specific evidence about a given party, the base rate determines the *a priori* trust that would be put in any member of the community.

The opinion space can be mapped into the interior of an equal-sided triangle, where, for an opinion $\omega_x = (b_x, d_x, u_x, a_x)$, the three parameters b_x, d_x and u_x determine the position of the point in the triangle representing the opinion. Fig. 7.12 illustrates an example where the opinion about a proposition x from a binary state space has the value $\omega_x = (0.7,\ 0.1,\ 0.2,\ 0.5)$.

The top vertex of the triangle represents uncertainty, the bottom left vertex represents disbelief, and the bottom right vertex represents belief. The parameter b_x is the value of a linear function on the triangle, which takes value 0 on the edge which joins the uncertainty and disbelief vertices, and takes value 1 at the belief vertex. In other words, b_x is equal to the quotient when the perpendicular distance between the opinion point and the edge joining the uncertainty and disbelief vertices is divided by the perpendicular distance between the belief vertex and the same edge. The parameters d_x and u_x are determined similarly. The base of the triangle is called the probability axis. The base rate is indicated by a point on the probability axis, and the projector starting from the opinion point is parallel to the line that joins the uncertainty vertex and the base rate point on the probability axis. The point at which the projector meets the probability

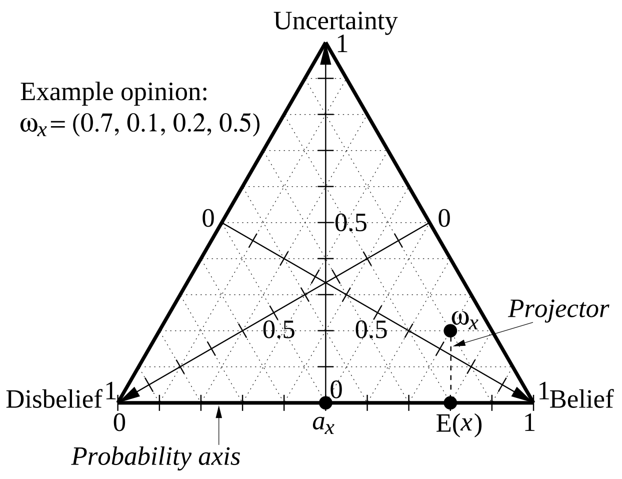


Figure 7.12: Subjective logic opinion triangle with example opinion ([JGK06])

axis determines the expectation value of the opinion, *i.e.* it coincides with the point corresponding to expectation value $E(\omega_x^A)$.

Opinions can be ordered according to probability expectation value, but additional criteria are needed in case of equal probability expectation values. We will use the following rules to determine the order of opinions[Jøs01]:

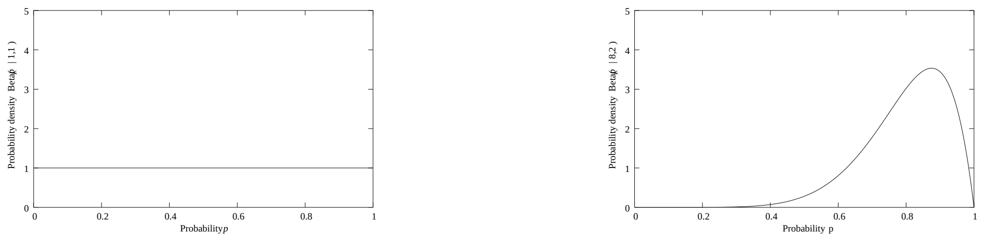
Let ω_x and ω_y be two opinions. They can be ordered according to the following rules by priority:

1. The opinion with the greater probability expectation is the greater opinion.
2. The opinion with the lower uncertainty is the greater opinion.
3. The opinion with the lower base rate is the greater opinion.

The probability density over binary event spaces can be expressed as beta PDFs (probability density functions) denoted by beta(α, β) [DS01]. Let r and s express the number of positive and negative past observations respectively, and let a express the *a priori* or base rate, then α and β can be determined as:

$$\alpha = r + 2a\,, \qquad \beta = s + 2(1-a)\,. \tag{7.19}$$

A bijective mapping between the opinion parameters and the beta PDF parameters can



(a) Uniform beta(1,1) (b) beta(8,2)

Figure 7.13: Example beta probability density functions ([JGK06])

be analytically derived [Jøs01, JP05] as:

$$\begin{cases} b_x = r/(r+s+2) \\ d_x = s/(r+s+2) \\ u_x = 2/(r+s+2) \\ a_x = \text{base rate of } x \end{cases} \iff \begin{cases} r & = 2b_x/u_x \\ s & = 2d_x/u_x \\ 1 & = b_x+d_x+u_x \\ a & = \text{base rate of } x \end{cases} \tag{7.20}$$

This means for example that a totally ignorant opinion with $u_x = 1$ and $a_x = 0.5$ is equivalent to the uniform PDF beta$(1,1)$ illustrated in Fig. 7.13.a. It also means that a dogmatic opinion with $u_x = 0$ is equivalent to a spike PDF with infinitesimal width and infinite height expressed by beta$(b_x\eta,\ d_x\eta)$, where $\eta \to \infty$. Dogmatic opinions can thus be interpreted as being based on an infinite amount of evidence.

After r positive and s negative observations in case of a binary state space (i.e. $a = 0.5$), the *a posteriori* distribution is the beta PDF with $\alpha = r+1$ and $\beta = s+1$. For example the beta PDF after observing 7 positive and 1 negative outcomes is illustrated in Fig. 7.13.b, which also is equivalent to the opinion illustrated in Fig. 7.12

A PDF of this type expresses the uncertain probability that a process will produce positive outcome during future observations. The probability expectation value of Fig. 7.13.b. is $\mathrm{E}(p) = 0.8$. This can be interpreted as saying that the relative frequency of a positive outcome in the future is somewhat uncertain, and that the most likely value is 0.8.

The variable p is a probability variable, so that for a given p the probability density beta$(p \mid \alpha, \beta)$ represents second order probability. The first-order variable p represents the probability of an event, whereas the density beta$(p \mid \alpha, \beta)$ represents the probability that the first-order variable has a specific value. Since the first-order variable p is continuous, the second-order probability beta$(p \mid \alpha, \beta)$ for any given value of $p \in [0,1]$

is vanishingly small and therefore meaningless as such. It is only meaningful to compute $\int_{p_1}^{p_2} \text{beta}(p \mid \alpha, \beta)$ for a given interval $[p_1, p_2]$, or simply to compute the expectation value of p. The expectation value of the PDF is always equal to the expectation value of the corresponding opinion. This provides a sound mathematical basis for combining opinions using Bayesian updating of beta PDFs.

7.6.2 Reasoning with Beliefs

Subjective logic defines a number of operators[Jøs01, PJ05, JPD05]. Some operators represent generalizations of binary logic and probability calculus operators, whereas others are unique to belief theory because they depend on belief ownership. Here we will only focus on the *discounting* and the *consensus* operators. The discounting operator can be used to derive trust from transitive paths, and the consensus operator can be used to derive trust from parallel paths. These operators are described below.

- **Discounting** is used to compute trust transitivity. Assume two agents A and B where A has referral trust in B, denoted by ω_B^A, for the purpose of judging the truth of proposition x. In addition B has functional trust in the truth of proposition x, denoted by ω_x^B. Agent A can then derive her trust in x by discounting B's trust in x with A's trust in B, denoted by $\omega_x^{A:B}$. By using the symbol '$\otimes$' to designate this operator, we define

$$\omega_x^{A:B} = \omega_B^A \otimes \omega_x^B \qquad \begin{cases} b_x^{A:B} = b_B^A b_x^B \\ d_x^{A:B} = b_B^A d_x^B \\ u_x^{A:B} = d_B^A + u_B^A + b_B^A u_x^B \\ a_x^{A:B} = a_x^B \; . \end{cases} \tag{7.21}$$

  The effect of discounting in a transitive chain is that uncertainty increases, not disbelief [Jøs02].

- **Consensus** is equivalent to Bayesian updating in statistics. The consensus of two possibly conflicting opinions is an opinion that reflects both opinions in a fair and equal way. Let ω_x^A and ω_x^B be A's and B's opinions about the same proposition x. The opinion $\omega_x^{A\diamond B}$ is then called the consensus between ω_x^A and ω_x^B, denoting an imaginary agent $[A,B]$'s opinion about x, as if she represented both A and B. By using the symbol '$\oplus$' to designate this operator, we define $\omega_x^{A\diamond B} = \omega_x^A \oplus \omega_x^B$.

$$\omega_x^{A\diamond B} = \omega_x^A \oplus \omega_x^B \qquad \begin{cases} b_x^{A\diamond B} = (b_x^A u_x^B + b_x^B u_x^A)/(u_x^A + u_x^B - u_x^A u_x^B) \\ d_x^{A\diamond B} = (d_x^A u_x^B + d_x^B u_x^A)/(u_x^A + u_x^B - u_x^A u_x^B) \\ u_x^{A\diamond B} = (u_x^A u_x^B)/(u_x^A + u_x^B - u_x^A u_x^B) \\ a_x^{A\diamond B} = a_x^A \end{cases} \tag{7.22}$$

where it is assumed that $a_x^A = a_x^B$. Limits can be computed [JDV03] for $u_x^A = u_x^B = 0$. The effect of the consensus operator is to amplify belief and disbelief and reduce uncertainty.

