Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelor Thesis

Connor Röhricht

Applications of Decentralized Trust Management in V2X Communications

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Connor Röhricht

Applications of Decentralized Trust Management in V2X Communications

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Christoph Klauck
Zweitgutachter : Dipl.-Ing. Frank Siedel

Abgegeben am 27.10.2015

Connor Röhricht

Thema der Arbeit
Anwendungen eines Dezentralisierten Vertrauensmanagements in der V2X-Kommunikation

Stichworte
Vertrauen, Dezentralisiertes Vertrauensmanagement, Sicherheit, V2X, Internet of Things

Kurzzusammenfassung
Die zunehmende Konzeption und Implementierung sogenannter V2X-Dienste, basierend auf intelligenten kommunizierenden Fahrzeugen, stellt die Automobilindustrie und andere Stakeholder vor eine Vielfalt neuer Herausforderungen. Ein sicheres Vertrauensmanagement ist Kern jeder VANET (= Vehicle Ad-Hoc Network) Anwendung. Ähnliche Herausforderungen entstehen in anderen Bereichen der Software-Branche durch die Ankunft allgegenwärtiger (Mikro-)Computer und des sogenannten *Internet of Things*. Die für diese Umgebungen charakteristischen Vertrauensprobleme sind jedoch noch relativ oberflächlich erforscht. Traditionelle Sicherheitssysteme sind oft zu statisch und einfach um die Anforderungen komplexer und dynamischer Multiagentensysteme zu erfüllen. Diese Arbeit bietet eine Einführung in bestehende Konzepte für Vertrauensmanagement, was mit der Entwicklung eines generischen abstrakten Vertrauenssystems abschließt. Die Ergebnisse werden im V2X-Kontext untersucht und die Eigenschaften des Systems in einer simulierten VANET-Umgebung analysiert.

Connor Röhricht

Title of the paper
Applications of Decentralized Trust Management in V2X Communications

Keywords
Trust, Decentralized Trust Management, Security, V2X, Internet of Things

Abstract
The increasing conception and implementation of so-called V2X services based on smart communicating cars presents the automotive industry and other stakeholders with a variety of new challenges. A secure trust management is fundamental to any VANET (= Vehicle Ad-Hoc Network) application. Similar challenges also arise in other software domains with the approach of *ubiquitous computing* and the *Internet of Things*. However, the trust-related problems that are characteristic of these new environments have been subject to relatively little research. Traditional security systems are often too static and simple to meet the requirements of complex and dynamic multi-agent systems. This thesis offers a survey of existing concepts of trust management, which culminates in the design of a generic abstract trust system. The results of this are evaluated in the context of V2X communications, and the trust system is analyzed in a simulated VANET environment.

Table of Contents

1 Introduction

1.1 Introduction

The vision of ubiquitous computing and the "Internet of Things" has been slowly solidifying within the last years. Smartphones and wearable technology such as smartwatches are already transforming humans into "always online" entities that can interact with remote real or virtual objects at any time, and our environment is gradually entering our digital network[1]. Notable examples of this include smart homes, smart vehicles, and the monitoring and control of infrastructure and machinery.

It seems unavoidable that the number of digitally connectable objects with embedded computing systems will substantially rise in the near future. These devices will interact with each other and with humans, either representing their owner or manufacturer according to simple deterministic programming, or acting relatively autonomously (ambient intelligence).

As the number of different types and brands of smart devices rises, the number of possible scenarios where heterogeneous devices should interact rises too, and it becomes increasingly difficult to predefine behavior. This naturally raises concerns regarding security and privacy; both humans and autonomous intelligent devices face questions such as "Whom can I trust to not abuse my personal information?", "Which data can I rely on?" and "Is anyone trying to manipulate my behavior?". All of these questions are already present in human societies. We intuitively answer them using highly complex and dynamic, often subconscious, mechanisms to manage interpersonal relationships and assessing the reliability of interaction partners, i.e., their trustworthiness.

Similar mechanisms can be found in almost every piece of information technology; however, traditional mechanisms such as rule-based access control lack both the fine granularity and the dynamism that their social equivalents offer. They are also often based on a single central perspective of trustworthiness, which is by nature subjective; hence they lack the scalability to model complex relationships among users or nodes in highly

[1] Which is often the global internet.

distributed systems. As a consequence, conceptions of trust management, i.e., of subsystems that operate on symbolic representations of trust, move into the focus of research; these systems often aim to solve the aforementioned issues with a decentralized architecture.
Nevertheless, the abstract concept of trust and especially decentralized trust management is still theoretically underrepresented in the fields of computer sciences and software engineering. A useful practice-oriented model system could be found in V2X communications, emerging technologies that wirelessly connect intelligent vehicles in order to enable software cooperation in traffic. In this thesis, I will investigate core aspects of decentralized trust management and their applications in V2X environments, aiming to give the reader an insight into the underlying theory and its future significance. In writing it, I was supported by NXP Semiconductors, with resources, coordination, and feedback from colleagues for whose work this research could be relevant.

1.2 Task Formulation

We[2] set ourselves the following objectives:

- To provide a short introduction to the challenges of V2X communications
- To gain a deeper understanding of the meaning and properties of trust in open multi-agent-systems[3]
- The design of a generic abstract system that integrates the necessary functionality to operate a user-centric trust management system. A future developer of a concrete, application-specific trust system could then benefit from our survey of mechanisms and specialize the scheme we propose
- The design and development of simulation software that can be utilized to investigate the properties of our system in a V2X environment

and

- To identify and suggest potential future areas of application for trust management schemes (also outside of V2X communications).

1.3 Scope

The target audience for this paper consists of holders of a Bachelor of Science degree in applied computer science; the reading should not require expertise in traditionally unrelated fields. We will therefore venture into the sociological and ethical theories of trust not further than necessary, and limit the mathematical effort to basic formal definitions.
We will investigate different theoretical aspects of trust with the aforementioned goals in mind; nevertheless, some concepts need to be included for completeness, without factoring into the design of our abstract system and simulation software. Otherwise, we

[2] I will use the academic plural pronoun from here on, which is not to be seen as a Sybil attack to gain the reader's trust. ☺
[3] We will elaborate on this term in section 3.1.6.

would give a reader, that is unfamiliar with the matter, an understanding of trust management theory that is biased towards the aspects we regard as important for our work. For readers that are only interested in our results and their derivation, we provide details on the contents of each section in section 1.4, including which aspects could be skipped.

1.4 Structure and Methods

To begin with, we quickly survey the current state of V2X technology without going into too much detail in section 2. We mostly focus on the new risks and challenges for V2X systems, to which decentralized trust management could give remedy.
In the subsections of the sections 3.1 and 3.2, we tackle the questions posed and answered by trust as a concept and trust management systems, incrementally. By gradually including important aspects of the underlying theories, we aim extend our understanding, while guiding the reader through the process. In order to do this, we provide quick surveys of existing ideas and approaches, draw conclusions, and provide illustrative examples and abstractions.
Generally, section 3.1 deals with trust between individuals and is mostly based on a local view of trust relations, while section 3.2 rather describes the global effects of trust in larger systems.
We begin this investigation with the notion of trust and formalize it, without specifying too concrete representations of individual statements, since the requirements on the nature of these statements vary considerably among different approaches and areas of application (see sections 3.1.1-3.1.3). Subsequently, we investigate trust in objects (section 3.1.4), which is mostly for completeness and might be skipped by the reader, although we reference some core aspects later on. In section 3.1.5, we consider trust as a computational concept, which is important for software implementations. In section 3.1.6, we take a short excursus into agent ethics to justify the need for trust in an open multi-agent system; concepts introduced here are then applied in section 3.2. Section 3.1.7 deals with the notion of trust transitivity and introduces the trust coherency problem, a novel approach for modelling trust propagation. Section 3.1.8 discusses the dynamic nature of trust and its causes at the individual level; this could also be skipped by readers only interested in the results. Lastly, in section 3.1.9, we aim to provide a clear distinction between trust and the similar term reputation that we will often refer to in the following section.
Subsections of 3.2 illuminate various aspects of the design of trust systems, including the specification of ideal agent behavior and the use of trust for resource management.
The sections 3.3 and 3.4 compare the advantages and disadvantages of (de-)centralization in trust systems. We then design and propose our aforementioned generic abstract trust system in section 3.5.
The initially unrelated contents of the sections 2 and 3 are combined in section 4. As promised by the title of this thesis, we investigate the applications of decentralized trust management in V2X communications. We extend an existing VANET simulation software to

include a basic trust system based on the results of chapter 3. This trust system aims to solve some of the obstacles referred to in section 2, within the simulation. Subsequently, we test our system with different trust-related scenarios and analyze the results.
Lastly, we list some potential future areas of application of decentralized trust systems (section 5), and draw conclusions from all previous sections while summarizing our results in section 6.

2 Introduction to V2X Communications

2.1 Introduction to V2X Services

2.1.1 V2X

The term “V2X” covers the services of V2V (= Vehicle-to-Vehicle) and V2I (= Vehicle-to-Infrastructure) communication. The common denominator of both technologies lies in the conception of road-bound vehicles as communicating nodes, and also, in many applications, as at least partially autonomous agents. Current implementations are predominantly experimental and not commercially available. Their present development is driven by a demand for increased traffic security and efficiency, as well as the emerging trends of ubiquitous computing. It is creating new challenges for all branches of the automotive industry.

The mobility of the participants requires wireless connections with fast buildup and clearing (Ad-Hoc connections). As cars currently do not include the necessary hardware to support a V2X infrastructure, mass deployment must be achieved before individual drivers can profit from cooperative V2X services.

V2X usually denotes the network and transportation layers of intervehicular communication, but is also used as an umbrella term for all services and applications that are primarily based on it. To avoid ambiguity, we will refer to those as V2(X/V/I) services.

2.1.2 V2V

In Vehicle-to-Vehicle communication, road-bound vehicles (mostly cars) exchange data over a peer-to-peer connection. This data can describe the vehicles’ internal states and environment, to expand the communication partner’s perceptual horizon. This expanded horizon, also known as “telematics horizon”, closes the gap between local perception (given by the driver’s field of view and onboard sensors) and long distance communication (e.g. traffic reporting), as shown in Figure 2.1.2-I.

Intervehicular communication is necessary to prevent fatal road accidents, and has always been present in the form of automotive lighting, such as turn signals and brake lights. It seems natural for it to evolve into V2V communication in the digital age.

If single nodes can also act as routers and forward messages to a third party[4], the resulting network is a VANET (= Vehicular Ad-Hoc Network). These networks can be regarded as a dynamic peer-to-peer variant of the mesh network topology (= MANET), with additional requirements brought forth by the high, yet very predictable, node velocities.
VANETs could form the basis for future implementations of Intelligent Transportation Systems (ITS).

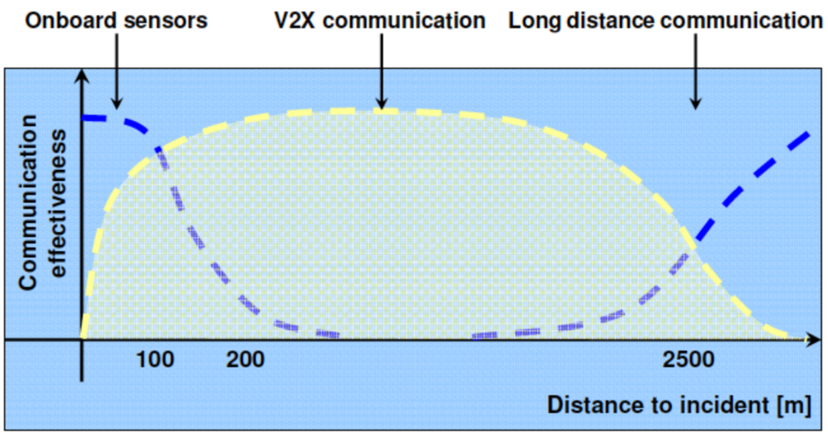


Figure 2.1.2-I: V2X closes the communication gap. Source: [SCHM08]

2.1.3 V2I

The United States Department of Transportation defines V2I communication for safety purposes as follows:

(2.1.3A) US DoT definition of V2I for safety purposes:
"Vehicle-to-infrastructure (V2I) Communications for Safety is the wireless exchange of critical safety and operational data between vehicles and roadway infrastructure, intended primarily to avoid motor vehicle crashes." [USDO15]

This wireless exchange of data can also be used to provide information and entertainment services to drivers and passengers. Automotive entertainment devices could, for example, download current weather forecasts, traffic information, or music and other media.
As with V2V services, predecessors of this technology have been around since the adoption of motor vehicles, initially in static form (such as road signs), but also in dynamic form (such as traffic lights). While most of their current manifestations can information-theoretically be regarded as unidirectional broadcasts in information theory, V2I services enable a bidirectional communication. Vehicles can interact with infrastructure, for instance to alter the duration of traffic light phases or to report traffic situation and internal state of the vehicle to an ITS. Roadway infrastructure could also be used for routing purposes in a

[4] Recently this term has also been extended to any intervehicular network, even if it only contains direct communication and incorporates V2I services.

VANET. A VANET incorporating roadway infrastructure might be connected to the internet, thereby integrating in-car devices into the Internet of Things.

2.1.4 Technological Support

Implementation of V2X services requires the installment of antennas, WLAN modules and dedicated controllers in cars. V2V services often need positional data, especially for accident prevention, and thus require the integration of a GPS component. To provide safety in case of malfunction, data from additional sensors like cameras and ultrasonic distance detection can be evaluated to prevent "insane" decisions. Cameras can also recognize traffic signs and other vehicles' visible characteristics (e.g. license plates) to assist in driving decisions and identification of communication partners; this would provide backwards compatibility with the current system until V2X is ubiquitous.

2.2 Applications and Chances

2.2.1 Categorization

V2X services, once of purely academic interest, have moved into the center of the automotive industry's attention, creating a variety of related use- and business cases. In this section, I will present and categorize future applications of V2X services, and discuss their benefits in accident prevention and traffic optimization.
First of all, we will focus only on traffic-related services, excluding, for example, in-car entertainment. Traffic-related V2X services aim to leverage the collaborative power of traffic participants and their onboard sensors to optimize and secure traffic.
Ways to categorize these services for further analysis include:

- By their underlying network structure (V2V, V2I, VANET…)
- By their main user groups (drivers, governmental agencies, manufacturers)
- By their scope of influence on traffic

Since the means of influencing the traffic also determine the set of entities that users put trust in, I will choose the latter for primary classification of V2X services.
The table in Figure 2.2.1-I gives an overview of the primary channels through which V2X applications can affect traffic, or individual vehicle behavior.
For a more detailed list of possible usecases (within the European traffic system), see [ETSI09].

Name	**Actor (Scope)**	**Decision-making**	**Communication**
Driver assistance	Driver & Vehicle	Driver & Vehicle	V2X and HIDs
Autonomous vehicle	Vehicle	Vehicle	V2X
ITS	Vehicle & Infrastructure	Central authority	V2X

Figure 2.2.1-I: V2X service categories according to scope of influence

2.2.2 Driver Assistance Services

Driver assistance services use a Human-Machine-Interface to convey critical information to the driver of the receiving vehicle. While basic applications such as lane departure warning already exist, they use only the vehicle's onboard sensors, instead of wireless communication with other observers. V2X communication can be integrated with these applications to increase the safety of both technologies, while simultaneously expanding the driver's perceptual horizon.