The discounting and consensus operators will be used for the purpose of deriving trust measures in the example below.

7.6.3 Example Derivation of Trust Measures

This numerical example is based the trust graph of Fig. 7.11. Table 7.2 specifies trust measures expressed as opinions. The DSTC Subjective Logic API[5] was used to compute the derived trust values.

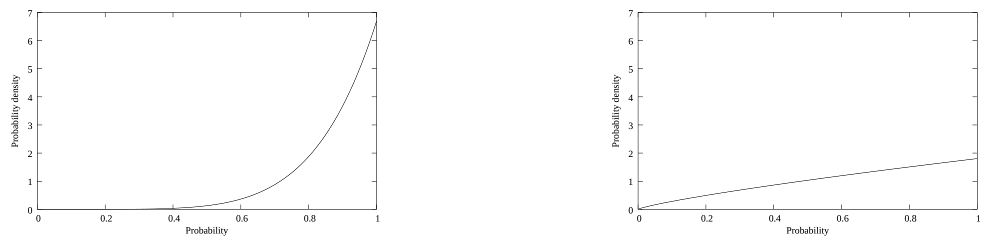
Table 7.2: Direct trust measures of Fig. 7.11

Source	Target	Variant	Measure	Time
A	B	r	$\omega_B^A = (0.9,\ 0.0,\ 0.1,\ 0.5)$	$\tau_B^A = 18.04.2006$
A	D	r	$\omega_D^A = (0.9,\ 0.0,\ 0.1,\ 0.5)$	$\tau_C^A = 18.04.2006$
B	C	r	$\omega_C^B = (0.9,\ 0.0,\ 0.1,\ 0.5)$	$\tau_D^B = 13.04.2006$
C	E	f	$\omega_E^C = (0.9,\ 0.0,\ 0.1,\ 0.5)$	$\tau_E^D = 13.04.2006$
D	C	r	$\omega_C^D = (0.3,\ 0.0,\ 0.7,\ 0.5)$	$\tau_D^C = 13.04.2006$
A	B	r	$\omega_B^{\prime A} = (0.0,\ 0.9,\ 0.1,\ 0.5)$	$\tau_B^{\prime A} = 19.04.2006$

By applying the discounting and consensus operators to the expression of Eq. 7.18, the derived indirect trust measure can be computed.

- Case a:

[5]Available at `http://security.dstc.com/spectrum/`



(a) $\omega_E^A \equiv \mathrm{beta}(6.7,\ 1.0)$ (b) $\omega_E^{\prime A} \equiv \mathrm{beta}(1.8,\ 1.0)$

Figure 7.14: Derived trust visualized as beta probability density functions ([JGK06])

First assume that A derives her trust in E on 18.04.2006, in which case the first entry for $[A,B]$ is used. The expression for the derived trust measure and the numerical result is given below.

$$\begin{aligned} \omega_E^A &= ((\omega_B^A \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.74,\ 0.00,\ 0.26,\ 0.50) \end{aligned} \tag{7.23}$$

- Case b:

Let us now assume that based on new experience on 19.04.2006, A's trust in B suddenly is reduced to that of the last entry for $[A,B]$ in Table 7.2. As a result of this, A needs to update her derived trust in E and computes:

$$\begin{aligned} \omega_E^{\prime A} &= ((\omega_B^{\prime A} \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.287,\ 0.000,\ 0.713,\ 0.500) \end{aligned} \tag{7.24}$$

The derived trust measures can be translated into beta PDFs according to Eq. 7.20 and visualized as density functions as illustrated by Fig. 7.14 below.

It can be seen that the trust illustrated in Fig. 7.14.a is relatively strong but that the trust in Fig. 7.14.b approaches the uniform distribution of Fig. 7.13.a and therefore is very uncertain. The interpretation of this is that the distrust introduced in the arc $[A,B]$ in case (b) has rendered the path $([A,B] : [B,C] : [C,E])$ useless. In other words, when A distrusts B, then whatever B recommends is completely discounted by A. It is as if B had not recommended anything at all. As a result A's derived trust in E must be based on the path $([A,D] : [D,C] : [C,E])$ which was already weak from the start.

7.7 Discussion

We have presented a notation for expressing trust networks, and a method for trust network analysis based on graph simplification and trust derivation with subjective logic. This approach is called Trust Network Analysis with Subjective Logic (TNA-SL).

Our approach is very different from trust network analysis based on normalization, as done in PageRank, EigenTrust, and the original UniTEC trust chain evaluation algorithm. The main advantage of normalization is that it can be applied to large highly connected random graphs without losing any trust information. The main disadvantages of normalization is that it makes trust measures relative, which prevents them from being interpreted in any absolute sense like e.g. statistical reliability. It is also very difficult to express and analyze negative trust in models based on normalization. Neither PageRank, EigenTrust nor the original UniTEC algorithm handle negative trust. In PageRank, negative trust cannot be expressed[6]. EigenTrust cuts off all negative trust during the normalization step. In UniTEC, the strongest trust chain takes precedence over all weaker trust chains due to its optimistic heuristic.

One advantage of TNA-SL is that negative trust can be explicitly expressed and propagated. In order for distrust to be propagated in a transitive fashion, all intermediate referral arcs must express positive trust, with only the last functional arc expressing negative trust. Using subjective logic allows trust measures to be interpreted in statistical terms, e.g. as measures of reliability. This also makes it possible to consistently derive trust measures from statistical data.

One of the drawbacks of TNA-SL is that a complex and cyclic network must be simplified before it can be analyzed, which can lead to loss of information. While the simplification of large highly connected networks could be slow, heuristic techniques can significantly reduce the computational effort. This is done by ignoring paths for which the confidence level drops below a certain threshold, and by including the paths with the strongest confidence level first when constructing a simplified network. This also leads to minimal loss of information.

The approach to analyzing transitive trust networks described here provides a practical method for expressing and deriving trust between peers/entities within a community or network. It could be incorporated in the UniTEC prototype but is also generally

[6]It is clear, that there is no point in taking into account all webpages *not* having references to a certain page.

applicable to a wide range of applications, such as monitoring the behavior of peers and assisting decision making in P2P communities, providing a quantitative measure of quality of web services, or possibly for determining the quality of foreign mobile devices in ad-hoc routing. Combined with subjective logic, TNA-SL allows trust measures to be efficiently analyzed and computed, and ultimately interpreted by humans and software agents.

Chapter 8

Trust Protection – Prevention of Sybil Attacks

Reputation systems suffer from malicious entities being able to create an arbitrary number of virtual identities and imbalance the system, a threat that is called Sybil attack. Recommendation systems suffer from easy copying of recommendations and recommenders attaching themselves to *trustworthy* recommenders to benefit from their good reputation. Electronic commerce in general and electronic payment systems in particular suffer from the uncertainty of potential customers about the reputation of online merchants and the quality of the offered goods or services. We address these issues in this chapter, whose main parts were previously published as a full paper [KR04] at IEEE CCNC 2004.

We structure this chapter as follows: In the next section, we introduce the general concept of Sybil attacks, outline the approaches offered by the related work, and highlight possible Sybil-related attacks on UniTEC. Our solution to counter these attacks is three-fold. Due to the fact that one major part of our solution combines the payment protocol SET with the reputation system UniTEC, we give a brief introduction on SET before going into the depths of our approach. We present the initial protocol and two variations and evaluate their distinct features. Although the protocol is described in conjunction with the SET payment scheme, it is easily applicable to other payment systems with the features outlined in this chapter.

8.1 Sybil Attacks

Employing digital pseudonyms as shown in Chapter 5 makes a system prone to the *Sybil attack*, first mentioned by Douceur in [Dou02]. Douceur points out that in a system, where virtual identities are created without centralized certification authorities, malicious entities could create an arbitrary number of virtual identities and thereby overwhelm the honest parties.

8.1.1 Related Work

The publication of John Douceur in 2002 raised the interest of several researchers[1] in the area of trust and reputation systems but also in social networks research and even general distributed systems.

In their survey paper [LSM06] on the Sybil attack, Levine et al. outline the general options that might be taken depending on the scenario in question to address this attack: *trusted certification* to control pseudonym creation, *resource testing* and *auditing* to identify Sybil nodes, *mobile networks* to track attackers via their network location, *trusted devices* and *recurring participation fees* respectively whole *cash economies* based on micropayments to increase the cost of an attack, and finally *reputation systems* which present inherent trust mechanisms to identify malicious participants.

Douceur himself suggested trusted certification as the only real protection against Sybil attacks. With the pseudonym creation under central control, attackers could not perform their identity multiplication undetected. However, this centralized scheme is not applicable to all scenarios.

Resource tests strive to determine whether a set of pseudonyms has actually lower resources available than what would be expected were they not linked. This includes checks on the IP address of the pseudonyms, as done in [FM02] by the developers of the Tarzan system and by Cornelli et al. in [CDdV$^+$02].

The approach SybilGuard by Yu et al. [YKGF06] relies on the analysis of social networks and the limited availability of real-world human trust relationships. Users in SybilGuard share symmetric keys via out of band mechanisms to set up the real-world trust. While attackers can share keys amongst themselves, the actual threat is posed

[1]The interested reader might notice, that our approach for addressing Sybil attacks [KR03, KP03, KR04] was published already from May 2003 until January 2004, before the hype.

by the links from attackers to the honest users in the network. SybilGuard works under the assumption that the number of these links is relatively small and incorporates mechanisms to bound the number of identities a malicious user can create.

Cheng and Friedman analyze in [CF05] the Sybil resilience of existing reputation systems and differ between what they call symmetric and asymmetric approaches. Asymmetric approaches follow the notion of trust being propagated along paths between the named nodes whereas symmetric approaches are invariant of the renaming of nodes and depend solely on the edge structure. To illustrate, Google's pagerank algorithm would be an example for a symmetric approach. Cheng and Friedman prove that symmetric algorithms in general are susceptible against Sybil attacks.

Seigneur et al. [SJ04, SGJ05] present a system that allows to transfer trust between pseudonyms via recommendations in a tightly controlled negotiation scheme. Although it may seem on first sight counter-intuitive that trust in a pseudonym is reduced when giving out a recommendation, the approach is not prone to Sybil attacks while still allowing recommendations as the negotiation mechanisms ensure that the total trust in the system does not increase.

Cvrček and Moody present in [CM05] a solution that focuses especially on the calculation of risk in addition to trust computation mechanisms. The risk analysis performs pattern mining on a huge set of data that is logged by the system during the transactions. System states (called contexts in their work) are identified in which positive and negative outcomes are recorded. By separating the risk from the trust assessment, the authors claim that the system can learn automatically from the evidence that has been gathered, and therefore can react on a Sybil attack by recognizing the similar behavior of attacking nodes.

The approach of David Ingram [Ing05] published in 2005 is vaguely related to ours. While our solution builds on combining payment systems with a recommendation system and thereby proving the authenticity of a recommendation, Ingram suggests to use existing companies like banks, the post office, phone companies, or Visa/Mastercard to act as a CA using blind signatures. In his threat analysis, he points out how the Entrapped platform is protected against Sybil attacked via the limitation of pseudonym generation.

8.1.2 Possible Sybil-based Attack on UniTEC

An attacker can create an arbitrary number of virtual identities via the IMC of its UniTEC agent. The possibilities of the attacker depend on how much effort he or she wants to invest to cause havoc.

If the attacker wants to start misbehaving immediately, he can rely on the knowledge/expertise advertisement mechanisms to become known. Obviously, the advertisement messages claim that the pseudonym has “lots” of expertise in many context areas. If one or more of the pseudonyms is queried for TDIs, the attacker may return wrongful recommendations to the requester in the hope of influencing the decision that the requester is about to perform based on the answers he got. The drawback from the attacker’s point of view is that the UniTEC mechanisms of the requester’s TMC calculate the trust in each recommendation. In this case, the attacker’s pseudonyms are not yet trusted, therefore the wrongful recommendations might have very little influence.

A more promising but also more time-consuming attack therefore consists of the attacker investing in the trustworthiness of the pseudonyms. In addition to advertising the knowledge areas as before, the attacker needs to copy TDI content from reputable recommenders and issue this information as new recommendations in the name of his or her own pseudonyms. Over time and with enough copied information, the UniTEC mechanisms will ensure that these pseudonyms will become trusted and eventually arrive in the level-0 neighborhood of requesters.

Once this has happened, the attacker has several options how to make use of this illegitimate trust: Firstly, as before, the attacker may issue wrongful recommendations, which now have a much higher impact since they appear as recommendations from trusted recommenders. Secondly, the attacker can defame other recommenders. This could be done by putting a trust statement with a very low trust in the trust chain before forwarding a query to further recommenders. Therefore, depending on the attacker’s influence and the used transitive trust algorithms, the attacker can keep recommendations from trustworthy recommenders from being heard by requesters.

Certainly, the feedback mechanisms ensure that the malicious pseudonyms are detected eventually once they “turned bad”. However by then, the damage is done already and the attacker might have any number of still trusted pseudonyms in store. We present in the following our approach that prevents these presented attacks from happening respectively increases the financial cost for these attacks such that an attacker would have to invest heavily before owning a large number of trusted pseudonyms.

8.1.3 Our Approach

As described before, UniTEC protects the strength of pseudonyms of both honest and malicious users and therefore could be considered as being prone to Sybil attacks. To counter this threat, we suggest a three-fold solution:

Firstly, as we presented in Chapter 4, Trusted Platform (TP) technology can be used for creating the pseudonyms via TP attestation identities. In the case of wrongful malicious recommendations causing financial loss, this allows the victims to find out the real identity of a pseudonym from its privacy-CA with sufficient legal proof being presented and depending on the CA policy. Therefore, the approach itself protects the anonymity of its users, but the identities may be revealed via out of band measures through legal entities.

Secondly, the trust update algorithms should take into account the transaction value, as we suggested in Chapter 6. This means that high-value transactions will lead to a stronger increase in trust as do low-value transactions. Building up a reputation is then a costly affair, and the reputation can be lost again quickly, for instance when performing badly in few high-value transactions. By binding monetary value to trust, cheating is still possible, however doing so becomes much more expensive, which is an incentive to behave properly in the system.

Thirdly, we present in the following a method, how to link reputation systems and payment systems (thereby securing solution two) with benefits for both. During the payment process for a certain item or service, an originality statement is created. This originality statement can be bound just once to a certain recommendation regarding the transaction in process. The employed process does protect the anonymity of the participants, while at the same time the recipient of a recommendation can check whether or not the transaction that the recommendation is about really took place. Therefore, in order to build trust, an entity really has to perform the financial transactions, which would be expensive for Sybil attackers.

8.2 Secure Electronic Transaction (SET) Background

The secure *electronic transaction protocol* (SET) is a payment protocol developed in a joint effort by Visa, Mastercard and several other companies in 1997. The *objectives* of this protocol are among others to provide confidentiality for payment and order in-

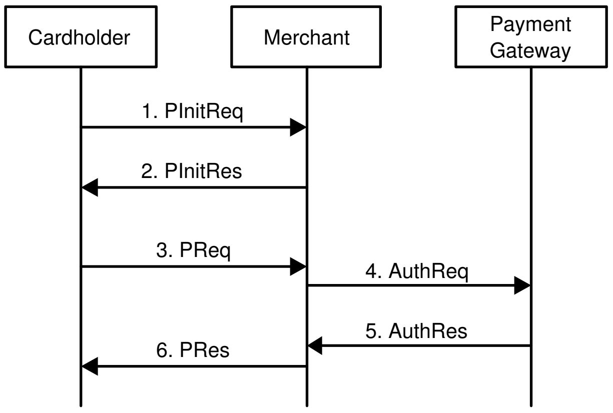


Figure 8.1: Typical SET message flow

formation, to ensure the integrity of all transmitted data and to provide cardholder- and merchant authentication. The *key players* are cardholder, merchant, payment gateway (acquirer) and certificate authorities. The cardholder is using a SET wallet software on his or her computer to invoke SET after having selected the goods in the merchant's online store. Figure 8.1 describes one typical SET message flow performed for a purchase transaction.

The optional *Initiate Request* and *Initiate Response* message pair (*PInitReq, PInitRes*) is used for the cardholder to obtain the payment gateway's certificate and certificate revocation list (CRL). The main purchase is initiated by the cardholder sending the *Purchase Request* (*PReq*) to the merchant. This mainly contains two distinct parts, namely the order information and payment information whereas the payment information is encrypted with the payment gateway's public key, hiding its content from the merchant. The merchant forwards this payment information in the *Authorization Request* message (*AuthReq*) to the payment gateway. In case of an authorized payment transaction, the payment gateway sends the confirmation to the merchant in *AuthRes* and the merchant sends the confirmation *PRes* to the cardholder so that fulfillment of the order can take place.

Brief Evaluation

According to SETCo[2] merchants can expect increased sales due to the increased confidence of the buyers in SET-compliant merchants and increased savings through a

[2] *http://www.setco.org*

reduction of exception handling and reduced cost associated with fraud. From the perspective of the cardholders, we see that SET offers increased protection of their privacy by keeping the payment information (credit card data) and order information separate from each other and only visible to the organization with a need-to-know.

On the downside, however, cardholders do have to install the wallet software and obtain the certificates which is somehow burdensome compared to alternative technologies like *Secure Sockets Layer* (SSL) and its successor *Transport Layer Security* (TLS) that are becoming more and more accepted due to seemingly "sufficient" security features and their integration in modern web-browsers. Merchants are relatively slow at adopting the standard due to its complexity and the involved cost.

The mechanisms we introduce in this paper are aimed at increasing the value and usefulness of SET for consumers and credit card companies alike. However, we stress the fact that our set of protocols can be applied to other payment systems as well, as long as a payment gateway is used that is available for direct communication with consumers and can perform basic cryptographic functions.

8.3 The Extended SET Protocol

In this section we describe how payment systems (respectively SET) and reputation systems (respectively UniTEC) can be combined by generating an *originality statement* in the payment process and integrating it in recommendation messages, thereby linking a recommendation to a real purchase.

The SET protocol may terminate with the merchant sending the *PRes* message to the cardholder as described in Section 8.2 and in much more detail in [VM97]. The entries in this message correspond to one or more current credit card transactions identified by their transaction identities *TransID*. Each transaction is associated with a completion code and data that further explains the code. In case of a successful credit card transaction, the message component *AuthStatus* contains the *AuthCode approved* and *AuthRatio* equals 1 as it refers to the ratio of authorized amount to required amount of the transaction. Fig. 8.2 illustrates the simplified content of this message.