The unique characteristic of these services lies in the decision making process. Unlike in autonomous applications, the driver is in full control of the vehicle, and responsible for preventing accidents and complying with traffic laws. Since this corresponds to the status quo, adaptation to new technology will be both easier and faster than to autonomous services.

In the case of software failure (or breach of security), the driver can use their own observations to prevent potential accidents. The already existing status of safety is largely retained. Nevertheless, as V2X technology proves successful, driver can become more reliant on it, allowing for more relaxed driving.

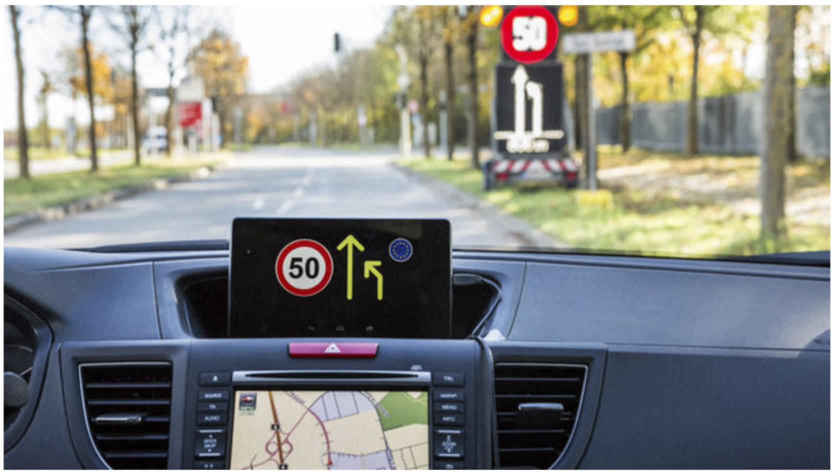


Figure 2.2.2-I: HID for driver assistance services. Source: NXP Semiconductors

Other subcategories of driver assistance services include:

- Warning of approaching emergency vehicles
- Warning of other vehicles in anomalous state (slow/stationary/wrong way driving)
- Assistance in cooperative maneuvers (overtaking/merging/left-turn assistance)
- Optimal speed advisory (e.g. approaching traffic lights)[5]

[5] This can also optimize fuel efficiency and offset the cost of V2X implementation through the beneficial ecological and economical impact.

2.2.3 Autonomous Vehicle Services

Autonomous, or self-driving, vehicles are capable of performing some (or, in a possible future, all) traffic maneuvers without human input.

All autonomous services, when implemented in a secure and safe way, increase safety and efficiency in traffic by eliminating the delay caused by human reaction time. An intelligent agent can use onboard sensors to observe the vehicle's environment, but will most likely need to communicate with other agents, or roadway infrastructure, by means of V2X communication.

Plans for the initial implementation of autonomous vehicle services usually include the driver, who can take control of the vehicle in emergency situations, possibly caused by malfunction of the system. However, there are also scenarios in which the AI might need to intervene to prevent an accident caused by human error[6]. If V2V services detect an impending collision, the software could trigger an emergency stop. Nevertheless, one could imagine completely autonomous, driverless vehicles of the future that humans can give orders to.

2.2.4 Intelligent Transport System Services

In an Intelligent Transportation System (ITS), an intelligent software analyzes, controls and optimizes traffic. Vehicles are only partially autonomous and serve as both sensors and actors of the ITS. The ITS can also influence dynamic roadside infrastructure such as traffics lights, direction-changing lanes and adjustable speed limits (while currently rare, this infrastructure already exists).

The resulting global[7] view and control permits the ITS to operate on large-scale emergent patterns, such as traffic flows and jams. Especially the prevention of traffic flow is a promising feature of ITSs. ITSs could also optimize logistics and public transport.

2.3 Risks and Limitations

2.3.1 Performance

When exchanging safety critical information, vehicular agents need to transmit and process messages within very short time frames in order to avoid collisions. Both [ETSI09] and [NHTS14] primarily list usecases that require latencies (reaction times) of less than 100 ms at message frequencies of 1-10 Hz, but also describe usecases with less than 50 ms or even 20 ms allowable latency. At intersections or in traffic jams, vehicles can easily become surrounded by more than 50 other nodes within communication range. Since most scenarios specify point-to-multipoint broadcast messages, even assuming a message frequency of only 1 Hz, each vehicle would have to process each incoming message in less

[6] Or sudden incapacity, for instance caused by a heart attack.

[7] In the mathematical sense, not "surrounding the globe".

than 20 ms in such a situation. Otherwise, the vehicular agent would risk discarding an important safety message.
“In order to create the required environment of trust, a V2V system must […] credential each message [...]” ([NHTS14] preface p. xviii). With currently available embedded technology, the verification of a secure signature in this timeframe, combined with the additional load of other cryptographic operations such as key generation for pseudonym changes and signing own message would hardly be feasible. Fortunately, not all messages need to be verified, as most of them would be irrelevant for the agent’s operation. To ignore unimportant messages, the agent needs to partially interpret them before performing any expensive operation. This prevents encryption of the messages (or at least complete encryption), since decryption is both expensive and a prerequisite to interpretation. To allow for partial encryption, the nodes would have to attach unencrypted promises of severity to their message that the recipient can utilize for prioritization at peak times. As of now, most conceptions of future V2V implementations do not include the encryption of messages.
The expensiveness of signature verification also renders VANETs susceptible to Denial-of-Service (DoS) attacks carried out by intruders in the wireless medium.

2.3.2 Safety & Security

Due to the high risks involved, trust is an important quantity in a V2X environment; trusting an incorrect message could potentially cause a dangerous or even fatal accident when vehicles are autonomous. As mentioned in section 2.1.4, sensory data can be utilized to prevent unsafe decisions, but a high level of trust is nevertheless required to enable operation especially during cooperative maneuvers. Message must be authenticated and the signatures must be non-repudiable to prevent abuse.
Current approaches call for the integration of tamper-resistant devices, but a V2X solution would contain a variety of components that are hard to integrate in a way that also prevents the exchange of single components, which could be used to control other tamper-resistant components by simulating a different environment. Trustworthy identities would be managed by certificate authorities; this opens another area of attack, an attacker that can temporarily gain control over the registration component could issue new identities that are falsely recognized as trustworthy. It would therefore be imprudent to assume that no incidents would ever occur where a hacker gains control over a node identity and can launch a so-called *insider attack*. As a consequence, the underlying trust relations between nodes in the traffic system change; unlike V2X nodes, humans did not have to be on the lookout for other drivers that try to manipulate them.

A survey on security attacks in VANETs, [ALKA12], lists the following attack scenarios:

(2.3.2A) Security attacks in VANETs:

(1) *Bogus information / Illusion attack*
A malicious node purposefully spreads wrong information on traffic situation. The paper defines an illusion attack[8] as the variant where the adversary simulates a fake environment to its own sensors to circumvent tamper-resistant devices.

(2) *Denial-of-Service*
A malicious node broadcasts much more messages than necessary, jamming channels and reducing performance and efficiency of the network.

(3) *Masquerade / Impersonation attack*
A malicious node assumes a fake but verifiable identity by breaking authentication security and sending on behalf of others or under a new generated identity, or by resending old messages signed by other nodes (replay attack[9]).

(4) *Timing attack / Black Hole attack*
When forwarding a message, a malicious node fakes a higher processing time (or for the latter, does not forward at all) to introduce delay and reduce network performance and distribution of the message.

(5) *Sybil attack*
An attacker generates or assembles multiple identities to simulate multiple nodes.

Note that some attacks listed in [ALKA12] are not included here since they are countered by defense strategies that are already included in the established prerequisites for VANET operation.
One might question whether there exist sufficient motives for attackers on V2X systems. To answer this, there are also examples of such attacks without digital V2X systems. A V2V trusted warning technology that predates V2X systems is found in the flashing lights of emergency vehicles. In 2010, a protest movement (known as the Society of Blue Buckets) emerged in Russia, against the abuse of such lights by “VIP” vehicles of government officials, businessmen and celebrities. This abuse was often tolerated by government agencies and occasionally resulted in serious accidents; one could imagine similar scenarios with trusted V2V messages, where the driver might not even be able to recognize the abuse.

2.3.3 Privacy

The proposal of large-scale V2X implementation naturally raises some questions regarding user privacy. A car is usually regarded as a private place, similar to a home, and people dislike the idea that corporations or government agencies could observe their actions or collect data that describes their behavior.

[8] The term illusion attack is often also used to describe attacks where the illusion only affects the other nodes.

[9] The latter can easily be prevented by attributing messages with time and position information.

Perhaps the most prominent privacy concern with V2X technology is the possibility of location tracking. Operators of V2X infrastructure could store the location data of interacting vehicles permanently, and use this data to create movement profiles of individuals. A second possible attack scenario on privacy includes the establishment of a wireless sniffer network by malicious third parties. This could be possible, since messages are sent unencrypted in most approaches (due to the broadcast nature of safety messages) and often contain identification information (for example, via the certificate used to create the digital signature). Vehicles already possess publicly visible identification markers, namely license plates, but V2X communication would lower the cost of a tracking network based on such markers significantly.

Current approaches deal with privacy concerns by equipping vehicular nodes with huge sets of pseudonyms that the node can switch between at random intervals. Provided that observers are not able to link pseudonyms[10], this protects users from tracking by third parties, but not from tracking by infrastructure operators.

2.3.4 Deployment

The future deployment of many V2X technologies faces a “chicken-and-egg”-problem: Their quality depends on current distribution; if too few vehicles contain V2V systems, they have nobody to communicate with. Getting many vehicles to include such systems requires getting many customers to buy such systems (which could be costly), but the willingness of customers to buy them depends on the quality again.

Government agencies could mandate the integration of V2X systems in the future; however, this would raise even more privacy concerns.

This process is further obstructed since there is no singular entity that would produce, integrate and deploy V2X systems. The cooperation of many manufacturers is necessary, and they would have to agree on common standards and implement them correctly. With current approaches, all of these conditions would be a precondition to the initial deployment. If the V2X then would not find sufficient user acceptance, an gigantic economic value of work effort would have been wasted.

[10] For example, by observing the switch.

3 Decentralized Trust Management

3.1 The Notion of Trust

3.1.1 Defining Trust

As of this writing, the Merriam-Webster's Collegiate Dictionary [MERR03] lists 5 possible definitions of the word trust, all with a different meaning. What they have in common is that they are based on either social interaction or action theoretical[11] concepts.
Since we aim to create a model in which trust directly affects the behavior of an agent, it seems sensible to choose a definition that reflects the action theoretical impact:

(3.1.1A) "[Trust is] assured reliance on the character, ability, strength, or truth of someone or something." (keyword "trust". Retrieved 2015-06-27, from the online version of [MERR03])

According to [MCKN96] (p. 3), "Trust makes cooperative endeavors happen". The intent to make cooperative endeavors happen lies at the very core of our investigation, so trust seems to be a useful concept. However, the second disjunct in (3.1.1A), "reliance on [...] something", suggests that this might not always be the case: agents can cooperate, but inanimate objects (= something) can not; hence we ignore the second disjunct for now.
The first disjunct in (3.1.1A), "reliance on [...] someone", describes the social aspect of trust, humans trusting each other. Yet we hope to implement trust in a system composed of vehicular nodes and other artificial agents. This is why we extend this definition to any pair of homogenous entities. We refer to this kind of trust as *entity trust*[12].
In order to scientifically study this concept and to embed it in software architecture, we should begin by representing it in a formal, mathematical, model.

3.1.2 Formal Representation of Entity Trust

In what follows, let E denote the aforementioned set of homogeneous entities which can trust one another. To find an appropriate mathematical representation for the concept of

[11] Philosophical action theory, not to be confused with social action theory.
[12] Note that this is not completely identical with the interpretation of entity trust in [MASH11].

trust, we analyze its properties in common language usage. The simplest expression of entity trust in normal speech is:

(3.1.2A) "A trusts B"[13]

This sentence describes a relationship, which prompts us to model entity trust as a binary relation over E^2. We call this simple representation of trust a *binary trust relation*. Obviously, (3.1.2A) does not imply the inverse, hence a *binary trust relation* is directed and not necessarily symmetric. We call the entity A in this circumstance the *trustor*, and B the *trustee*.
The existence of sentences such as

(3.1.2B) "A trusts B a lot" and
(3.1.2C) "A barely trusts B"

indicates that there can be varying degrees of trust, we call these *trust values*. To incorporate trust values, we need to extend our trust relation. Let T denote a set of trust values. We define the *entity trust relation* $R \subseteq E^2 \times T$, with $(A, B, \varphi) \in R$ iff the entity A trusts the entity B to a degree represented by φ. (3.1.2B) and (3.1.2C) are mutually exclusive; an entity can not trust another to multiple degrees at the same time[14]. R is therefore right-unique. If we require modelling every possible trust relationship between entities as a trust value[15], R becomes also left-total. We can regard it as a function, the *trust function*.
Since we have not yet considered a potential context dependency of trust, we simply assume that the trust function models a context-independent kind of trust, for now. This does not hinder us from modelling context-dependent trust, since we could simply define a set of trust functions and let an agent choose the most appropriate one for a given situation. Also, we did not impose any requirements on the nature of trust values, so they could be functions whose common domain is composed of contexts[16].
Because our interpretation of trust requires it to be relevant for an agent's decisions, it needs to be interpretable via a structure attached to the set of trust values T, rendering it a space: the *trust space*. Concluding this section, we arrive at the following definition:

(3.1.2D) Definition of a context-free trust function:
A context-free trust function over a set of entities E and a trust space T is a function

$$t_{free}: E \times E \to T$$

[13] $A, B \in E$
[14] Regarding the same behavior in the same situation. This dependency on context is discussed in section 3.1.3.
[15] This includes the absence of any knowledge regarding the other's trustworthiness.
[16] This is equivalent to the currying of a context-dependent trust function.

3.1.3 Semantics of Entity Trust

The previous two sections provided us with a basic syntax and constraints for modelling (entity) trust, but not with any advice on how an agent should interpret a trust value. What does it mean to trust an individual? Jøsang et al.[17] distinguish between *reliability trust* and *decision trust*. For the former, they usually give a definition based on the definition of trust in [GAMB00]:

(3.1.3A) Definition of reliability trust[18]:
"Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends." ([JOSA05] p. 2)

Is this definition adequate for discussing the impact of entity trust on cooperative behavior in our setting? Cooperative behavior does not necessarily require performing an action. When interfering in a situation would reduce the global benefit, passive behavior is cooperative; thus, one agent can trust another to not act maliciously. Moreover, the term "subjective probability"[19] seems misleading, at least in our case of application, because the trusting expectation lacks the property of negation. Not completely relying on cooperative behavior does not imply relying on uncooperative behavior to some degree.
The definition of *decision trust* integrates a relevance for own decision making:

(3.1.3B) Definition of decision trust:
"Trust is the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible." ([JOS205] p. 1)

While this is slightly misquoted from [MCKN96], where this defines *trusting intention*, our focus on cooperative behavior makes it desirable to include the decision making process. Combining these aspects of trust, we define entity trust as follows:

(3.1.3C) Definition of entity trust:
Entity trust is the certainty with which a decision making entity assumes that another such entity shows cooperative behavior regarding a given domain of action.

Attentive readers might have noted two oddities in this definition: Firstly, we need to elaborate on the term "certainty", we will address its implications in section 3.1.6. Secondly, we replaced the "given situation" from (3.1.3B) with a "given domain of action". A reference to context is necessary. We may trust a person to give a lecture, but at the same time not trust them to fly an airplane ([DENN93] p. 2). However, we do not need to consider the entire situation. Trusting a pilot to fly a plane usually comes along with trusting them to fly any other plane of the same class, regardless of the plane's serial number or the names of the passengers. The sum of relevant information is called the *domain of action* in [DENN93]. While this multidimensionality of trust could be captured in

[17] In [JOSA05], [JOS205] and [JOSA07]
[18] Term was not used in [GAMB00].
[19] Also known as *bayesian probability*.

multidimensional trust values (see section 3.1.2), this approach can be impractical when entities exchange trust-related information[20]. For that purpose, we define another function type:

(3.1.3D) Definition of a context-sensitive trust function:
A context-sensitive trust function over a set of entities E, a set of domains D and a trust space T is a function

$$t_{ctx}: E \times E \times D \rightarrow T$$

Humans often distinguish another dimension of trust: There is trust in the ability to perform an action, and trust in willingness to perform it (see Figure 3.1.3-I). Using only the definition in (3.1.3C), there should be no difference: the decision making entity only cares about observable behavior, not the intention behind it. However, in a social context, this distinction manifests itself in the assignment of blame. This is not without reason: Unlike lack of ability, which is heavily dependent on the discipline, intentional uncooperative or even malicious behavior typically crosses domains of action. Therefore, it is sensible to generally distrust someone who intentionally harms us, but this does not apply if they do so by accident. Nevertheless, the detection of malicious entities is a different task than that of assessing the certainty mentioned in (3.1.3C), so we will not include this distinction in our model. It is important to not confuse trustworthiness with benevolence, or likewise untrustworthiness with hostility[21].

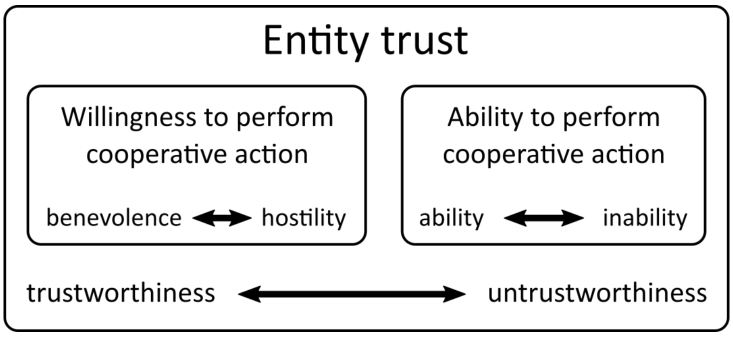


Figure 3.1.3-I: Components of entity trust

3.1.4 Trust in Objects

We just investigated trust in active entities or agents. Is it also possible to trust inanimate objects? Evidently, we can not apply our previous definitions, because objects can neither cooperate with agents, nor can they put trust in them[22]. When we say that we trust or distrust an object, we usually express an assumption about its functionality. Not trusting

[20] In the example: An airline interviewing a pilot will not ask them about their cooking skills, so the transfer of trust related information does not include all domains of action.
[21] In this chapter, only (un-)trustworthiness is relevant. We will briefly discuss the domain-specific detection of malicious entities in chapter 4.
[22] Because of the latter, we can not include them in the set of homogenous entities.

food means not relying on it being safe for consumption; not trusting a tool or machine means supposing that it is defective. This also occurs in software: When a developer does not check for fault conditions of a utilized library, they implicitly assume that it is functional, i.e., they trust it. When a user installs software on their PC, there is also trust involved: trust in the program not being a computer virus.
One might simply treat trust in these examples as shifted entity trust towards the developer, or more general, the object's producer. We find the clearest expression of this shifted trust in the handling of messages. When we receive a message and trust it, we trust its originator that the content of the message is correct, or at least that they did not intentionally lie to us. Thus, one might argue that entity trust is sufficient for the evaluation of messages. But a message is not equivalent to "direct", e.g., verbal, communication. Equating the reception of a message with the observation of behavior relevant for (3.1.3C) would neglect several factors that might have affected the message. To trust a message, we have to rely on its authenticity. Some deliverer of the message could have altered its contents, so we also have to consider entity trust towards all middlemen. Another possibility is that the assumed originator did not send any message at all and someone is impersonating them. There is not always an entity whom we could trust to prevent this; for example, post offices usually do not demand proof of identity for sending mail. [MASH11] introduces the notion of *data trust* for dealing with suchlike concerns. Summing up its properties in the context of the paper, we give the following definition of data trust:

(3.1.4A) Definition of data trust:
Data trust is the certainty with which a decision making entity assumes that a message carries information that will be consistent with its perception, without knowledge of this information.

Contrary to our definition, [MASH11] suggests generating data trust values based on a number of factors which depend on knowledge of the message's contents, such as opinions from other entities and spatial proximity of these opinions to the described event[23]. Some other systems[24] also validate a message based on its contents. We find this approach problematic, because it mixes two different concepts: data trust and *data truth*[25]. Suppose someone implemented an agent that relies on received messages and sensory data. The information flow in the agent's reasoning component might look as illustrated in Figure 3.1.4-I.

[23] See [MASH11] p. 5 section C.
[24] [GOLL04], [MINH10]
[25] Meaning its accuracy in describing the "real world" (correspondence theory of truth).

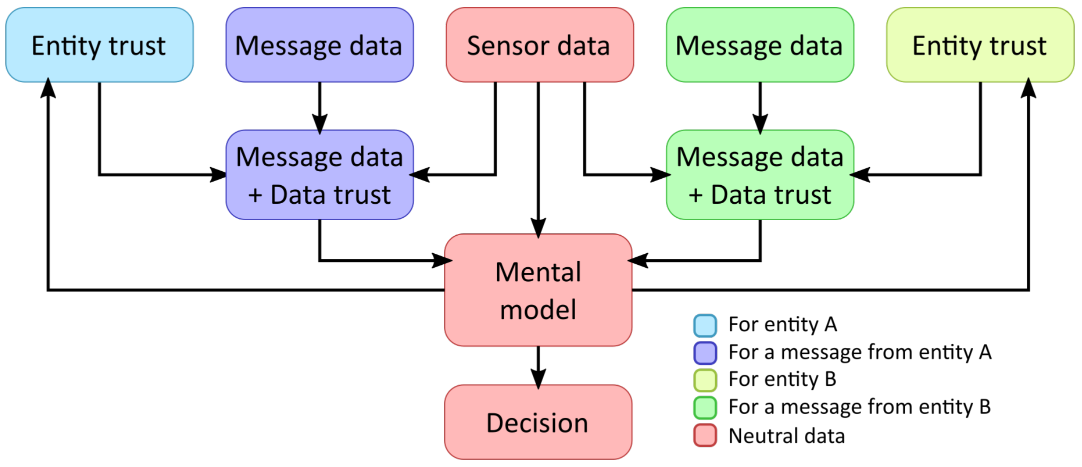


Figure 3.1.4-I: Simple data trust model

First, the agent derives data trust/validity from entity trust in the sender[26], as well as from the message's contents and sensory data. Now, it uses all available data, including sensory data[27], to generate a model of its environment (*mental model*), in order to plan its actions and make a decision. The agent also updates entity trust according to data trust[28] ("I trust you because you gave me useful information" / "I distrust you because you lied to me").
In this scenario, the agent unnecessarily interprets the sensory data multiple times, implicitly generating multiple models that might not be consistent, while potentially giving more weight to the sensory data than it should. The verisimilitude of the message data should be derived from the agent's single mental model, not the other way around.
Therefore, we suggest an information flow as shown in Figure 3.1.4-II. Before interpreting it, each message is attributed with the corresponding data trust according to (3.1.4A). This data trust can be derived from the corresponding entity trust, combined with additional useful information as, for instance, a measure of the message's cryptographic security. After having derived a single model of its environment, the agent compares it with message data to calculate a measure of data truth[29]. According to this data truth, the agent updates its entity trust regarding the sender.
Data trust should be independent of message content, while data truth should only be dependent on its content (and the agent's conception of reality[30]).

[26] This approach is also found in [MASH11].
[27] Which could also contain information that did not affect data trust, but does affect the model.
[28] Also conceived in [MASH11].
[29] One could choose a coherence theoretical interpretation of truth in such a system. All sources of perception that could be used for retrospective judgment are also considered during planning, via the single model. There is no "more objective" reality that the agent could reason about.
[30] Keen observers might have noticed that data truth in Figure 3.1.4-II partially depends on data trust and entity trust for the same message after all. Should "Entity A sent this message" be a part of said "conception of reality"? An agent could filter out the effect of data trust on data truth when updating entity trust, to prevent feedback effects (cf. the feedback filters in 3.1.4-II).

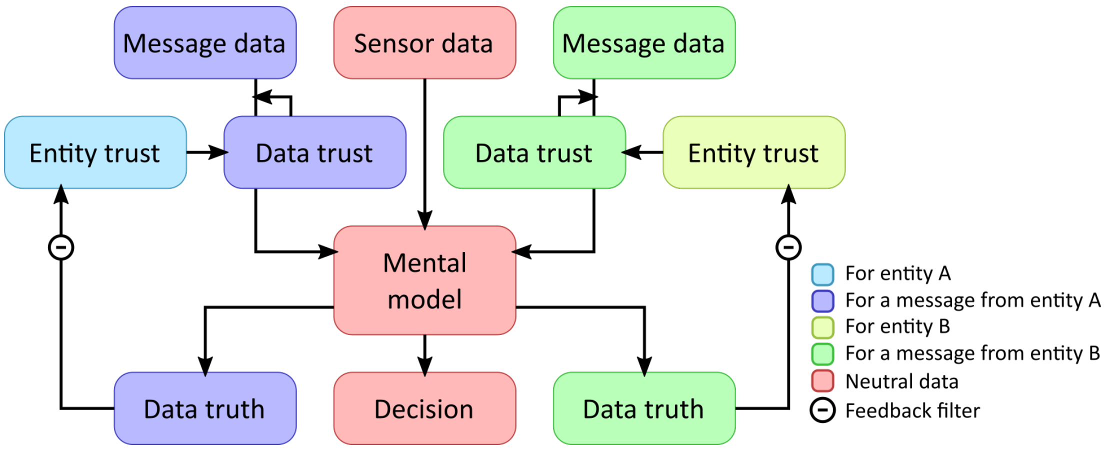


Figure 3.1.4-II: Improved data trust model

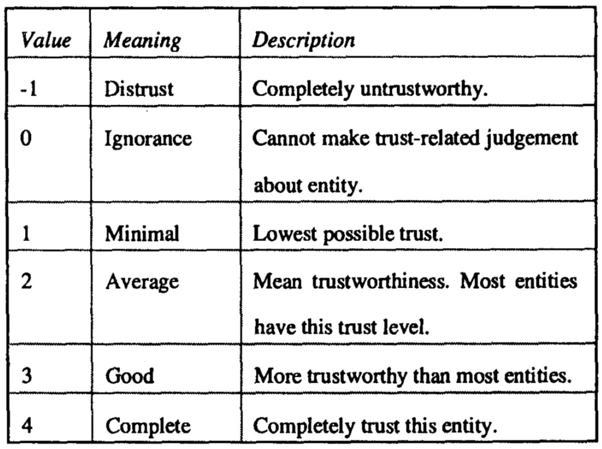
We already mentioned cryptographic security. An important source of data trust in a message is trust in its authenticity. Humans have been employing physical authentication techniques for that purpose, such as seals and signatures. Their digital equivalent is constituted by asymmetric cryptography. By securely binding keys to identities and using message authentication codes, software agents can reach a level of security at which trust in authenticity becomes virtually binary. However, this comes at the cost of performance, so an agent might choose weaker cryptographic techniques at peak times.

3.1.5 Trust as a Computational Concept

Because of the advantages of reasoning about trust for cooperative endeavors, multi-agent systems (MAS) should be a promising area of application for software based on entity trust. How can we enable software agents to reason about entity trust? To possess knowledge about entity trust, an agent needs to be capable of partially evaluating the entity trust function. Domain and codomain of the entity trust function need a computational representation to become data types.

The first two parameters of our trust function are entities. To represent them, one could simply assign unique identifiers to them. While this allows for easy computation, additional efforts must be made to ensure their uniqueness. In the simplest case, the first parameter is constant and represents the reasoning agent itself; the trust function then measures the agent's own trust in other agents. More sophisticated agents might also consider a third party's trust in another agent. If the system uses a context-sensitive trust function, the domain of action must also be represented. Provided that a finite number of such domains are considered, an enumeration is sufficient. Lastly, there is the codomain of the trust function, the trust space T. There exist various approaches for representing trust values.

First of all, one might regard trust as an ordinal variable within a fixed domain of action. An agent could then rank its peers according to its measured trust in them, in order to choose who to cooperate with. [ABDU97], [ZHAO04] and [SURY06] arrange various degrees of entity trust on a small ordinal scale (see Figure 3.1.5-I). This is particularly useful when humans provide the trust information, because the scale can then be mapped to a small set of discrete verbal statements which humans can agree or disagree on [JOSA07].



Value	*Meaning*	*Description*
-1	Distrust	Completely untrustworthy.
0	Ignorance	Cannot make trust-related judgement about entity.
1	Minimal	Lowest possible trust.
2	Average	Mean trustworthiness. Most entities have this trust level.
3	Good	More trustworthy than most entities.
4	Complete	Completely trust this entity.

Figure 3.1.5-I: Possible enumeration of direct trust values and semantics. Source: [ABDU97]

A major disadvantage of small[31] scales is the representation of minor changes in trust. When an agent observes cooperative behavior, it should naturally memorize this information. Frequent collaborators should be assigned higher trust values. However, if the observation does not justify raising the trust level by a full step on the ladder, it must either be ignored (defeating the purpose) or stored elsewhere, for example in the form of a counter. The latter case would be equivalent to using a larger, more finely graduated scale, with thresholds corresponding to values on the smaller scale (see Figure 3.1.5-II). As a result, small discrete scales alone are only practical when trust is either static or subjectively defined by humans, as is the case with PGP key trust.