After the order has been fulfilled and the cardholder has made experiences with the product or service that the transaction was about, she or he initiates the protocol extension as can be seen in Fig. 8.3 by forming the *Recommendation Signature Request* message (*RecSigReq*) and sending it directly to the payment gateway. The payment

PRes:

```
[ [ TransID ]+,
  [ CompletionCode,
    AcqCardMsg,
    AuthStatus,
    CapStatus,
    CreditStatusSeq ]+ ]
```

AuthStatus:

```
[ AuthDate,
  AuthCode,
  AuthRatio  ]
```

Figure 8.2: Simplified content of the SET PRes message

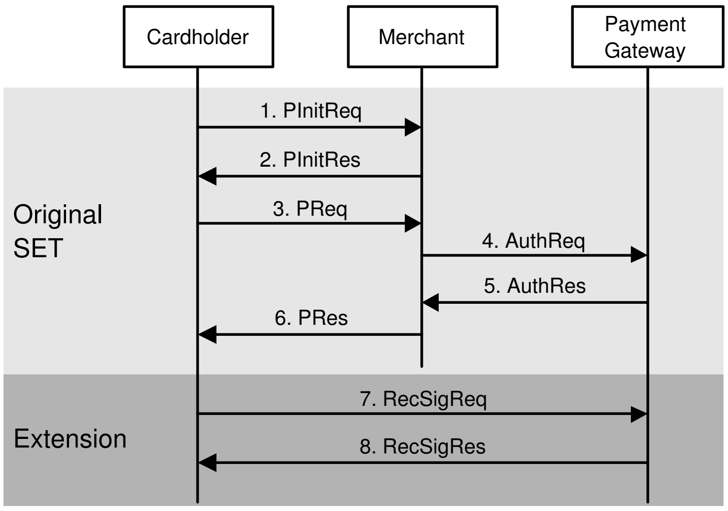


Figure 8.3: The extended SET protocol

gateway processes the request and answers with a *Recommendation Signature Response* message (*RecSigRes*) that contains an *Originality Statement* (*OStat*) that the cardholder can include in his or her recommendation to prove its originality.

We acknowledge the fact that the information communicated in *RecSigReq* and *RecSigRes* could be included in the original SET messages (Messages 3 to 6 in Fig. 8.3) as well. Since the recommendation has not been formed yet at that point in time, we can use an encrypted identifier instead of the recommendation hash to be signed by the payment gateway.

8.3.1 Protocol Messages

In the following, we present the content of the two protocol messages *Recommendation Signature Request* (RecSigReq) and *Recommendation Signature Response* (RecSigRes).

Recommendation Signature Request: RecSigReq

In order to understand the structure of this message it is important to keep in mind the content of UniTEC recommendations introduced in Subsection 3.2.4. They are digitally signed and contain the following components:

- *Recommendation Identifier: RID*
- *Target Identity: TID*
- *Rating: RData*
- *Recommender Certificate (pseudonymous and self-signed): RCert*

The notation used for describing the structure of the protocol messages can be found in Appendix A.

After the cardholder has formed the recommendation concerning the purchased product or service the *RecSigReq* message is created with the following structure:

$$\begin{aligned}
RecSigReq = \{\, & Enc(\, PuK_{PaymentGateway} \\
& \qquad K), \\
& Enc(\, K, \\
& \qquad Sign(\, PrK_{Cardholder}, \\
& \qquad\qquad \{\, TransID, \\
& \qquad\qquad\; Enc(\, PrK_{Pseudonym}, \\
& \qquad\qquad\qquad Hash(\{RID, TID, RData\})\,)\,\}\,)\,)\,\}
\end{aligned} \tag{8.1}$$

Firstly, a cryptographic hash function is computed over several important parts of the recommendation: the recommendation identifier RID, the target identity *TID* and the rating *RData* itself. The hash is then encrypted with the private key of the pseudonym that the cardholder intends to use for the recommendation in question. Obviously, this signed hash could only have been created by the recommender since only he has access to the mentioned private key. The field *TransID* from the *PRes* message is added to allow the payment gateway to process the request and to link this extension to the preceding credit card transaction.

This combination is signed with the signature key of the cardholder for the payment gateway to authorize this transaction. To protect the link between *TransID* and the *encrypted hash* from eavesdroppers, this part of the message has to be encrypted. For efficiency reasons we put it in an digital envelope instead of encrypting it with an asymmetric algorithm and the payment gateway's public key. If this part was not encrypted, it would be easier for an attacker to break the link between the cardholder's real identity and his pseudonym as will be seen later. This completed message is sent to the payment gateway.

Recommendation Signature Response: RecSigRes

Upon receipt of the *RecSigReq* message, the payment gateway retrieves the symmetric key by decrypting it with its private key. The symmetric key is used to gain access to the signed statement. In case of an invalid cardholder signature the request is discarded.

If the signature is correct the payment gateway checks whether the included transaction identifier fits to a transaction that this cardholder has performed, whether this transaction has been performed successfully and whether no previous *RecSigRes* message with a different encrypted hash has been sent to the cardholder for this transaction. This ensures that a single *OStat* is created for a single recommendation corresponding to one real transaction if and only if this transaction really took place. If these tests are successful, the *RecSigRes* message is formed:

$$\begin{aligned}
RecSigRes = \{\ & TransID, \\
& Enc(\ K, \\
& \quad Sign(\ PrK_{PaymentGateway}, \\
& \quad\quad \{\ Enc(\ PrK_{Pseudonym}, \\
& \quad\quad\quad Hash(\{RID, TID, RData\})\ ), \\
& \quad\quad PGCert\}\ )\ )\ \}
\end{aligned} \tag{8.2}$$

The encrypted hash that has been received in *RecSigReq* and the digital certificate of the payment gateway *PGCert* are signed with the private key of the payment gateway. We will refer to this signed item as *originality statement OStat*. In order to protect the link between the real identity and the pseudonym of the cardholder *OStat* is encrypted with the symmetric key used in the previous message. To enable the cardholder to link request to response, the *TransID* is included and the message sent.

8.3.2 Integration of the Originality Statement OStat in Recommendations

Upon receipt of the *RecSigRes* message the cardholder takes the symmetric key corresponding to *TransID* to decrypt *OStat*. The digital signature on *OStat* is checked and in case of a correct payment gateway ought to be correct. The cardholder can now insert the originality statement in the recommendation and publish it via the mechanisms offered by the used reputation system, e.g. UniTEC:

$$\begin{aligned} Recommendation = Sign(\ & PrK_{Pseudonym}, \\ & \{RID, TID, RData, RCert, OStat\}\,) \end{aligned} \tag{8.3}$$

Requesters receiving recommendations including an originality statement will perform several tests that all have to succeed in order to accept the recommendation as valid.

First, the validity of the recommender's signature on the recommendation is checked. If the signature is valid, the payment gateway's signature on *OStat* is verified by using the included certificate (which should be a trusted SET certificate). In case this signature is valid as well, it is certain that *OStat* originated from the payment gateway and a transaction really took place. The *RID*, *TID* and *RData* are hashed. The encrypted hash contained in *OStat* is decrypted with the key contained in *RCert*. If both hashes match, this serves as proof that the recommendation is linked to a real transaction performed at the payment gateway.

8.3.3 Evaluation

From a reputation system's point of view, the most important gain is that a recommendation can only be created if a real transaction concerning the recommendation target took place. This also means that identity switching is hindered. A pseudonymous identity will become more valuable, since it is not possible to simply take over the recommendations to a newly created identity. Since copying recommendations is no longer possible without indeed having bought the product or service that the recommendation is about, it is harder respectively more expensive to attach oneself to a well reputable recommender and gain a good reputation by copying the recommendations from this expert. Obviously, even a valid recommendation is not necessarily trusted. The question of whether or not to trust the recommendation and its recommender depends on the trust mechanisms in the used reputation system.

For the financial institutions that are operating the payment gateways, one possible gain from such a combination is the possibility to offer their customers a better service. This is a differentiator from other payment gateway providers that is not to be underestimated. Furthermore, the participation in a reputation service is a motivation for all participants in payment transactions to behave properly.

On the downside, there is a certain privacy loss through the possibility for the payment gateway to learn the link between the real identity and the pseudonym. If it stores the whole *OStat* instead of just marking completed transactions (with sent *RecSigRes*) and then receives recommendations, it can follow the link from the encrypted hash in *OStat* in the recommendation to the encrypted hash received through the *RecSigReq* messages (signed with the real-identity SET cardholder certificate) and find out the link. However, it is quite likely that protection of this data is covered by the current banking confidentiality legislation already and besides some measure of trust in those institutions that handle our bank accounts might be in order. For those readers that are not as trusting, we will address this privacy loss in variation 2 of our protocol below.

8.4 Variation 1: Include Transaction Value

We will now present a minor variation of the protocol presented in Section 8.3 in order to solve a common problem of reputation systems that suffer from malicious entities building a good reputation with *low-value transactions* and consequently use this reputation for *dishonest high-value transactions* until they are discovered.