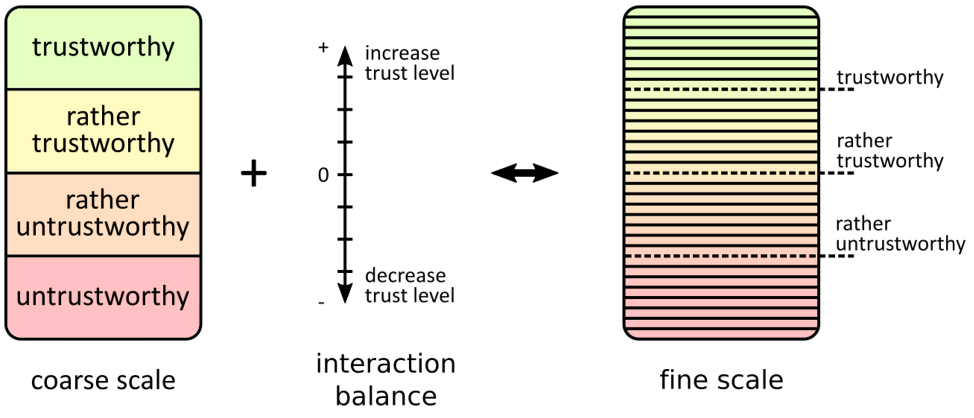


Figure 3.1.5-II: Moving on a coarse trust scale according to interaction balance is equivalent to moving on a fine scale.

[31] In terms of cardinality, not interval size.

One-dimensional continuous trust values such as those conceived in [MARS94], [GAMB00], [GERL07], [MINH10], [BISM12] and [WEIZ14] allow for subtle trust adjustments. They can be mapped to floating point numbers and are frequently interpreted as bayesian probabilities. As indicated in section 3.1.3, problems can arise when assigning meanings to the extreme values. Suppose we (as an agent) encounter a previously unknown agent, and have no categorical knowledge about unknown agents[32]. Of course, we could not be less certain about its intentions or abilities. This profoundly differs from situations where we strongly expect uncooperative behavior from our peer, and can be regarded as a third extreme value (besides complete trust and complete distrust). Jøsang[33] addresses this issue using a belief calculus, namely subjective logic (see Figure 3.1.5-III for an introduction to subjective logic).

Subjective logic [JOSA01] formalizes subjective beliefs as 4-tuple opinions. Let x be the statement "Entity B is trustworthy" within Entity A's mental model. We can denote this as the opinion $\omega_x^A = (b_x^A, d_x^A, u_x^A, 0.5)$[34], where b_x^A, d_x^A and u_x^A represent A's belief, disbelief and uncertainty regarding x, respectively. This can also be written as ω_B^A, and regarded as the entity trust of A in B. The trust space would then be identified with the opinion space, which can be mapped to the interior of an equal sided triangle using the barycentric coordinate $B((b_Y^X, d_Y^X, u_Y^X, a_Y^X) = (b_Y^X, d_Y^X, u_Y^X)$. Now all three extreme values are represented in the corners of the triangle, as the both absolute opinions and the point of absolute uncertainty. Subjective logic might be the ideal foundation for our system, since it is accompanied by a rich mathematical framework including operators for computing the effects of recommendations and derivation of trust/opinions from multiple sources.

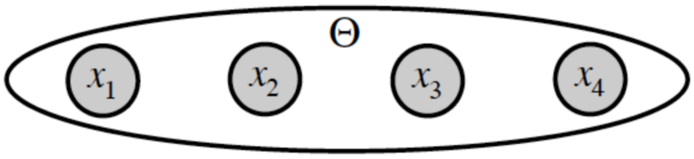
Some other approaches ([MANC98] and [SABA02]) use fuzzy logic for reasoning with trust values. While subjective logic operates on uncertain measures about crisp propositions, fuzzy logic operates on crisp measures about linguistically vague (and therefore "uncertain") propositions (cf. [JOSA01] p. 2). A fuzzy-logical representation of entity trust might be more suitable for our definition in (3.1.3C), because certainty (the measure) is relatively easy to describe accurately, for example by putting it into context with risk[35], whereas (non-binary) cooperative behavior (part of the proposition) is hard to give an linguistically clear account of.

[32] I.e., no statement like "30% of all agents in this universe will always act cooperative."

[33] For example, in [JOS205] pp 64-67

[34] The fourth element represents *relative atomicity*, the size of the state space relative to the frame of discernment. In this example, the frame of discernment is binary, so $a_x^A = 1/2$.

[35] This is essentially being realized in [MANC98].

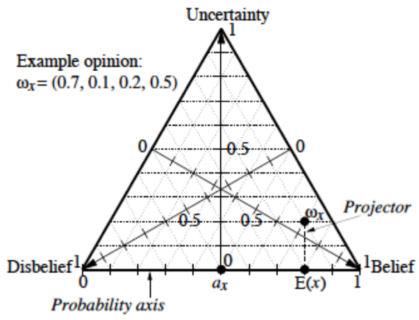


Frame of discernment Θ containing 4 elementary states (possible situations) [JOSA01]

Only one elementary state can be true at a time. States are described using sets and can be grouped into superstates (supersets). Superstates are true iff one of their substates is true; in the picture above, Θ would always be true. A subject can distribute a total belief mass of 1 over the set of all possible superstates (= 2^{Θ}).
In a frame of discernment with two disjoint states x and $\neg x$ that contain all substates of the frame, $b(x)$ is the mass assigned to x (belief), $d(x) = b(\neg x)$ the mass assigned to states disjoint from x (disbelief), and $u(x)$ the mass assigned to all other states, i.e., Θ in a binary frame of discernment (uncertainty).
It holds that (Belief function additively):

$$b(x) + d(x) + u(x) = 1$$

Combined with the relative atomicity $a(x)$ (the proportion of elementary states contained in x), these masses form the opinion $\omega_x \equiv (b(x), d(x), u(x), a(x))$. The opinion space can be visualized as a triangle:



A subjective *probability expectation* can be obtained by projection an opinion point onto the probability axis.

Figure 3.1.5-III: Introduction to Subjective Logic

All approaches that introduce multidimensionality into context-free trust spaces[36] inevitably complicate decision-making, because the trust space loses a meaningful total order. Since both certainty and display of cooperative behavior[37] are at least ordinal variables, it would be possible to preorder the space according to one of them, and then order the equivalence classes according to the other. Such orderings based on prioritized criteria have been suggested[38], but they are hard to defend when related to intuitive choice. Should we rather trust someone from whom we uncertainly expect cooperative behavior than someone from whom we certainly expect minimally less cooperative behavior? In real-world situations, when a multitude of factors must be considered, it becomes unlikely that two options are equal in the first criterion at all, effectively rendering the second criterion as well as the second variable meaningless and the trust space one-dimensional[39].

When representing context-dependent trust values with domains of action that are independent of each other, vectors of context-free trust values are ideal. When reasoning within a domain of action, the corresponding trust value can be extracted through vector access, and a total ordering of context-dependent trust values should not be necessary.

Apparently, there is a multitude of concepts regarding the computational representation of entity trust. The ideal choice for a system heavily depends on the requirements regarding complexity and efficiency. Such being the case, we will attempt to make all schemes designed in this study independent of the underlying trust spaces and not specify types of trust values. Instead, we will focus on general trust management (see section 3.2).

Because data trust is usually derived statically from a small number of sources, e.g. entity trust in the originator and a measure of cryptographic strength, it could also be represented by simply linking the base measures from each source (e.g. "This message was sent from a trustworthy agent along with a strong MAC). These combined propositions can also be used for communicating data trust, leaving the interpretation up to the recipient and reducing information loss. The information carried by such a message is not identical with data trust as defined in (3.1.4A), but it becomes data trust when combined with an interpretation procedure in the recipient's reasoning component.

3.1.6 Acting upon Trust / The Ideal Agent

An agent usually pursues a strategy or follows a protocol when interacting with other agents. This guideline is either hardcoded in its programming, or derived by an artificial intelligence. It serves the purpose of accomplishing specific objectives in the short or long term. Perhaps the best-known basic strategies are egoism and its complement altruism. The aim of an egoistic agent is to maximize its total benefit, the so-called *utility* for itself.

[36] Those which are limited to a single domain of action from the codomain of a context-sensitive trust function, and thus do not have an intrinsic multidimensionality from context sensitivity.

[37] Or other degrees of freedom considered in the respective approach.

[38] For example, in [JOS205] p. 65, which stems from definition 10 in [JOSA01] p. 9.

[39] At least within any reasoning that is based on the ordering.

Thus, it always acts in self-interest. An altruistic agent, by contrast, aims to maximize the utility for all other agents, not including itself; it always acts in the interest of others. Relating these strategies to cooperative behavior, we notice intrinsic problems in both.
On the one hand, if we built a system that only contained egoistic agents, situations might arise where the utility for one agent, which results from not cooperating, is just slightly greater than that received by each agent individually (including the first agent), which results from cooperating. This agent would then refuse to cooperate, vastly decreasing total utility. The risk for such scenarios naturally increases with the number of agents involved in a cooperative endeavor, because the total utility is distributed over all agents, whereas a saboteur might claim the (lower) total utility[40] resulting from a failed endeavor for itself, with potentially dangerous results for the system. For instance, the formation of an emergency corridor for ambulance vehicles depends on the cooperation of hundreds of agents[41]. A purely egoistic agent would abuse this for overtaking the other vehicles with the side-effect of obstructing the ambulance's way.
On the other hand, if the system only contained altruistic agents, they would be obliged to help others and might also do so in situations where this reduces total utility for all, because their own loss is greater than the sum of the other's gains. With altruistic strategies, the risk for adverse effects increases when the number of agents involved decreases. When there is more interaction, the chance for cooperative bonuses rises, and the agent is likely to be compensated. An agent that relies on potential compensation, but only acts altruistic when the expected compensation is greater than the initial loss, is acting *reciprocally altruistic*. An ethical axiom that such a strategy could be based on is the *golden rule*:

(3.1.6A) "Do unto others as you would have them do unto you." ([PUKA15])

Considering dependencies between own reasoning and the reasoning of others[42] can provide solutions in situations where ubiquity of egoism or altruism has adverse effects: If two naïve egoistic vehicular agents meet at an intersection, none of them will give way, resulting in a collision. If two altruistic agents meet, both will give way, resulting in a stalemate. By reasoning about the respective other's reasoning, both cars conclude not just that "one driving while the other gives way" is the correct solution, but also that the other car is aware of this solution. Without this information, nobody could safely drive[43].
As pointed out, both egoistic and altruistic strategies result in suboptimal agent behavior with regard to the system. This is because their aim does not necessarily include optimizing

[40] We refer to this difference as the *cooperative bonus*. The cooperative bonus is lost in this scenario.
[41] Currently human drivers, but the existence of this paper is justified by their possible future (at least partial) replacement.
[42] This is not a causal dependency; it results from the entities' homogeneity. Their awareness of this homogeneity would be essential categorical knowledge in this scenario.
[43] This is still insufficient for deriving each agent's behavior: They both discard the options where both agents would be doing the same, but they still do not know which one should give way. We will discuss this later in this section.

the system's efficiency. If we equate the system's efficiency with the total additive utility for all agents, we arrive at yet another strategy, namely utilitarianism.
According to one of the most influential proponents of classical utilitarianism, Jeremy Bentham,

(3.1.6B) "it is the greatest happiness of the greatest number that is the measure of right and wrong." ([BENT76])

In Bentham's theory, happiness is a measure for utility; in a multi-agent-system, it could be efficiency of the system, depending on the system's purpose. Unlike an egoistic agent, which lacks consideration of the other agents' total benefit, and an altruistic agent, which lacks consideration of its own benefit, a utilitarian agent aims to maximize utility for all agents (see Figure 3.1.6-I).

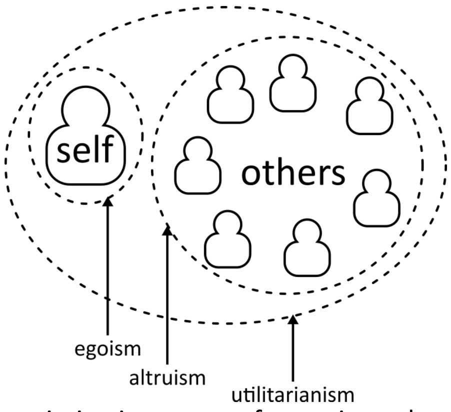


Figure 3.1.6-I: Utility maximization targets for egoism, altruism and utilitarianism

Naïve utilitarian agents do not encounter the problems we described while criticizing egoism and altruism, but they still can not solve the intersection problem without considering behavior dependencies. In order to solve all of the aforementioned problems, we should attempt to combine utilitarian strategies with the advantages of the golden rule.
First of all, the golden rule only considers the behavior of interaction partners. Utilitarianism extends the perspective to the collectivity of agents, so we should in like manner extend the scope of the golden rule[44]. There is an ethical axiom that fits these requirements:

(3.1.6C) Categorical imperative[45]:
"Act only according to that maxim whereby you can, at the same time, will that it should become a universal law." ([KANT85] p. 30)

[44] This eliminates its interactive character, because the affected agent does not interact with all other agents.
[45] This is Kant's first formulation, also called the "universalizability principle".

An agent whose strategy conforms to the categorical imperative can not chose any behavior that would be detrimental to itself if other agents chose the same behavior. It would act optimal, regarding its own aims, in a system entirely composed of homogeneous entities. However, the categorical imperative in itself is not utilitarian. For heterogeneous agents, the same consequence might produce different utilities; the agent might choose a behavior that would also benefit itself when universalized, but has adverse effects for the system. The following definition combines the ideas of (3.1.6B) and (3.1.6C):

(3.1.6D) Categorical utilitarian imperative:
Act only according to that maxim, which would maximize global utility when established as a universal law.

A system composed entirely of agents that follow (3.1.6D) would operate at maximum efficiency per definition, unless there are situations where the best solution can not be expressed as a universal law[46]. How would two agents solve the intersection problem? The problem's difficulty is rooted in the solution being asymmetrical, while the situation is symmetrical in the agents' perception. Both options ("A gives way" and "B gives way") will receive equal valuation[47]. Some kind of tie-breaking is necessary.
Firstly, one could introduce non-determinism. The agents could let some source of randomness influence their behavior. Of course, there would be additional communication necessary to prevent a collision, but a solution is theoretically possible. Unfortunately, the agents' actual behavior would be completely unpredictable for third entities. Additionally, the system can easily be cheated when agents have to rely on values produced by other agents being truly random.
Secondly, it could occur to us that the agents' perceptions in this scenario are actually not exactly identical, because of chirality. An agent could use this to derive a solution if it could rely on the other agent interpreting the chirality in the same way. This can be achieved by introducing common knowledge (rules) that all agents need to possess. For the intersection problem, "priority to the right" would be such a *tie-breaking rule* that is also based on chirality[48]. A peculiarity of these rules is that they can not be totally justified; priority to the right (combined with driving on the right) is not inherently better than priority to the left (combined with driving on the left). Independent intelligent agents could not derive the same rule on their own. An agent's maxim should therefore include: "In case of doubt, stick to the rules.". At this point, doubt implies perfectly equal valuation for all alternatives. This would render most traffic regulations obsolete[49], because the agent could derive them on

[46] The law can be parameterized so that the resulting behavior depends on perception. It does not force homogenous agents to act the same way, unless they share the same perception.
[47] And they should, because otherwise naïve deterministic agents acting only according to (3.1.6D) could collide when both make the same "better" choice.
[48] Such a rule does not to be parameterized based on chirality, it can depend on a factor that would otherwise not affect valuation.
[49] For instance, safety distances are not tie-breaking, their necessity results from the system's physical properties.

its own, or find a better solution for specific situations. This is only desirable under the premise that the system only contains infinitely intelligent omniscient agents. If we do not wish to rely on that, we could increase the margin of doubt, reducing the agents' relative self-confidence. Maximizing this margin of doubt would mean minimizing the agents' relative self-confidence, resulting in them strictly following the rules.