The *RecSigReq* message stays the same as before. However, instead of the payment gateway including only the encrypted recommendation hash from *RecSigReq* in the originality statement, it inserts the *transaction value TValue* as well. This is known from the corresponding SET transaction. The new *RecSigRes* message looks as follows:

$$\begin{aligned}
RecSigRes = \{\, & TransID, \\
& Enc(\, K, \\
& \quad Sign(\, PrK_{PaymentGateway}, \\
& \qquad \{\, Enc(\, PrK_{Pseudonym}, \\
& \qquad\quad Hash(\{RID, TID, RData\})\,), \\
& \qquad TValue, \\
& \qquad PGCert\}\,)\,)\,\}
\end{aligned} \tag{8.4}$$

The new *OStat* is defined as the payment gateway-signed component and this time includes the transaction value. As before, the cardholder inserts *OStat* in the recommendation to be published by UniTEC.

On the recommendation requester's side, the same tests for signatures and hashes are performed as already described with the basic variant. This time however, the requester is able to weigh the impact of this recommendation against other received recommendations with the transaction value if he or she wishes to do so. In addition to that, the update of trust of the requester in the recommenders, which is performed after own experiences have been made and the quality of recommendations can be judged, can be weighted with the transaction value as well.

Evaluation of Variation 1

In addition to the points raised in the evaluation of the basic protocol, we gain the ability to weight recommendations depending on the values of the corresponding transactions. This solves up to a certain degree the problems that modern reputation systems such as the one at EBay[3] and other online auction sites face with malicious sellers that first build up a reputation by performing lots of successful but very low-value transactions and then start causing havoc with few (until they are discovered) high-value transactions with missing fulfillment. Building up a good reputation with this weighted scheme should be too expensive to risk losing the good reputation again by showing malicious behavior.

There is more information provided that could in theory be used for profile building e.g. by linking the transactions of one pseudonym to construct a financial profile. However, through the use of pseudonyms, the danger of detailed profile building is still kept at bay and the additional value of the provided data for the participants outweighs (in the author's view) the slightly increased privacy concerns.

[3]*http://www.ebay.com*

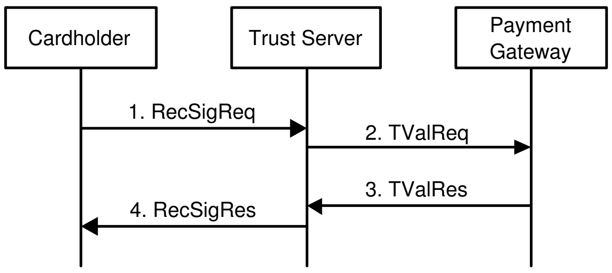


Figure 8.4: Protocol to receive the originality statement

8.5 Variation 2: Credit Card Companies as Trust Enabler

In this variation, we address the privacy concerns raised in both aforementioned protocols. Instead of making both recommendation and payment information available to the payment gateway we divide those responsibilities between a "trust server" operated by a credit card company and the payment gateway. The trust server is responsible only for the reputation information part whereas the payment gateway processes the payment data.

Instead of sending the recommendation signature request message *RecSigReq* to the payment gateway, the cardholder sends it to the trust server and receives a *RecSigRes* message from this server. Since the trust server provider is not directly involved in the credit card transaction performed between the cardholder and the payment gateway, information from the corresponding SET messages is needed to authorize the signature request.

The purchase amount is not included in any signed message *received* by the cardholder during the SET transaction. Thus the cardholder cannot prove the correctness of a certain transaction value to the trust server. If the transaction value – as presented in Section 8.4 – is to be included in *OStat*, it is necessary to introduce a query-response message pair *Transaction Value Request* (*TValReq*) and *Transaction Value Response* (*TValRes*) as can be seen in Fig. 8.4 between the trust server and the payment gateway, which ensures that the transaction value that the cardholder mentioned to the trust server is correct.

8.5.1 Protocol Messages

In this subsection we present the content and processing of the four protocol messages *Recommendation Signature Request* (RecSigReq), *Transaction Value Request* (TValReq), *Transaction Value Response* (TValRes), and *Recommendation Signature Response* (RecSigRes).

Recommendation Signature Request: RecSigReq

The *RecSigReq* message is extended to contain an authorization for the trust server to request the transaction value from the payment gateway. This information however could be used to link the real and the pseudonymous identity of the cardholder and is hidden from the trust server by encrypting it with the payment gateway's public key. A new transaction identifier *TransID2* is created and inserted instead of the real *TransID*. The encrypted hash is later used to confirm to recommendation requesters that the underwriter of the recommendation is indeed the one who performed the mentioned transaction.

$$\begin{aligned}
RecSigReq = \{ & Enc(\, PuK_{TrustServer}, \\
& \quad K), \\
& Enc(\, K, \\
& \quad Sign(\, PrK_{Pseudonym}, \\
& \qquad \{ Enc(\, PuK_{PaymentGateway}, \\
& \qquad\quad Sign(\, PrK_{Cardholder}, \\
& \qquad\qquad \{ TransID, \\
& \qquad\qquad\; TValue\})), \\
& \qquad Enc(\, PrK_{Pseudonym}, \\
& \qquad\quad Hash(\{RID, TID, RData\})), \\
& \qquad TransID2, \\
& \qquad TValue, \\
& \qquad PGCert\}))\}
\end{aligned} \tag{8.5}$$

Transaction Value Request: TValReq

Upon receipt of the *RecSigReq* message, the signed part is taken out of the digital envelope and the signature is checked. If this check is successful, the component that is encrypted with the payment gateway's public key is copied to the *TValReq* message. The trust server creates a third transaction identifier *TransID3* which is added to the message that is then signed and sent.

$$\begin{aligned} TValReq = Sign(\ & PrK_{TrustServer}, \\ & \{\ Enc(\ PuK_{PaymentGateway}, \\ & \qquad Sign(\ PrK_{Cardholder}, \\ & \qquad\qquad \{\ TransID, \\ & \qquad\qquad\ \ TValue\}\ )\ ), \\ & TransID3\}\ ) \end{aligned} \tag{8.6}$$

Transaction Value Response: TValRes

After successfully checking the signature of the *TValReq* message, the payment gateway decrypts the authorization information with its private key and checks the cardholder's signature and whether that cardholder has indeed successfully performed the SET transaction with the stated transaction identifier and value. In case of successful tests, the *TValRes* message is built.

$$\begin{aligned} TValRes = Sign(\ & PrK_{PaymentGateway}, \\ & \{TransID3, TValue\}\ ) \end{aligned} \tag{8.7}$$

Recommendation Signature Response: RecSigRes

After having received *TValRes* with a valid payment gateway signature and matching transaction value, the trust server builds the originality statement *OStat* by signing the encrypted hash, the transaction value and the trust server's digital certificate *TSCert* with its private key.

$$\begin{aligned} RecSigRes = \{\ & TransID2, \\ & Sign(\ PrK_{TrustServer}, \\ & \qquad \{\ Enc(\ PrK_{Pseudonym}, \\ & \qquad\qquad Hash(\{RID, TID, RData\})\ ), \\ & \qquad\ \ TValue, \\ & \qquad\ \ TSCert\}\ )\ )\} \end{aligned} \tag{8.8}$$

Upon receipt of the *RecSigRes* message the cardholder checks the trust server's digital signature and can now insert *OStat* into its recommendation as shown before.

8.5.2 Evaluation of Variation 2

As opposed to variation 1, we have gained improved privacy protection by strictly separating the SET information (with the real cardholder identity and payment data) and the reputation system information (with the recommendation and the pseudonymous identity). This obviously depends on the players keeping to their role and sticking to the protocol as it is. In case of the payment gateway and the trust server working together, there is no way of keeping the link of pseudonym to real identity private. Besides the increased privacy protection, we retain the benefits mentioned in the evaluation of the basic version and variation 1 as well.

8.6 Summary

In this section, we described the threats to reputation systems posed by Sybil attacks and in particular outlined possible Sybil-based attacks on UniTEC. We presented a three-fold approach to ward off these attacks: Firstly, trusted platform technology can be used to create TP attestation identities at a privacy CA instead of creating pseudonyms locally at the UniTEC agent. Secondly, trust update takes into account the utility represented by the transaction value. Thirdly, combining UniTEC with a payment system proves the originality of a recommendation and secures the attached transaction value.

The combination benefits both worlds, recommendation systems and payment systems alike. UniTEC gains especially an improved quality of the recommendation through its linkage to a real transaction via the originality statement. Furthermore, the transaction value signed by a payment gateway increases the recommendation's quality further and protects against attacks via copied/forged recommendations on the trust update process. Payment system providers benefit from increased confidence of their customers in purchasing online and reduced fraud due to identified untrustworthy merchants.

We designed this approach with a special focus on the protection of the participant's privacy. This was achieved through the usage of pseudonyms, as required by UniTEC, and by limiting the information available at each party to entities with a need-to-know.

Chapter 9

Conclusion

9.1 Summary

In this dissertation, we presented several key concepts required for a privacy-aware distributed reputation system that supports detailed trust modeling while protecting the user profiles from being attributable to a real life entity.

We pointed out how *trusted computing* technology and in particular TPM-protected storage, trusted attestation and integrity checking mechanisms can be used to enhance the security of our reputation system in a cost-effective and flexible manner. Despite the criticisms on trusted computing, we see many benefits of a secure *trust foundation* rooted in hardware that guards the platform against secret manipulations, which in case of a reputation system could not only cause damage to the owner's reputation but also cause financial loss to people relying on the owner's opinion.