Up to now, the system we discussed in this section relies on all agents being designed by the system designer and being free of defects and manipulations. In reality, at least the latter two requirements can not be achieved. If an agent, due to a bug or manipulation, follows an egoistic strategy, it could attempt to disrupt the system for its own benefit. We call such a behavior an *insider attack*. In the intersection example, an egoistic saboteur that doesn't communicate could hardly gain an advantage; driving instead of giving way is very risky for themselves. A more promising insider attack, the *illusion attack*[50] is realizable when agents give account of their environment; a malicious (or only misbehaving) agent can spread incorrect information (the illusion). For example, a malicious vehicular node in a VANET might wrongfully announce the approach of an emergency vehicle, in order to get other vehicles out of the way and thus gain a speed advantage. To defend against insider attacks, agents have to not treat all other agents equally, and to recognize egoists, or more general, untrustworthy entities. They have to consider their own certainty in assuming that another entity shows cooperative behavior, which is entity trust[51], according to (3.1.3C).

To sum up, we identify the following requirements for a MAS subject to insider attacks[52]:

(3.1.6E) Requirements for a MAS subject to insider attacks:

(1) Agents should behave in accordance with the categorical utilitarian imperative defined in (3.1.6D).

(2) Agents should calculate and keep track of entity trust. They should interpret it within their decision making process (i.e. They should *act upon trust*).

(3) The system should specify a set of rules whose compliance does not conflict with rule (1). These rules do not need any further justification. Agents should prefer the behavior stipulated in the rules to similar alternatives.

Due to the nature of the system, not all agents will conform to these requirements. In addition, (3.1.6E1) is hard to achieve in complex systems, even for agents designed by the system designer. Nevertheless, we can utilize the concept of an entity that conforms to all of these requirements; we call such an entity an *ideal agent*.

What does it mean to act upon trust? In situations similar to the (symmetric 2x2) prisoner's dilemma ([KHUN14]), cooperative behavior is dangerous when one can not rely on the opponent's cooperation. Acting upon trust would imply cooperating when entity trust is high, and refusing to cooperate when entity trust is low. This can be extended to series of interactions, and also to data trust which is derived from entity trust. Providing accurate

[50] An explication for VANETs is given in [ALKA12] p. 5.

[51] In the case of illusion attacks, the determining factor is more precisely data trust.

[52] We will refer to such a system as an *open multi-agent system*. This corresponds to the definition given in [ARTI09] p.461.

information and trusting each other's messages is definitely beneficial for both parties in the long term, and can be regarded as cooperation. However, when entity trust (and hence also data trust) is low, an agent should doubt of the messages' validity, to avoid falling victim to an illusion attack. The impact of entity trust naturally depends on the situation. In the prisoner's dilemma, the trust necessary for choosing to stay silent grows with the level of penalties received if the other player should confess.

It is this amount of trust that we referred to as "certainty [to assume] cooperative behavior" in (3.1.3C). It can be measured based on risk. The more certain we are in assuming something, the higher are the risks we will take in regard to being wrong. Certainty in a proposition is not equivalent with willingness to rely on it. The latter is influenced by our valuation of the risks involved, while the former can be determined a priori. The relation between risk and trust is important and will be used as a general guideline to determine when entity or data trust should increase or decrease.

[MANC98], instead of formalizing trust directly, describes an entity's trust model using risk-trust matrices of trust-related variables. By defining regions of trustworthiness, the entity can make decision based on fuzzy-logical inference rules (see Figure3.1.6-II). This approach could also be useful for the implementation of trust-based decision making when trust is directly represented. Agents in an MAS that includes the communication of trust values could adapt their interpretation of these trust values by "learning" fuzzy trust zones in matrices where one variable is a trust value.

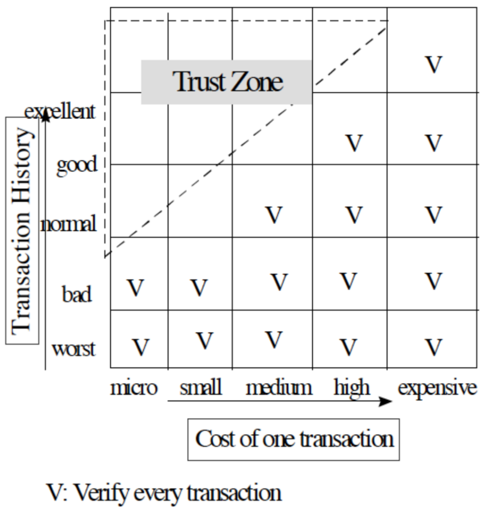


Figure 3.1.6-II: Trust Matrix between "Cost of Transaction" and "Transaction History". Source: [MANC98]

3.1.7 Trust Transitivity

Many papers[53] mention transitive, or at least partially transitive, trust. First of all, it needs to be mentioned that the common understanding of a trust relationship contradicts mathematical transitivity. Transitivity is a property of binary relations, and any interpretation of trust that goes beyond simple statements such as (3.1.2A) can not be expressed as a binary relation over entities. The notion of partially transitive trust usually combines two observations: Firstly, that trust can be propagated along chains (also called *trust paths*), possibly being weakened or diluted in the process ([JOSA07] p. 9). And secondly, that trust is *conditionally transitive* ([ABDU97] p.52); i.e. that its propagation is subject to certain semantic constraints. These constraints have been investigated in [JOS205]. One important constraint is induced by the diversity of domains of actions[54]: that Alice trusts Bob to fix her car, and Bob trusts Claire to look after his child, does not imply that Alice trusts Claire to fix her car or to look after her child ([JOS205] p. 60). Trust purpose, or the corresponding attribute of a domain of action, should be homogeneous within a trust path. Moreover, Alice trusting Bob to look after her child would also not suffice to derive that Alice trusts Claire for this purpose. Instead, Alice needs to put trust in Bob's ability to *recommend* a babysitter. This trust in the ability to refer to a third party is called *referral trust* in [JOS205] and enables the definition of another constraint for trust propagation:

(3.1.7A) "A valid transitive trust path requires that the last edge in the path represents functional trust and that all other edges in the path represent referral trust […]" ([JOS205] p. 61)

Figure 3.1.7-I illustrates this kind of transitive trust derivation along paths. We come to the conclusion that even this elaboration is not specific enough: (3.1.7A) implies that except for the last edge, all edges in the trust path represent the same kind of trust; however, the second-last edge represents trust in the ability to refer to a functional expert, while the third-last edge represents trust in the ability to refer to another referrer.

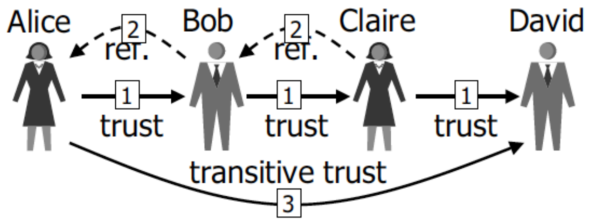


Figure 3.1.7-I: Transitive trust derivation. Source: [JOS205]

For a trust path with only two edges, this distinction is irrelevant. Therefore, when an entity A trusts another entity B, B trusts C, and the aforementioned constraints are met, can we derive that this affects A's trust in C? We need to consider the model in which this

[53] For example, [ABDU97], [RANG98], [KAMV03], [JOS205], [JOSA07] and [MASH11].

[54] [JOS205] uses the term *trust purpose*, which can be regarded as equivalent to "domain of action" in this context (but not later, after the introduction of referral trust).

deduction should be made. In an objective model, i.e., an omniscient observer's knowledge about this scenario (see Figure 3.1.7-II), these statements are not sufficient to make the deduction. It could be possible that A is not aware of B trusting C, or does not even know C at all.

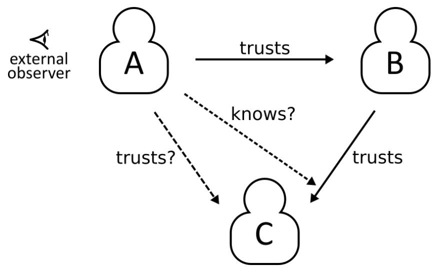


Figure 3.1.7-II: Objective model of trust transitivity

In a subjective, epistemic model, e.g., from the viewpoint of A (see Figure 3.1.7-III), A's awareness of B trusting C is not a necessary extension, because it is implied by the very existence of this trust relationship in the model. However, "B trusts C" is not a legal statement in this model. Trust is subjective by nature, so A can never definitely know if B trusts C. A can only indirectly measure this trust, for example by observing B's behavior, or by receiving a recommendation from B.

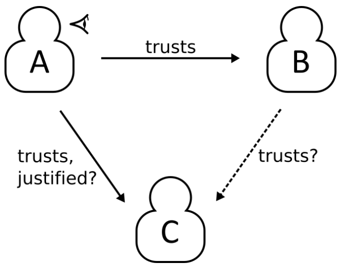


Figure 3.1.7-III: Subjective model of trust transitivity

The above considerations deal with trust in the ability to cooperate, which is only a component of our concept of entity trust. If an entity wants to rely on another's cooperation, it must also take the other's willingness to cooperate into account. One can easily see that this component of entity trust can not always be propagated. If B trusts C, it means that B assumes that C will cooperate with B. A can not infer that C will also cooperate with A, unless it assumes that C's willingness to cooperate results from its attitude towards a category of agents that includes B and A. A could also draw other abductive inferences from a recommendation; for instance, that B, generally being trustworthy, is being deluded by the untrustworthy entity C.

An example situation where trust appears to be transitive is found when investigating access rights. For example, it is normal human behavior to not allow strangers to enter one's home. By inviting people, we put ourselves at risk; our property could be damaged or stolen. An invitation therefore implies a risk tolerance based on the assumption that the

invitee cooperates, and can be regarded as an expression of entity trust. We can allow invitees to bring other people, so one might argue that we also put trust in those other people by transitivity. However, we can not put trust in someone without knowing of their existence. It is at that point in time when we learn of their existence, that we also learn that the person in question was brought along by an invitee, who implicitly recommended the person as trustworthy. This modification of entity trust, according to what an agent learned about a second agent's trust in a third agent, is trust propagation.
Intuitively, it would seem that trust propagation should always increase trust. If A learns that B, whom it trusts a lot, puts some trust in C, this should increase A's trust in C. This is only correct in combination with the implicit assumption that A would have not trusted C without this knowledge. If A initially trusts C a lot, and then learns that B only trusts C a little, this should *decrease* A's trust in C. It becomes evident that the effects of trust propagation not only depend on the trust relations that are edges of the trust path, but also on the trustor's initial trust in the trustee. For combining different sources of trust, [JOS205] offers the model of parallel trust combination (see Figure 3.1.7-IV).

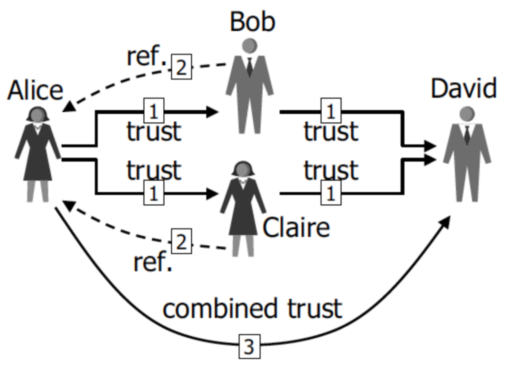


Figure 3.1.7-IV: Combination of parallel trust paths. Source: [JOS205]

The connector $\diamond$ is used to denote the combination of parallel trust paths[55] and, for a trust model based on subjective logic, the *consensus operator* $\otimes$ can be used to compute the resulting trust value. Unfortunately, this is only sufficient for directly deriving A's trust in C if we can represent A's "initial" trust in C as a combination of other trust paths of lengths ≥ 2. If this is not the case, for example if A has also derived its trust in C from previous interactions, the parallel trust combination becomes a fixpoint "equation":

(3.1.7B) $(A,C) \equiv ((A,B) : (B,C)) \diamond (A,C)$

Solving this iteratively[56] (starting with A's initial trust in C) should result in a trust value that reflects a "fair" combination of the initial trust and the trust path. Another option for

[55] Trust paths are denoted as a concatenation of their edges. A trust path from entity X over Y to Z would be denoted as $(X,Y) : (Y,Z)$. (cf. [JOS205] p. 61)
[56] This requires a modified interpretation of the connector. In [JOS205], duplicate combination with $\otimes$ would equal adding a second identical opinion. Iterated application is a contraction, so it would

solving the equation would be for A to alter its trust in B. This seems sensible; if someone tells us to distrust a trusted person, this should also prompt us to reduce trust in the recommender. When the fixpoint equation holds, a further recommendation would have no effect, which should be the objective of trust propagation. This does not necessarily require that A has actually reasoned about B's trust in C. The condition could also be reached "coincidentally", or by a fourth entity altering trust values in order to fulfil it.
Perhaps a more apt term than trust transitivity would be *trust coherency*. We provide the following definitions:

(3.1.7C) Definitions of subjective and objective trust coherency:
Let $\mathcal{P}(X,Y,V)$ denote the set of all simple paths from X to Y within the complete directed graph with vertex set V. For two trust sources p,q for entity X regarding entity Y (for example trust paths from X to Y), let $p \diamond q$ denote the combined trust source for X. Two trust sources p,q are equivalent ($p \equiv q$) iff the trustor would derive the same trust value from either of them. The trust network defined by the trust function t over a set of entities E is *subjectively coherent* for an entity $A \in E$ iff $\forall B \in E \setminus \{A\} : \forall p \in \mathcal{P}(A,B,E) \diamond (A,B) \equiv (A,B) \diamond p$. A trust network is *objectively coherent* iff it is subjectively coherent for all its members.