We introduced the novel concept of *Extended Destination Routing* (EDR) as a general approach for enabling communication between pseudonyms while at the same time preserving unlinkability between sender and recipient as well as strong sender and recipient anonymity. The security evaluation shows that our approach indeed offers the required anonymity properties. Additionally, we achieve fault tolerance through a completely decentralized design. The performance evaluation of the prototype's implementation shows that the communication performance is sufficient for the reputation system scenario and can still be improved with optimizations to address the needs of further application scenarios. Although the UniTEC system as a whole provides many additional functions, the anonymous peer-to-peer communication component and the data management component work independently of the other components and can be applied to other reputation systems and even other application areas as well.

In addition to privacy protection techniques for reputation systems, we investigated the various dimensions of trust relationships. Based on that we presented our approach for a *generic trust model*, which represents these dimensions and includes measures for trust, certainty, and experiences required for trust calculations. The model is based on insights from social science research and observations gained through the analysis of a set of well-known trust models from the literature. It is generic in that it allows to plug in different specialized models and trust update algorithms and provides a bijective mapping between each local model and the generic trust representation. We discussed the adaptations of the original models that became necessary because we introduced new trust relationship dimensions and ones that are not supported as such by all algorithms.

This generic trust model provides the possibility to compare various *trust update algorithms*. We developed a set of test scenarios to assess the subjective quality of each supported algorithm. Our evaluation points out several important qualities – but also deficiencies – of the algorithms. To summarize our findings, we conclude that the Abdul-Rahman–Hailes algorithm in our generic trust model suffers from its discrete four step metrics in comparison to the field. In our view, the Beta Reputation system with an aging factor provides the best overall results. The only drawback is the limitation of the trust value bandwidth, which is proportional to the aging factor. The ReGreT algorithm provides responses to our test scenarios that meet our expectations, but its dynamics proved to be highly dependent on the history size: Highly variable trust values for a small history of experiences, and slower dynamics as more experiences were collected. Finally, the original UniTEC proposal provided a simple yet efficient algorithm and eased integration of the various dynamics. However, a deficiency of this algorithm lies in focusing merely on the current trust value and the latest experience and not taking into account patterns of past experiences.

We provided a concise notation for specifying trust networks consisting of multiple paths between the relying party and the trusted party, and an algorithm for analyzing and deriving measures of trust in such environments. Our method, which is called Trust Network Analysis with Subjective Logic (TNA-SL), is based on analyzing trust networks as directed series-parallel graphs that can be represented as canonical expressions, combined with measuring and computing trust using subjective logic. We showed in a numerical example how *transitive trust* can be analyzed and computed using our method.

We presented three mechanisms to defend the UniTEC system against *Sybil attacks*: Combining UniTEC with trusted platform technology, assigning monetary value to a recommendation and taking this into account for the trust update, and finally combining UniTEC with the payment system SET to secure the attachment of value to the recommendation while still securing the strength of the pseudonyms. The gains on the reputation system side are an improved quality of recommendations by linking the recommendations to real transactions and therefore hindering attacks based on identity switching, copying of recommendations and malicious entities attaching themselves to reputable recommenders. Payment systems benefit from users being at ease with paying electronically due to good recommendations from other cardholders that have made already good experiences with certain merchants. Merchants behaving improperly will be identified and lose business, whereas reputable merchants may gain new customers and increase their revenue. Providing this trust enabling service might turn out to be a new business model for credit card companies like VISA or Mastercard and be a differentiator[1] among payment systems.

The *scientific results* [KR03, KP03, JGK03, KR04, KTR05a, KBR05, KTR05b, JGK06] were published in international journals and at conferences in the trust research field and are therefore accepted by the related work. In addition, a *prototype of UniTEC* was developed which enabled us to experiment with and show the feasibility of the theoretical concepts.

9.2 Discussion and Outlook

The mechanisms proposed in this work show that it is possible to create a *distributed reputation system* with *detailed trust modeling* and *enhanced privacy* as design features. It is clear that not all areas can be covered in equal depth. We now point out, where we see significant potential for further work.

The *neigborhood formation* process outlined in Subsection 3.3.4 is responsible for determining the subset of entities that the next request in the context area in question

[1]In 2000, lack of trust was identified by Cheskin research [Che00] as the major inhibitor of successful electronic commerce, whereas 6 years later in 2006 still 50% of the Internet users avoid buying online because of fear that their financial information might be stolen [Ver07]. We conclude that although the topic of trust has been researched in an IT environment for over a decade, the challenge of building trust in electronic payment systems is not adequately resolved yet.

is forwarded to. We pointed out that a certain number of trusted entities is chosen first, hub entities and authority entities are chosen next and a number of random entities last. It would be interesting to investigate the optimal neighborhood composition, such that a sufficiently high number of trusted recommendations can be received while not eliminating the possibility for getting access to new recommenders.

For users to be able to understand each other's trust assessments, a common understanding of the *trust context areas* is required. In the current version of UniTEC, the different trust context areas including the semantic distances between them are specified by the applications and can then be modified by each user. We do not assume that it is possible to define a single set of trust context areas that fits all applications. Therefore, in the case of several applications making use of the reputation system functionality, mechanisms should be developed for communicating context areas possibly including dependencies, and for integrating this knowledge into the local model. The context area sets could possibly be integrated into the concept of knowledge providers outlined in Subsection 3.3.1.

For the area of trust update algorithms, we see high potential for further research in analyzing *patterns of past experiences* to better detect misuse attempts and enhance the calculation of trust certainty. Furthermore, giving more weight to negative experiences as opposed to positive ones would be one improvement to better reflect social science research results which dictate that trust is hard to build, but easy to destroy.

For the area of transitive trust, it could be argued that negative trust in a transitive chain can have the paradoxical effect of strengthening the derived trust. Take for example the case where Bob distrusts Claire and Claire distrusts Eric, whereas Alice trusts Bob. In this situation, Alice might actually derive positive trust in Eric, since she relies on Bob's advice and Bob says: "Claire is a cheater, do not rely on her". So the fact that Claire distrusts Eric might count as a pro-Eric argument from Alice's perspective. The question boils down to "*Is the enemy of my enemy my friend?*", which relates to how multiple levels of distrust should be interpreted. This topic could be interesting to analyze in further research.

Appendix A

Notation

We use the following notation for describing structure and contents of packages that contain signed and encrypted blocks. It is used mainly in Chapters 5 and 8:

A_S, A_R	*IP address* of the agent, where the sender's pseudonym S or the recipient's pseudonym R is registered.
A_{M_X}	*IP address* of the agent M_X that has enabled its mix service.
Id_S, Id_R	Identifier of pseudonym S respectively R.
PrK_X, PuK_X	*Private key* respectively *public key* of entity X (X might be a pseudonym, mix, payment gateway, cardholder, or trust server).
K	*Symmetric key*.
$\{D_1, D_2\}$	Data item D_2 appended to data item D_1.
$Hash(D)$	*Hash* of data D (does not include the data D itself).
$Enc(PuK_X, D)$	Data D *encrypted* with the public key of entity X. Data encrypted with a symmetric key K is represented $Enc(K, D)$.
$Sign(PrK_X, D)$	Data D digitally *signed* with the private key of entity X (includes data D). This translates to $Sign(PrK_X, D) = \{D, Enc(PrK_X, Hash(D))\}$

Appendix B

List of Abbreviations

ACC	Anonymous Peer-to-Peer Communication Component
ACL	Access Control List
B2C	Business-to-Consumer
BIOS	Basic Input Output System
C2C	Consumer-to-Consumer
CA	Certification Authority
CRL	Certificate Revocation List
CRTM	Core Root of Trust for Measurement
DC	Dining Cryptographers
DHT	Distributed HashTable
DMC	Data Management Component
DSPG	Directed Series-Parallel Graph
EDR	Extended Destination Routing
ERH	Extended Routing Header
IMC	Identity Management Component
IRC	Internet Relay Chat
JAP	Java Anonymous Proxy
LAN	Local Area Network
MTT	Message Transfer Time
NET	Network Emulation Testbed
NGSCB	Next Generation Secure Computing Base
OStat	Originality Statement
P2P	Peer-to-Peer
PDF	Probability Density Function

PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
POC	Peer-to-Peer Overlay Component
PReq	Purchase Request
PRes	Purchase Response
QoS	Quality of Service
RCert	Recommender Certificate
RData	Rating Data
RecSigReq	Recommendation Signature Request
RecSigRes	Recommendation Signature Response
RID	Recommendation Identifier
RPC	Remote Procedure Call
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SAML	Security Assertion Markup Language
SCVP	Server-based Certificate Validation Protocol
SDSI	Simple Distributed Security Infrastructure
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SPKI	Simple Public Key Infrastructure
SSL	Secure Sockets Layer
STS	Security Token Service
TCG	Trusted Computing Group
TCP	Transmission Control Protocol
TCP	Trusted Computing Platform
TCPA	Trusted Computing Platform Alliance
TDI	Trusted Data Item
TET	Trusted Execution Technology
TID	Target Identity
TLS	Transport Layer Security
TMC	Trust Management Component
TPM	Trusted Platform Module
TransID	Transaction Identity
TTP	Trusted Third Party
TValReq	Transaction Value Request
TValRes	Transaction Value Response

TValue	Transaction Value
UniTEC	Universal Trust Architecture for Electronic Commerce
WS-Trust	Web Services Trust Language
XML	Extensible Markup Language

Bibliography

[AD01] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In Henrique Paques, Ling Liu, and David Grossman, editors, *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*, pages 10–317, Atlanta, November 2001. ACM Press.