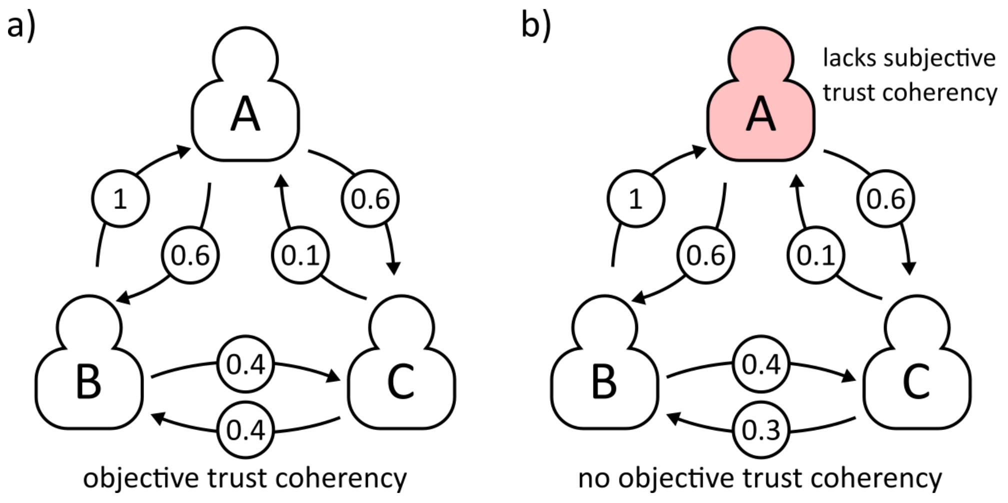

a)
A
B
C
1
0.6
0.6
0.1
0.4
0.4
objective trust coherency
b)
A
B
C
lacks subjective trust coherency
1
0.6
0.6
0.1
0.4
0.3
no objective trust coherency

Figure 3.1.7-V: Objective trust coherency, assuming an interpretation where[57] $x : y \diamond z \equiv z$ iff $trust(x) < 0.5$ or $|trust(y) - trust(z)| \leq 0.2$

If a trust network is objectively coherent (example with generalized trust shown in Figure 3.1.7-V), further trust propagation would have no effect. Trust coherency can be thought of as a constraint satisfaction problem (CSP):

converge according to the Banach fixed-point theorem (the opinion triangle is a metric space); however, the fixpoint would not reflect a "fair" combination. An alternate combination operator could limit the effects of untrustworthy recommendations regardless of their number.

[57] The trust space in this model is $\mathbb{R}$, and $trust\big((P,Q)\big) = t(P,Q)$ for all entities $P, Q \in E$.

(3.1.7D) Definition of the trust coherency problem:
Let $\mathcal{E}(V)$ denote the set of all directed edges and $\mathcal{P}(V)$ denote the set of all simple paths within the complete directed graph with vertex set V. Let $e_k(p)$ denote the k-th directed edge in a trust path p, $d(p)$ denote the directed edge from the first to the last vertex and $\ell(p)$ denote the length, i.e., the number of edges of said trust path. Let $\mathcal{C}_k$ denote the $k+1$-ary relation over a set of entity trust relationships that, for all trust paths p of length k, contains the tuple $\big(d(p), e_1(p), e_2(p), \dots, e_{\ell(p)}(p)\big)$ iff $d(p) \equiv d(p) \diamond p$. The *trust coherency problem* for a trust network, defined by the trust function t over the set of entities E and the trust space T, is the constraint satisfaction problem $\langle X, D, C\rangle$ with the set of variables $X = \mathcal{E}(E)$, the set of domains D which assigns T as the domain to all variables, and the set of constraints $C = \big\{\langle\{d(p), e_1(p), e_2(p), \dots, e_{l(p)}(p)\big\}, \mathcal{C}_{l(p)}\rangle \mid p \in \mathcal{P}(E)\big\}$. The *subjective trust coherency problem* for an entity $A \in E$ is the subproblem including only those constraints induced by a path starting in A, extended with a fixed evaluation of edges that do not start in A (= the trust values not controlled by A.

An example of such a problem is shown in Figure 3.1.7-VI. A trust network is objectively coherent iff its trust function solves its trust coherency problem. If all agents "think rational", $x \equiv x \diamond x$ is a tautology and all binary constraints[58] always hold. Whether higher levels of consistency exist, depends on the entities' interpretations of parallel trust combination. If the trust space contains a value $\mathbb{0}$ that represents complete uncertainty/unreliability, all trust coherency problems possess the trivial solution where all variables evaluate to $\mathbb{0}$, the *trustless state*.

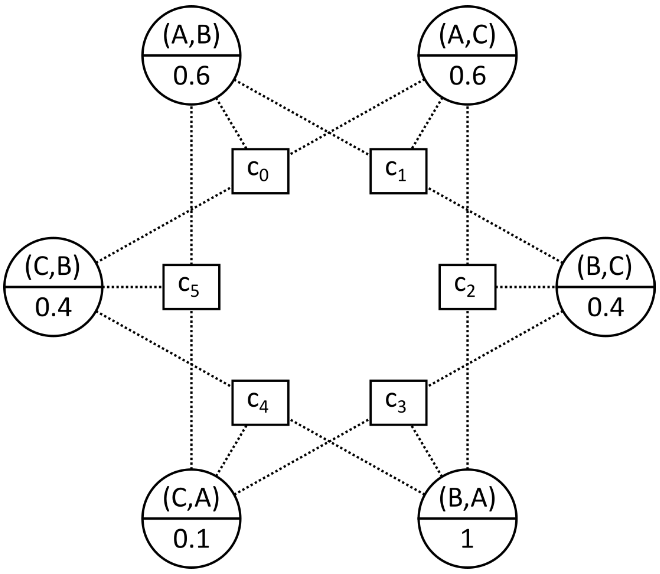


Figure 3.1.7-VI: Evaluated objective trust coherency problem (binary constraints not shown) that corresponds to (a) in Figure 3.1.6-V. The constraints are:

[58] These are basically unary constraints since they only include one variable.

$$\begin{aligned} c_0 &= (A,C) : (C,B) \diamond (A,B) \equiv (A,B) \\ c_1 &= (A,B) : (B,C) \diamond (A,C) \equiv (A,C) \\ c_2 &= (B,A) : (A,C) \diamond (B,C) \equiv (B,C) \\ c_3 &= (B,C) : (C,A) \diamond (B,A) \equiv (B,A) \\ c_4 &= (C,B) : (B,A) \diamond (C,A) \equiv (C,A) \\ c_5 &= (C,A) : (A,B) \diamond (C,B) \equiv (C,B) \end{aligned}$$

For the evaluation in (b), c_0 would not be satisfied.

Objective trust coherency (in a non-trivial solution) is a desirable property of a trust network, from a system designer's perspective; the same applies to subjective trust coherency from an entity's or agent designer's perspective. The arguments regarding the need for trust transitivity that [CHRI13] put forward can be applied here too; essentially, agents need to rely on others to inspect the state of remote elements in the system, but the trustworthiness of these agents can be a remote element itself. Since achieving trust coherency is a desirable objective for everyone, agents naturally cooperate in this endeavor. In order to do so, an agent can increase its subjective trust coherency by adjusting its entity trust in others and subsequently communicating these changes in the form of recommendations, prompting the recipients to optimize their own subjective trust coherency. This process can be referred to as trust propagation and corresponds with CSP solution strategies based on constraint propagation techniques.

Unlike trust transitivity models, the concept of trust coherency can also be applied when the "trust dependency graph", i.e., the subgraph of a trust network that only contains edges which do not evaluate to $\mathbb{0}$[59], contains cycles. Furthermore, each entity can interpret trust transitivity and parallel trust combination with according to a different computation. This only affects those constraints which are exclusive to the entity's own subjective trust coherency problem. Multidimensional, context-dependent and referral trust is also possible, enabling conditional trust transitivity.

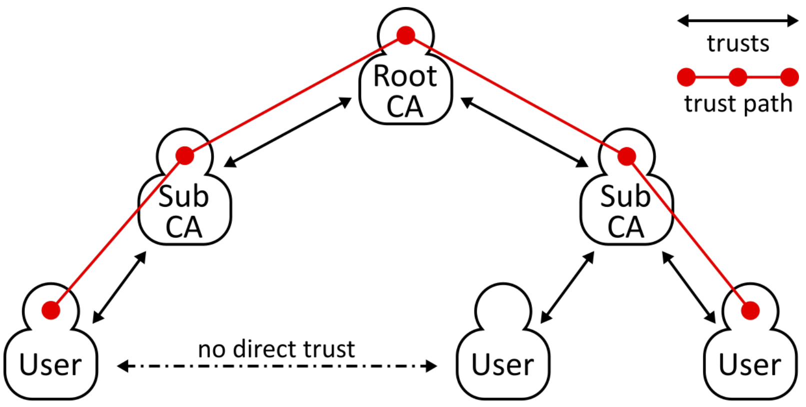


Figure 3.1.7-VII: PKI based on Certification Authorities (CA)

[59] Or any other value that renders the trustee's opinion of third parties irrelevant to the trustor (for instance, if it does not contain referral trust).

When the trust dependency graph is acyclic, it is possible to arrange the entities in a hierarchy where trust paths between two nodes are concatenations of one path upwards and one path downwards (higher nodes do not consider the opinions of lower nodes when computing entity trust in other higher nodes). If the graph is also a tree, simple “partial trust transitivity” can be assumed. Such is the case in a public key infrastructure (PKI) with a root certification authority (see Figure 3.1.7-VII).

Trust coherency does not ensure trust accuracy, because trust does not originate from recommendations alone. When propagating and adjusting trust, entities should always prefer trust values that approximate entity trust generated by direct sources such as experience. This initial trust usually varies over time; entity trust can rise or fall during further interaction, breaking trust coherency. To adapt to these changes and to restore trust coherency, trust propagation needs to be an ongoing process (see Figure 3.1.7-VIII). This is crucial in systems that are based on trust transitivity. For example, in an X.509 PKI [RFC5280], when trust in a certificate is lost, it is usually blacklisted on a certificate revocation list (CRL). This CRL needs to be constantly updated; in other words, local CRLs must be consistent with remote CRLs, and available to all entities at all times, or trust accuracy is lost. According to the CAP theorem (Brewer’s conjecture in [GILB02]), this comes at the cost of partition tolerance. When an agent witnesses malicious behavior of a peer with a certificate that it can not find in the CRL (for example, because it can not contact the CRL provider), the only possible solution that restores trust coherency is to (temporarily) reduce trust in the CA, which also reduces trust in all certificates, thereby incapacitating the agent. Recommendations provide an agent only with a momentary snapshot of the trust network; to ensure accuracy and trust transitivity, recommendations need to be not only veracious, but also faster than a possible attacker.

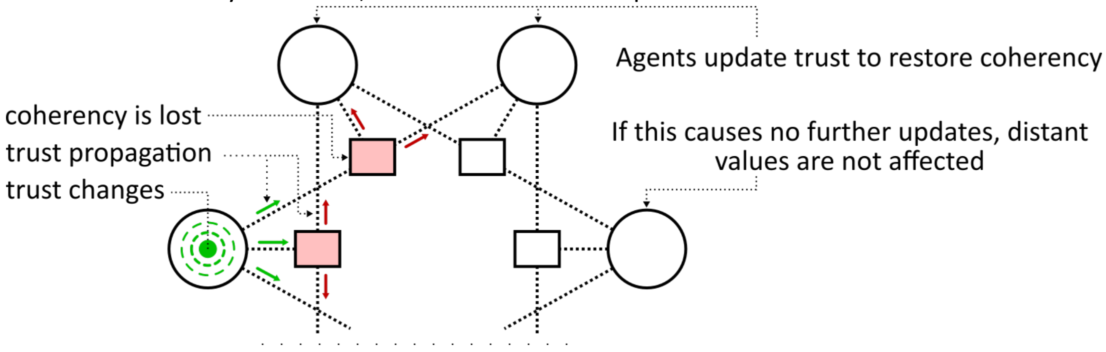

Agents update trust to restore coherency
coherency is lost
trust propagation
trust changes
If this causes no further updates, distant values are not affected

Figure 3.1.7-VIII: Propagation of trust changes (cf. Figure 3.1.7-VI)

To sum up this section: trust is not intrinsically transitive and the term “trust transitivity” is misleading. Instead, one should analyze trust coherency, which is an extrinsic property of a system that depends on its members’ individual trust reasoning. Trust coherency can be achieved by using trust propagation methods such as recommendations. However, a system can only rely on trust transitivity based on trust propagation when it sacrifices partition tolerance.

3.1.8 Dynamic Trust

In the last section, we briefly mentioned that trust is usually not constant over time. An agent uses entity trust to make assumptions about another agent's future behavior; its most evident support for these assumptions lies in its past experience with the other, i.e., in observations. Therefore, trust can be an empirical[60] quantity and is subject to observational error as such. Moreover, its "correct value" can change if an agent spontaneously decides to switch to a different strategy, acting more or less cooperative than before. When reflecting upon interaction, entities are learning and "improving" a trust value in order to make it more accurate. We call this process of inductive learning *trust refinement*, and the corresponding trust source a source of *a posteriori trust*. However, there are also trust sources that are independent of experience. For example, infant mammals instinctively trust their mother. As another example, for future autonomous AIs, it could be useful to define that they may never distrust their human owners. When an AI applies this knowledge (*categorical trust*) to infer that it should obey a specific human's commands, the inference is deductive. Such trust sources are of non-empirical nature and can be denoted as *a priori trust*.

Though a priori trust is usually static, it is not hard to imagine a scenario where it might be dynamic. When a trust based system does not perform well or has to conform to new requirements, we could alter its agents a priori trust values via a software update. This is another variant of trust refinement, but it is not empirical from an agent's perspective, unless the update server itself is modelled as a trustworthy entity. A posteriori trust is inherently dynamic, it can be further distinguished into direct and indirect sources of trust. When an agent makes the decision to cooperate based on entity trust, its cooperation partner's behavior creates a feedback that refines this entity trust (direct trust refinement). Instead of directly observing another agent's behavior, an agent could also observe the consequences of said behavior, for example, in the entity trust produced by third parties (trust propagation), via recommendations or observation of their behavior towards the trustee. An overview of trust refinement variants is shown in Figure 3.1.8-I.

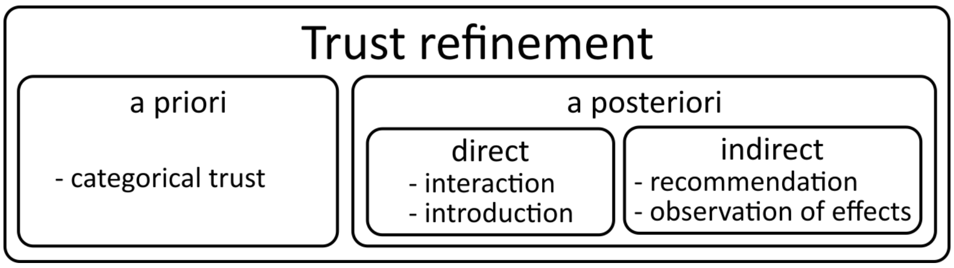


Figure 3.1.8-I: Trust refinement variants

Entities can purposely attempt to refine trust. In order to increase trust accuracy via direct trust refinement, an agent might choose to cooperate with another agent even if its current

[60] Albeit subjective; however, this is an intrinsic property of trust, an external observer could not measure it directly.

entity trust would indicate otherwise without consideration of the potential knowledge gain. In sociological settings, “Trust was built through risking betrayal” ([ADAM05] p. 318). Conscious learning via indirect trust refinement can be attained via communication with third parties. An agent that encounters a previously unknown agent lacks direct sources of entity trust, but it can compensate for this by making an inquiry that other agents can reply to with recommendations regarding the unknown agent.
We could include the time dependence of entity trust by adding a time parameter to the entity trust function. However, old entity trust values should be irrelevant in a decision making process. Any information that an agent could derive from them should already have been considered in the computation of the current entity trust value. The entity trust function itself is therefore time dependent, and only describes a momentary state of a dynamic trust network; agents do not need to memorize a “trust history”[61] for decision making purposes[62].
If trust propagation in a network is based on recommendations or other messages, the recipient of such a message can inevitably only derive a trust value that was valid at the time of transmission. Of course, one could define a transmission time that is considered “fast enough”; i.e. an agent can assume that no major changes in trust occurred between sending and reception of the message. Nevertheless, messages could be delayed due to hardware limitations, changes in network topology or man-in-the-middle attacks; they should therefore carry time information, so that agents can recognize and ignore outdated recommendations. To denote a context-dependent temporary measurement of entity trust, i.e., a single point of a context-dependent (3.1.3D) entity trust function $t: E \times E \times D \to T$, we need to combine the following data:

- Identification of the trustor / trust source $e^+ \in E$
- Identification of the trustee / trust target $e^- \in E$
- Identification of the domain of action $d \in D$
- Trust value / measurement $\mu \in T$
- Timestamp (time of measurement) τ

We call such a temporary measurement a *trust statement* and, adopting the trust edge notation[63] given in [JOS205], notate it as a tuple (e^+, e^-, d, μ, τ). We will later use these trust statements for the communication of entity trust (see section 3.5.1).