[Aic04] Fabian Aichele. Realisierung und Evaluierung von Trust-Update-Algorithmen für das UniTEC Reputationssystem. Student Thesis No. 1906, Universität Stuttgart, Stuttgart, Germany, February 2004.

[ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui Hawaii, USA, January 2000.

[Bai01] Annette Baier. Vertrauen und seine Grenzen. In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 37–844. Campus Verlag, Frankfurt/New York, 2001.

[Bas04] Ernesto Baschny. Generische Modellierung und Aktualisierung von Vertrauen im UniTEC Reputationssystem. Diploma Thesis No. 2203, Universität Stuttgart, Stuttgart, Germany, October 2004.

[BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos Keromytis. The KeyNote Trust-Management System. RFC 2704, Network Working Group, IETF, September 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Proceedings of the 17th IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, USA, May 1996.

[BG05] Rajat Bhattacharjee and Ashish Goel. Avoiding Ballot Stuffing in eBay-like Reputation Systems. In *Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems (P2PECON '05)*, pages 133–137, New York, NY, USA, 2005. ACM Press.

[CDdV$^+$02] Fabrizio Cornelli, Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Choosing Reputable Servents in a P2P Network. In *Proceedings of the eleventh international conference on World Wide Web (WWW 2002)*. ACM, May 2002.

[CF05] Alice Cheng and Eric Friedman. Sybilproof Reputation Mechanisms. In *Proceeding of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON 2005)*, pages 128–132, Philadelphia, Pennsylvania, USA, August 2005. ACM Press.

[CH96] Bruce Christianson and William S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the International Workshop on Security Protocols*, pages 171–176, London, UK, 1996. Springer-Verlag.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Cha88] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[Che00] Cheskin Research. Trust in the Wired Americas, July 2000.
Available via *http://www.cheskin.com/*.

[Cla04] Andrew Clausen. The Cost of Attack of PageRank. In *Proceedings of The International Conference on Agents, Web Technologies and Internet Commerce (IAWTIC 2004)*, July 2004.

[CM05] Daniel Cvrček and Ken Moody. Combining Trust and Risk to Reduce the Cost of Attacks. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 372–383, Paris, France, May 2005. Springer-Verlag.

[CS01] Mao Chen and Jaswinder Pal Singh. Computing and Using Reputations for Internet Ratings. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 154–162, Tampa, USA, October 2001.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of the ICSI International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 46–66, Berkeley, CA, USA, June 2000. Springer-Verlag.

[CY01] Rita Chen and William Yeager. Poblano – A Distributed Trust Model for Peer-to-Peer Networks. Technical report, Sun Microsystems, Inc., 2001. Available via *http://www.jxta.org/project/www/docs/trust.pdf*.

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

[Deu62] Morton Deutsch. Cooperation and trust: Some theoretical notes. In M. R. Jones, editor, *Proceedings of Nebraska Symposium on Motivation*, pages 275–319, Lincoln, NE, USA, 1962. University of Nebraska Press.

[DFHM01] Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. In Ira S. Moskowitz, editor, *Proceedings of the 4th International Information Hiding Workshop*, volume 2137 of *LNCS*, pages 126–141, Pittsburg, PA, USA, April 2001. Springer-Verlag.

[DGdV01] C. K. W. De Dreu, E. Giebels, and E. Van de Vliert. Social motives and trust in integrative negotiation: The disruptive effects of punitive capability. *Journal of Applied Psychology*, 83:308–422, 2001.

[Dou02] John Douceur. The Sybil Attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pages 251–260, Cambridge, MA, USA, March 2002. Springer-Verlag.

[DS01] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics*. Addison-Wesley, 3rd edition, 2001.

[DW06] Wilhelm Dolle and Christoph Wegener. Trusted Computing für Linux: Stand der Dinge. *Linux Magazin*, 4, April 2006.

[EFL$^+$99] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory. RFC 2693, Network Working Group, IETF, September 1999.

[Eic05] Christiane Eichenberg. Interview: Können wir Computern vertrauen? *Explore*, 1:32–33, 2005.

[End01] Martin Endreß. Vertrauen und Vertrautheit – Phänomenologisch-anthropologische Grundlegung. In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 161–203. Campus Verlag, Frankfurt/New York, 2001.

[End02] Martin Endreß. *Vertrauen.* transcript Verlag, Bielefeld, Germany, 2002.

[FHM$^+$06] T. Freeman, R. Housley, A. Malpani, D. Cooper, and T. Polk. Server-based Certificate Validation Protocol (SCVP). Internet Draft, Network Working Group, IETF, December 2006.

[FL03] Paola Flocchini and Flaminia L. Luccio. Routing in Series Parallel Networks. *Theory of Computing Systems*, 36(2):137–157, January 2003.

[FM02] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 193–206, Washington DC, USA, November 2002. ACM Press.

[FR01] Eric Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[Gam00] Diego Gambetta. Can We Trust Trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237, Oxford, UK, 2000. Department of Sociology, University of Oxford.

[Gam01] Diego Gambetta. Kann man dem Vertrauen vertrauen? In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 204–237. Campus Verlag, Frankfurt/New York, 2001.

[Gid95] Anthony Giddens. *Konsequenzen der Moderne.* Suhrkamp, Frankfurt/M., Germany, 1995.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, volume 1174 of *LNCS*, pages 137–150, Cambridge, UK, May 1996. Springer-Verlag.

[Har01] Russel Hardin. Die Alltagsepistemologie von Vertrauen. In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 295–332. Campus Verlag, Frankfurt/New York, 2001.

[Her05] Daniel Herrscher. *Emulation von Rechnernetzen zur Leistungsanalyse von verteilten Anwendungen und Netzprotokollen.* PhD thesis, Universität Stuttgart, Stuttgart, Germany, December 2005.

[HO01] Martin Hartmann and Claus Offe, editors. *Vertrauen: Die Grundlage des sozialen Zusammenhalts.* Campus Verlag, Frankfurt/New York, 2001.

[Hob51] Thomas Hobbes. *Leviathan.* 1651. Available via *http://www.uoregon.edu/ rbear/hobbes/leviathan.html.*

[HR02] Daniel Herrscher and Kurt Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN'02)*, pages 262–267, Miami, USA, October 2002.

[Ing05] David Ingram. An Evidence Based Architecture for Efficient, Attack-Resistant Computational Trust Dissemination in Peer-to-Peer Networks. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 273–288, Paris, France, May 2005. Springer-Verlag.

[Int03] Intel. Trusted Execution Technology, Architectural Overview, 2003. Available via *http://www.intel.com/technology/security.*

[Int06] Intel. Trusted Execution Technology, Preliminary Architecture Specification, November 2006.

[JDV03] Audun Jøsang, Milan Daniel, and Patrick Vannoorenberghe. Strategies for Combining Conflicting Dogmatic Beliefs. In Xuezhi Wang, editor, *Proceedings of the 6th International Conference on Information Fusion*, 2003.

[JG03] Audun Jøsang and Tyrone Grandison. Conditional Inference in Subjective Logic. In Xuezhi Wang, editor, *Proceedings of the 6th International Conference on Information Fusion*, 2003.

[JGK03] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In Theo Dimitrakos and Fabio Martinelli, editors, *Proceedings of the First International Workshop on Formal Aspects in Security & Trust (FAST 2003)*, pages 9–22, Pisa, Italy, September 2003.

[JGK06] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent Systems – An International Journal*, 4(2):139–161, May 2006.

[JI02] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, Bled, Slovenia, June 2002.

[Jon96] Karen Jones. Trust as an Affective Attitude. *Ethics*, 107(1):4–25, October 1996.

[Jøs99] Audun Jøsang. An Algebra for Assessing Trust in Certification Chains. In J. Kochmar, editor, *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99)*. The Internet Society, 1999.

[Jøs01] Audun Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

[Jøs02] Audun Jøsang. Subjective Evidential Reasoning. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty (IPMU2002)*, Annecy, France, July 2002.

[JP05] Audun Jøsang and Simon Pope. Normalising the Consensus Operator for Belief Fusion. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, Sydney, Australia, 2005.

[JPD05] Audun Jøsang, Simon Pope, and Milan Daniel. Conditional Deduction under Uncertainty. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, 2005.

[JT99] Catholijn M. Jonker and Jan Treur. Formal Analysis of Models for the Dynamics of Trust Based on Experiences. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1999)*, volume 1647 of *LNAI*, London, UK, July 1999. Springer-Verlag.

[Kas04] Ulf Bernd Kassebaum. *Interpersonelles Vertrauen – Entwicklung eines Inventars zur Erfassung spezifischer Aspekte des Konstrukts*. PhD thesis, Universität Hamburg, Hamburg, Germany, 2004.

[KBR05] Michael Kinateder, Ernesto Baschny, and Kurt Rothermel. Towards a Generic Trust Model. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 177–192, Paris, France, May 2005. Springer-Verlag.

[KLR$^+$06] Dirk Kuhlmann, Rainer Landfermann, Harigovind Ramasamy, Matthias Schunter, Gianluca Ramunno, and Davide Vernizzi. An Open Trusted Computing Architecture – Secure virtual machines enabling user-defined policy enforcement, June 2006. Available via *http://www.opentc.net*.