[61] Not to be confused with an interaction history, each element in a trust history would correspond to an accumulation of interactions.
[62] However, they might store it for logging purposes, allowing for reconstruction of events and debugging the system.
[63] That third element of a tuple in that notation contains a combination of trust type (direct/indirect + functional/referral). The equivalent here is the domain of action.

3.1.9 Related Terms

A term that is often mentioned together, and occasionally equated, with trust is *reputation*. The Merriam-Webster's Collegiate Dictionary [MERR03] defines it as follows:

(3.1.9A) Definition of reputation in [MERR03]:
"[Reputation is the] overall quality or character as seen or judged by people in general." (keyword "reputation". Retrieved 2015-10-25, from the online version of [MERR03])

When we defined entity trust in (3.1.3C), we based our definition on the trustee's expression of cooperative behavior, but we did not explicitly state whom the trustee should cooperate with. Since entity trust should provide a foundation for the trustor's *trust-based decision making*, it seems obvious that it would consider the trustee's cooperative behavior towards *the trustee itself*.
Herein lies the difference to reputation: if we tried to define reputation in this context, based on cooperative behavior, it would refer to cooperative behavior towards an agent's *interaction partners in general*, i.e., towards the average peer.
The corresponding entity trust is not necessarily identical, but can be derived from reputation, as the following two statements illustrate ([JOSA07] p. 7):

(3.1.9B): "I trust you because of your good reputation."
(3.1.9C): "I trust you despite your bad reputation."

Our definition of entity trust also contains a reference to the trustor's expectation, which makes it subjective, whereas the above definition of reputation is clearly objective. Agents might have different, subjective, opinions of an entity's overall quality or character; [MICH02] captures this using the term "subjective reputation". Seizing on this idea, we distinguish between *entity trust*, *subjective reputation* and *objective reputation* and also include the term *influence* for completeness, to measure the trust that an entity receives within a group. The distinction among these terms is illustrated in Figure 3.1.9-I.

Expectation of...

...regarding the trustee's cooperative behavior towards...

	the trustor	the average group member
itself	Entity trust (3.1.3C)	Influence
the average group member	Subjective reputation	Objective reputation

Figure 3.1.9-I: Distinction of entity trust, subjective and objective reputation, and influence

While entity trust should be considered when interacting with a peer, subjective reputation should be used when rating it. The difference can be illustrated using the example of an online product review system. When rating a product, the user should consider general quality, not the product's utility for the user; a review such as "Being deaf, I found no use for this audiobook; would not recommend 1/5." does not provide the reader with any useful information regarding quality. Nevertheless, an individual rating is always subjective. In the context of trust among agents, the review would correspond to entity trust (and the reviewer should act upon it and not buy another audiobook) and a good review would correspond to subjective reputation, enabling others to derive their own entity trust.

Objective reputation and influence can be regarded as measurements of an agents social standing within a group. The former describes the general consensus[64], whereas the latter describes the actual power an entity can utilize by requesting the cooperation of others.

[64] In the review system example, this would correspond to the often displayed average of ratings.

3.2 Trust Management

3.2.1 Aspects of Trust Management

In open MASs, the distribution of trust in the system strongly influences the incidence of cooperation and betrayal and thus also the efficiency of the system and the individual security for its members. There are multiple stakeholders who can also control the distribution of trust, e.g., system designers (by specifying rules and protocols), agent designers (by implementing trust calculation and interpretation), and individual agents (by propagating trust). *Trust management* is the application of such methods that improve a system's trust distribution by increasing trust coherency and trust accuracy. This includes trust establishment, trust update and trust revocation techniques (cf. [CHOJ11] p. 6).

We call a dedicated subsystem of an MAS or individual agent that contains trust management functionality a *trust system*.

We already mentioned some trust management methods in the previous sections. Individual agents can manage trust for their own immediate benefit by learning via direct or indirect trust refinement (see section 3.1.8), but they can also manage trust for others by issuing recommendations or certificates. Agent designers can implement agent trust behavior while taking into account its effects on the whole system. Their mission essentially is to unify the requirements of (3.1.6E1) and (3.1.6E2). They can also react to emergent risks and weaknesses by altering their agents' behavior via software updates. Lastly, system designers can specify rules that individual agents should follow according (3.1.6E3). This is necessary to enable cooperative trust management; for example, when agents communicate about trust values, they need a common syntax for parsing each other's messages, as well as common semantics that define the risk tolerance that a communicated trust value suggests. The set of all rules that regulate the propagation of trust through a communication medium is the *trust propagation protocol* (TPP) of the system. When multiple agent designers take part in the design of a system's agent population, those agent designers can be regarded as agents in an open MAS themselves and are subject to the requirements in (3.1.6E). The system designers of the actual MAS also design this "meta-MAS" implicitly. There can be various concepts for the interpretation and propagation of trust by agents, that are of equal value when universalized, but less efficient when combined in the system. In this case, the system designers are obligated to meet a decision by introducing tie-breaking rules, according to (3.1.6E3). In concrete terms, this means that system designers should develop standards for an ideal agent's behavior (see Figure 3.2.1-I for an overview of MAS and meta-MAS). We will discuss the requirements regarding managed trust propagation and the ideal agent in section 3.2.2.

One of the main tasks of a trust system is to provide security for its underlying functional distributed system and to protect it against attacks. However, the trust system itself can be vulnerable to attacks and inherent risks, especially when unknown agents can participate in trust management. Such risks for trust systems will be investigated in section 3.2.3.

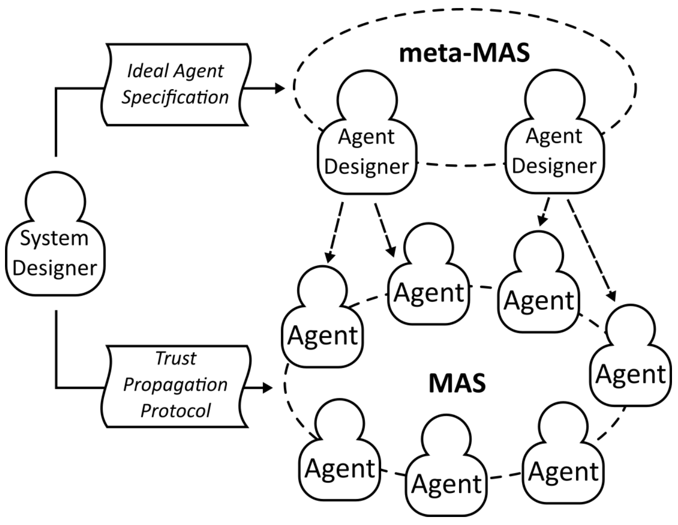


Figure 3.2.1-I: Open MAS and meta-MAS

Various approaches for trust management relate entity trust updates to economic exchange. Interestingly, it is even possible to treat entity trust as a currency without overriding its usual meaning or the meaning given in definition (3.1.3C). Section 3.2.4 addresses suchlike economic approaches for trust management.

When entities manage trust, the associated workload and responsibilities need not necessarily be evenly distributed. In section 3.2.5, we will examine the advantages of different distribution strategies for trust management activities.

3.2.2 Trust Propagation and the Ideal Agent

The simplest method of trust propagation between two agents is direct communication of entity trust values, i.e., the transmission of a trust statement from the trustor to a recipient: a recommendation regarding the trustee. A first distinction can be made between *push* and *pull trust propagation*.

Push propagation can be likened to the publish/subscribe or observer pattern; the request for initiating communication is made by the originator of the relevant data. The trustor asynchronously sends a recommendation to the recipient, for instance, a warning against cooperating with the trustee. Such a warning would usually be broadcast in order to inform all potentially affected agents as fast as possible. In contrast, pull propagation, similar to polling, relies on the recipient initiating the transaction, by sending an inquiry message. As with push propagation, the trustor then issues a trust statement and sends it to the recipient.

A distributed trust system should provide mechanisms for both pull and push trust propagation. On the one hand, if it included only pull propagation, betrayed agents could not warn others. A mobile attacker could exploit the system at will by moving whenever its local influence is depleted. Recipients would have no means of determining when the entity trust they aim to refine changes; therefore, they would need to poll for updates periodically. This forces agent designers to find a compromise between low trust coherency and accuracy (too few inquiries), and unnecessary network load (too much inquiries). On the other hand, in a network featuring only push propagation, agents can not utilize the indirect trust refinement via third parties mentioned in 3.1.8. An agent that comes across a stranger can not determine their trustworthiness. Without pull propagation, it needs to wait for push propagation initiated by third parties before cooperating with the trustee. Ubiquitous undemanded push propagation would create overhead, so agent designers need to compromise again. Figure 3.2.2-I illustrates the overhead introduced by the lack of either propagation mechanism in a trust system.

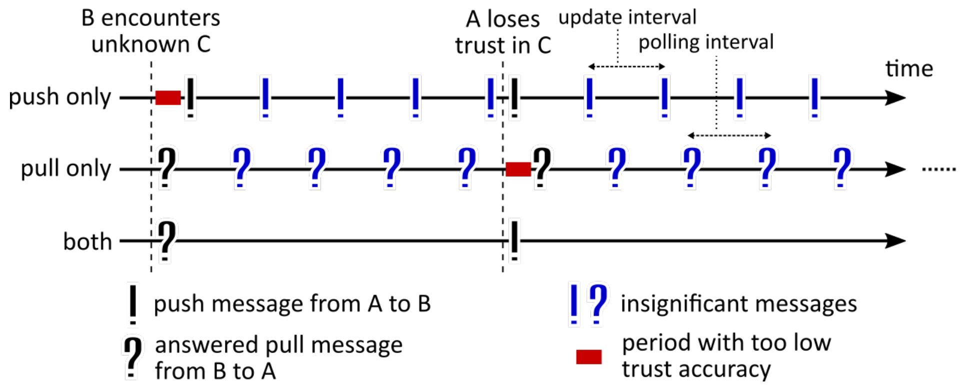


Figure 3.2.2-I: Superfluous trust propagation in systems with only push or pull propagation, compared with a system that features both

Entity trust can also be propagated indirectly with the help of intermediaries. An intermediary agent can conduct a propagation of trust by sending a signed[65] trust statement, issued by the trustor, to the recipient. This corresponds with a requirement imposed on trust propagation in [JOS205]:

(3.2.2A) Unaltered direct trust referral:
"It is thus necessary that [the recipient] receives direct trust referrals unaltered and as expressed by the original recommending party [= the trustor]." ([JOS205], p. 62)

The intermediary can improve objective trust coherency, and also justify its own trust statements. In order to prevent trust loss in itself when reporting significant changes (for example complete loss of trust in another agent after observing an attack), it can combine its recommendation with a similar trust statement it received earlier, sign the result, and send it. Because the signature is only valid for the entire message, nobody can receive only the new recommendation and decide to distrust the recommender based on it. A special case exists where intermediary and trustee are identical, such as with tickets or certificates: the intermediary can use the recommendation to attest its own trustworthiness. Intermediary propagation could also be initiated by an inquiry from the recipient. For instance,

[65] Otherwise the recipient could not validate the results, interpreting it without validation could harm the recipient or trustor.

they might ask for a justified opinion; the trustor would then search for statements that match their own opinion and attach them to their response. If it did not receive such a statement, it could ask other agents via an inquiry itself; this could be regarded as a propagation of inquiries. Push, pull and intermediary trust propagation are shown in Figure 3.2.2-II.

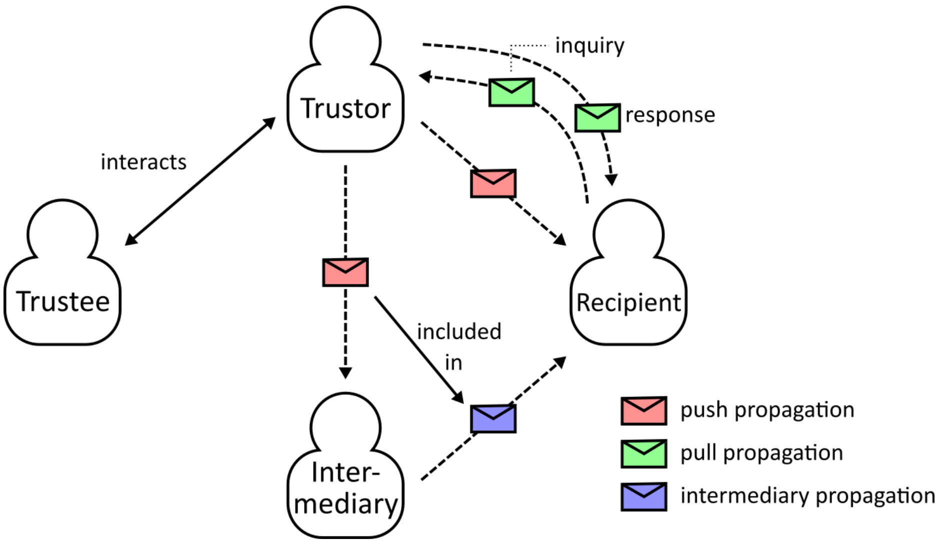


Figure 3.2.2-II: Push, pull and intermediary trust propagation

When an agent processes a received trust statement, the statement is, by nature, never up to date. Between the generation and processing of the statement, the described entity trust can vary. As a consequence, the accuracy of the recipient's entity trust in the trustee is reduced. Moreover, since trust propagation also affects the entity trust from recipient to trustor (see section 3.1.7), the accuracy of trust in the trustor is reduced too. It is therefore in the interests of all stakeholders in trust propagation to define an adequately short period of validity t_{valid} for communicated trust statements. Otherwise, an attacker masquerading as an intermediary could disseminate the statement as often as desired, in order to damage the trustor's reputation (and therefore also influence).