[KMM$^+$97] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[KP03] Michael Kinateder and Siani Pearson. A Privacy-Enhanced Peer-to-Peer Reputation System. In Kurt Bauknecht, A Min Tjoa, and Gerald Quirchmayr, editors, *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003)*, volume 2738 of *LNCS*, pages 206–215, Prague, Czech Republic, September 2003. Springer-Verlag.

[KR03] Michael Kinateder and Kurt Rothermel. Architecture and Algorithms for a Distributed Reputation System. In Paddy Nixon and Sotirios Terzis, editors, *Proceedings of the First International Conference on*

Trust Management (iTrust 2003), volume 2692 of *LNCS*, pages 1–16, Crete, Greece, May 2003. Springer-Verlag.

[KR04] Michael Kinateder and Kurt Rothermel. Bringing Confidence to the Web - Combining the Power of SET and Reputation Systems. In *Proceedings of the 2004 IEEE Consumer Communications & Networking Conference (CCNC 2004)*, Las Vegas, NV, USA, January 2004. IEEE Communications Society.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.

[KTR05a] Michael Kinateder, Ralf Terdic, and Kurt Rothermel. Strong Pseudonymous Communication for Peer-to-Peer Reputation Systems. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005)*, Santa Fe, New Mexico, USA, March 2005. ACM.

[KTR05b] Michael Kinateder, Ralf Terdic, and Kurt Rothermel. Trusting Pseudonyms: Anonymous Communication in Peer-to-Peer Reputation Systems. *International Journal for Infonomics*, Special Issue 'Selected papers of the ACM SAC 2005 TRECK Track', September 2005.

[Lag01] Olli Lagerspetz. Vertrauen als geistiges Phänomen. In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 85–113. Campus Verlag, Frankfurt/New York, 2001.

[Lev04] Raph Levien. *Attack Resistant Trust Metrics*. PhD thesis, University of California at Berkeley, 2004.

[Lib03] Liberty Alliance. Liberty ID-FF Architecture Overview, Version 1.2., 2003. Available via *http://www.projectliberty.org*.

[LSM06] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Tech report 006-052, University of Massachusetts Amherst, Amherst, MA, USA, October 2006.

[Luh00] Niklas Luhmann. *Vertrauen: ein Mechanismus zur Reduktion sozialer Komplexität*. Lucius & Lucius, Stuttgart, Germany, fourth edition, 2000.

[Luh01] Niklas Luhmann. Vertrautheit, Zuversicht, Vertrauen. Probleme und Alternativen. In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 143–160. Campus Verlag, Frankfurt/New York, 2001.

[Mar94] Stephen P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[McL07] Carolyn McLeod. Trust. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2007.

[Mic07] Microsoft. BitLocker Drive Encryption: Technical Overview, Version 1.02, 2007. Available via *http://technet2.microsoft.com/WindowsVista/en/library/ce4d5a2e-59a5-4742-89cc-ef9f5908b4731033.mspx*.

[MMH02] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A Computational Model of Trust and Reputation. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, Big Island, Hawaii, January 2002. IEEE.

[Mui03] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2003.

[OAS06] OASIS Security Services TC. Security Assertion Markup Language (SAML) V2.0 Technical Overview, October 2006. Available via *http://www.oasis-open.org*.

[OAS07] OASIS Web Service Secure Exchange TC. WS-Trust 1.3, March 2007. Available via *http://www.oasis-open.org*.

[Off01] Claus Offe. Wie können wir unseren Mitbürgern vertrauen? In Martin Hartmann and Claus Offe, editors, *Vertrauen: Die Grundlage des sozialen Zusammenhalts*, pages 241–294. Campus Verlag, Frankfurt/New York, 2001.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[Pea02] Siani Pearson, editor. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall, September 2002.

[PJ05] Simon Pope and Audun Jøsang. Analysis of Competing Hypotheses using Subjective Logic. In *Proceedings of the 10th International Command and Control Research and Technology Symposium*, 2005.

[PTJL05] Jigar Patel, W. T. Luke Teacy, Nicholas R. Jennings, and Michael Luck. A Probabilistic Trust Model for Handling Inaccurate Reputation Sources. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 193–209, Paris, France, May 2005. Springer-Verlag.

[PW87] Andreas Pfitzmann and Michael Waidner. Networks without User Observability. *Computers and Security*, 6(2):158–166, April 1987.

[QHC06] Daniele Quercia, Stephen Hailes, and Licia Capra. TATA: Towards Anonymous Trusted Authentication. In Ketil Stølen, William H. Winsborough, Fabio Martinelli, and Fabio Massacci, editors, *Proceedings of the Fourth International Conference on Trust Management (iTrust 2006)*, volume 3986 of *LNCS*, pages 313–323, Pisa, Italy, May 2006. Springer-Verlag.

[RK05] Sini Ruohomaa and Lea Kutvonen. Trust Management Survey. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 77–92, Paris, France, May 2005. Springer-Verlag.

[RKZF00] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, December 2000.

[RL96] Ron Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure. In *Proceedings of CRYPTO'96*, 1996.

[RR99] Michael K. Reiter and Aviel D. Rubin. Anonymity loves company: Anonymous Web transactions with Crowds. *Communications of the ACM*, 42(2):32–38, February 1999.

[RZSL06] Paul Resnick, Richard Zeckhauser, John Swanson, and Kate Lockwood. The Value of Reputation on eBay: A Controlled Experiment. *Experimental Economics*, 9:79–101, June 2006.

[Sab03] Jordi Sabater. *Trust and Reputation for Agent Societies*. PhD thesis, Institut d'Investigació en Intelligència Articial, Bellaterra, 2003.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, second edition, 1996.

[SGJ05] Jean-Marc Seigneur, Alan Gray, and Christian Damsgaard Jensen. Trust Transfer: Encouraging Self-recommendations Without Sybil Attack. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Proceedings of the Third International Conference on Trust Management (iTrust 2005)*, volume 3477 of *LNCS*, pages 321–337, Paris, France, May 2005. Springer-Verlag.

[SJ04] Jean-Marc Seigneur and Christian Damsgaard Jensen. Trading Privacy for Trust. In *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, volume 2995 of *LNCS*, pages 93–107, Oxford, UK, March 2004. Springer-Verlag.

[SKB$^+$98] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using Filtering Agents to Improve Prediction Quality in the Grouplens Research Collaborative Filtering System. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work (CSCVO)*, pages 345–354, Seattle, November 1998.

[Smi82] Adam Smith. Lectures on Jurisprudence. In R. L. Meek, D. D. Raphael, and P. G. Stein, editors, *Glasgow Edition of the Works and Correspondence of Adam Smith*, volume 5. Liberty Fund, Indianapolis, USA, 1982.

[SMK$^+$01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, San Diego, CA, USA, August 2001.

[SS01] Jordi Sabater and Carles Sierra. REGRET: reputation in gregarious societies. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 194–195, Montreal, Canada, 2001. ACM Press.

[ST03] Martin Schweer and Barbara Thies. *Vertrauen als Organisationsprinzip: Perspektiven für komplexe soziale Systeme*. Verlag Hans Huber, Bern, Switzerland, 2003.

[SZ06] Christoph Sorge and Martina Zitterbart. A Reputation-Based System for Confidentiality Modeling in Peer-to-Peer Networks. In Ketil Stølen, William H. Winsborough, Fabio Martinelli, and Fabio Massacci, editors, *Proceedings of the Fourth International Conference on Trust Management (iTrust 2006)*, volume 3986 of *LNCS*, pages 367–381, Pisa, Italy, May 2006. Springer-Verlag.

[Ter03] Ralf Terdic. Konzeption und Realisierung eines Peer-to-Peer Empfehlungsdienstes. Diploma Thesis No. 2097, Universität Stuttgart, Stuttgart, Germany, October 2003.

[Tho06] Alexander Thomas. Vertrauen im interkulturellen Kontext aus Sicht der Psychologie. Thesenpapier, Universität Regensburg, Institut für Experimentelle Psychologie, Regensburg, Germany, 2006.

[Tru01] Trusted Computing Platform Alliance. TCPA Main Specification, Version 1.1, 2001. Available via *http://www.trustedcomputing.org*.

[Tru04] Trusted Computing Group. TCG Specification, Architecture Overview, Revision 1.2, April 2004. Available via *http://www.trustedcomputing.org*.

[Ver07] VeriSign. The VeriSign Secured Seal Research Review, 2007. Available via *http://www.verisign.com*.

[VM97] Visa International and Mastercard International. SET Secure Electronic Transaction Specification Book 3: Formal Protocol Definition, May 1997.

[Wid03] Jochen Widmaier. Identitäts-Management in Peer-to-Peer Umgebungen. Diploma Thesis No. 2094, Universität Stuttgart, Stuttgart, Germany, October 2003.

[WP89] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology EUROCRYPT 89*, volume 434 of *LNCS*, Houthalen, Belgium, April 1989. Springer-Verlag.

[YKGF06] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 267–278, Pisa, Italy, September 2006. ACM Press.

[YS02] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 294–301, Bologna, Italy, July 2002. ACM Press.

[Zim95] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

[ZL04] Cai-Nicolas Ziegler and Georg Lausen. Spreading Activation Models for Trust Propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04)*, Taipei, Taiwan, March 2004. IEEE Computer Society Press.

[ZMM99] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative Reputation Mechanisms in Electronic Marketplaces. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Maui Hawaii, January 1999.