Depending on the trust combination operator used, it might be necessary to prevent recommendations from being considered twice. If the operator is not idempotent, an attacker could alter trust values using controlled overpropagation as an intermediary. Even if it was idempotent, intermediary propagation can result in messages being received twice via different communication paths. While this would not affect the trust network, further processing of both messages can waste resources, for example, processing time spent verifying the signature. A trust propagation protocol should therefore specify at-most-once semantics for recommendations. Recommendations are attributed with a unique identifier, and recipients memorize that identifier until no valid identical trust statement can be received, i.e., until the recommendation's validity expires. Any recommendation with an already memorized identifier is discarded upon reception. The identifier must also identify the trustor; if it were otherwise, an attacker that received a recommendation could issue a different recommendation with the same identifier and entities that receive the second recommendation first would ignore the original; if the attacker possessed a faster communication channel, they could shut down other communication completely. For the same reason, identifiers should not be memorized before the recommendation's signature is verified; however, ignoring duplicate messages does not require prior verification.

How would the ideal agent outlined in section 3.1.6 utilize the aforementioned trust propagation techniques and behave in a trust system? First of all, according to (3.1.6E2), it should act upon trust. When it encounters another agent, it should choose cooperation when entity trust is high, and refuse to cooperate when it is low. Generally, it needs to adapt its risk tolerance; assertions with a lower data trust (derived from entity trust) require more verification to reduce the chance of illusion attacks. This also affects the trust system itself; agents should not waste resources to propagate recommendations or inquiries from untrustworthy agents, or to verify their messages' signatures when they would probably ignore the content anyway. A characterization of trust-based behavior is illustrated in Figure 3.2.2-III. Cooperative behavior within an open MAS can be derived from the ideal agent specification; agents want to determine whether others follow the therein described rules (= they have ideal agent character), since such agents are potential cooperation partners.

Variable	**High entity trust**	**Low entity trust**
Cooperation	engage in	refuse
Resources available for message processing	high	low
Reliance on conformity to specification	high	low
Derived data trust	high	low
Adaption to recommendations	strong	weak
Tolerance of overreaching utility gain	high	low

Figure 3.2.2-III: Characterization of trust-based behavior

Since sufficiently up-to-date trust information is not always available (it would hardly be feasible to always send an inquiry and wait for the response for every decision), an agent needs to derive it, based on the last known value; for example, it could assume equality for low time differences, or reduce the certainty expressed by the old value. In order to utilize this last known value, the agent needs to store this value for each known entity, in the form of a mapping from the set of entities to a set of trust statements[66], the *trust table*. For simple agent implementations, storing own entity trust as statements with themselves as e^+ (trustor) should suffice.

As it is in the ideal agent's interest to increase subjective trust coherency[67], it should actively propagate trust by communicating its own entity trust values (i.e., those describing trust relationships where it is the trustor) in the form of recommendations. To meet the

[66] Not trust values, since time is a relevant component.

[67] This is universalizable (3.1.6E1) to objective trust coherency, according to (3.1.7C).

requirements described earlier in this sections, a recommendation only needs to contain a trust statement, useful additional information such as justifications, and a signature.
The ideal agent should not adjust any trust statements and only send accurate trust values; however, these trust statements refer to the recommender as the trustor, while recipients are rather interested in how the trustee would behave when interacting with themselves. Recipients therefore attempt to derive the latter from the former.
The recipient should be responsible for specializing the data, so the recommender should, instead of making any adjustments to their trust statement, interpret it as being general. What is being generalized is not the trustor as the judging entity, but rather the interpretation of trust regarding the affected entity. Said statement then describes the trustor’s expectations regarding the trustee’s cooperation with any entity (instead of just the trustor), which is subjective reputation (see section 3.1.9). The corresponding data flow is shown in Figure 3.2.2-IV.

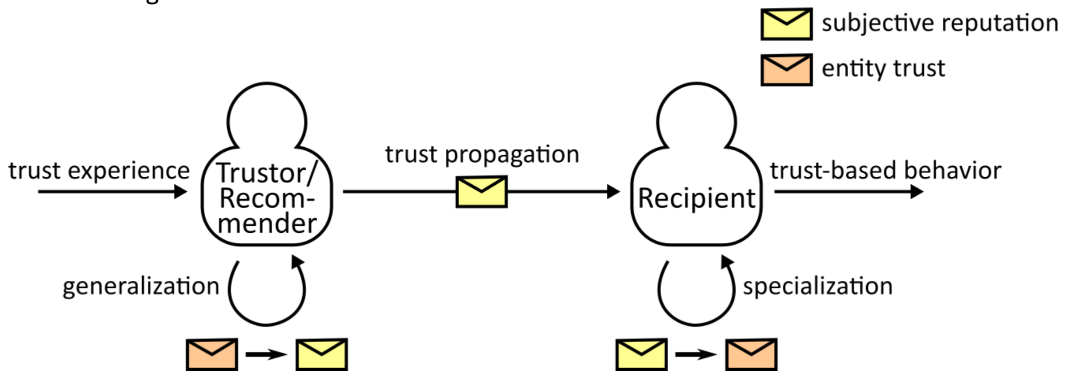


Figure 3.2.2-IV: Trust propagation / Recommendation dataflow

It goes without saying that agents can not constantly broadcast their opinions towards all trusted entities; a gigantic communication overhead would result, with most messages not carrying any significant information. Therefore, one needs to determine under which conditions trust should be propagated, as well as the appropriate propagation techniques.
Push propagation does not transfer significant information if a similar trust statement has only just been sent; hence it should only be applied when the trust value in question has drastically changed (i.e., betrayal was detected) or the communication resources would otherwise be wasted (e.g., unused bandwidth). In order to determine whether a trust value has changed sufficiently, a metric can be defined on the trust space, the *trust value metric*.
The application of push propagation in a MAS would show similarities to a recurrent neural network: the propagation layer can be regarded as a hidden layer with opinions as neurons, computing entity trust values (output layer) from external trust sources (input layer).
Using pull propagation, the recipient itself can determine when an update is necessary. It should only send an inquiry when additional knowledge is required, for example, when it encounters an unknown agent.
Lastly, an agent should use intermediary trust propagation to justify its opinions, whenever recent similar recommendations from third parties are available.

We just discussed variants of trust propagation shown from the viewpoint of the trustor that is also the recommender. It remains to examine the viewpoint of the recipient, particularly, how an agent should interpret and react to incoming recommendations. The agent aims to maintain subjective trust coherency, which is, if previously present, now broken or reduced. Combining known trust values, the agent recalculates entity trust in trustee and recommender to improve coherency, on the one hand aligning its entity trust regarding the trustee with the recommender's opinion, on the other hand potentially reducing trust towards the recommender. Attention should be paid to the fact that the coherency model never requires an increase in trust towards the recommender. If a combination operator required this, one could simply reduce this trust again, even to $\mathbb{0}$, without diminishing trust coherency. Of course, an agent might additionally reward good recommendations with an increase in entity trust later on, interpreting it as cooperative behavior.

A more vivid explanation for this kind of behavior which is also seen in humans might be the following: An agent generally trusts recommendations and adapts to them to a certain extent. However, if a recommendation seems too unreasonable to it, it distrusts the recommender, because they either express poor judgment, or try to manipulate the recipient.

It would hardly be practical for an agent to store all received recommendations and to incorporate the results into every trust coherency calculation, especially since the impact on these calculations will usually be negligible for too old trust statements. Nevertheless, the ideal agent should possess a "short-term memory" component, which stores all recommendations that are younger than a constant time t_{mem}. We establish this requirement for the following reasons:

(3.2.2B) Reasons for a short-term memory component in the ideal agent:

(1) At-most-once semantics

The agent can use the identifiers of previous recommendations to filter out incoming duplicate recommendations. This usecase requires $t_{mem} \geq t_{valid}$.

(2) Justifications

The agent can insert statements from the short-term memory into messages for the purpose of intermediary trust propagation, provided that the statements are still valid.

(3) Preventing race conditions

If multiple related recommendations[68] arrive during a short period of time, race conditions can easily occur. For example, this is the case when trust between two entities is lost spontaneously, in both directions. A third agent might receive both of the resulting negative recommendations. The consequences would then depend on which recommendation is received first, as the second one would be distrusted. Instead, the agent should process the first recommendation immediately (to enable quick propagation), but memorize it. When the second recommendation arrives, it can revise its previous calculation and combine both recommendations into a new calculation, effectively distributing the emerging distrust equally among both recommenders.

[68] I.e., recommendations that affect common coherency constraints.

The existence of such a short-term memory component can also improve trust coherency, because the combination of recommendations as mentioned in (3) permits the agent to solve multiple constrains in a single operation. An improvement in trust accuracy can also result: in the scenario described in (3.2.2B3), the distribution of distrust is more accurate than the state that results from the naïve sequential interpretation of one recommendation at a time.

Figure 3.2.2-V illustrates the necessary components of an agent's trust system in an open MAS. The structure shows similarities to the scheme introduced in [ADAM05]. When recommendations are received, they are processed by the trust management and, if not ignored, forwarded to the trust coherency solver. The trust coherency solver then solves the corresponding subset of the trust coherency problem, updating the trust table. The overall dataflow is analogous to the dataflow illustrated in Figure 3.1.4-II; the updated trust towards the recommended agents corresponds to the mental model of the current trust situation, and the resulting distrust towards the recommender corresponds to the depicted modification of entity trust. Attack detection is separated from the trust system due to the distinction between hostility and untrustworthiness described in 3.1.3.

A possible solution for the aforementioned requirements will be introduced in section 3.5.

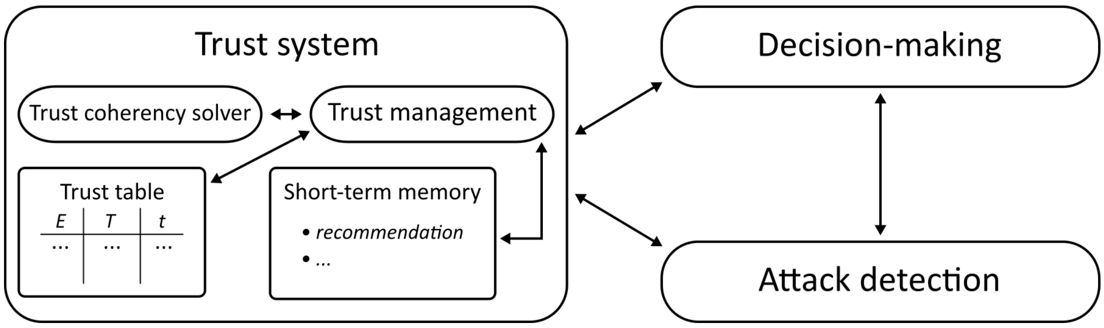


Figure 3.2.2-V: Internal trust system of an agent in an open MAS

3.2.3 Risks for Trust Systems

The operation of an open MAS is vulnerable to certain risks, emerging from either design flaws or malicious disruption of the system. Protection from the latter is of special importance for an open MAS, since its own agents, which inevitably need to be granted access to its resources, can attempt to disrupt it. The trust system protects the MAS from such attacks, but its operation also puts an additional strain on the system's resources. Trust management therefore also includes the management of these resources. On the one hand, too much trust propagation would negatively affect the performance of the system. On the other hand, too less trust propagation results in a lack of trust accuracy and leaves the system vulnerable to attacks.

It is also important to consider the strength of trust propagation. If agents mostly ignore recommendations, the trust systems efficiency is diminished. However, if they attach too great value to recommendations, entity trust can be lost faster than it is generated, eventually leading to ubiquitous complete distrust, the *collapse* of the trust system. Examples of trust system collapse can be found in smear campaigns and health scares; following the circulation of rumors (with stronger propagation than the corresponding data trust would allow), people quickly lose trust in a person, product or company. Trust system designers must therefore ensure that trust propagation is limited by validity periods for recommendations and adequate general distrust.

A phenomenon that is frequently observed in trust systems is the formation of relatively isolated groups with a high inner density of strong trust relations. Such groups can be identified in the corresponding social graph as strongly connected components[69] (*SCC*s, also referred to as *clusters*). Figure 3.2.3-I displays an example of clusters in a graph.

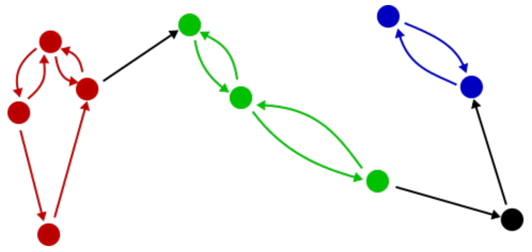


Figure 3.2.3-I: Clustering in a graph. The highlighted subgraphs are SCCs.

One example of this is the so-called *strong set* of a Web of Trust, i.e., the largest completely interconnected set of keys. Initially, clustering offers the advantage that cooperation within each SCC is promoted; a new member can be investigated by few old members and quickly gains the trust of the entire group. Unfortunately, along with the growing size of an SCC, the number of its single points of failure also grows. Essentially, every member of the cluster acts as a CA; the compromise of any of its keys endangers the whole cluster [PERR13]. Note that this risk only affects heavily decentralized systems; more centralized systems such as X.509 have significantly fewer single points of failure such as the root CAs. Clustering itself might not be preventable; if one were to compare a graph whose edges correspond to actual cooperative interactions between agents (nodes), and this graph exhibited clustering too, an optimal graph of trust relations must inevitably contain the same clustered structure. Apart from that, a trust system should aim to minimize *super-fluous* clustering. Most trust relations in a *strong set* do not provide any benefit, except for transitivity, to the nodes, as they will never interact and meet decisions based on this trust. From the viewpoint of an individual agent: trusting more entities than necessary can have a negative impact on security, by increasing the risk for betrayal attacks.

While the trust system protects the MAS, it might itself become the target of attackers who aim to disable it, in order to get access to the underlying resources. The table in Figure 3.2.3-II lists various attacks on trust systems and possible countermeasures which have been investigated in the body of literature that this paper is based on.

[69] A subgraph in which every vertex is reachable from every other vertex.

Name(s)	**Description**	**Countermeasures**
Betrayal attack ([MASH11], [ZHAN11])	Adversary[70] cooperates until reaching sufficient influence, then turns hostile.	Trust is hard to build up, but easy to lose ([CHEN10])
On-off-attack ([CHOJ11], [MASH11]) Conflicting behavior attack ([CHOJ11]) Inconsistency attack ([ZHAN11])	Adversary alternates between cooperative and hostile behavior to avoid detection and breed discord.	Good trust propagation, trust coherency solving[71]
False information attack ([CHOJ11]) Tampering ([REPA06]) Bad-mouthing ([ZHAN11]) Ballot stuffing ([ZHAN11]) Misrepresentation ([SURY06])	Adversary generates arbitrary (unfair) recommendations to increase or decrease a peer's reputation / influence.	Cluster filtering ([DELL00]), Trust coherency solving[72]
Collusion attack ([REPA06], [MASH11], [ZHAN11])	Multiple adversaries collude to isolate other nodes or to boost each others' reputations.	To some extent: Trust coherency solving[73]
Impersonation attack ([SURY06], [ALKA12]) Masquerade ([RAYA07], [ALKA12])	Adversary assumes the identity of another peer to gain its influence or harm its reputation.	Message authentication, Key-identity-binding
Newcomer attack ([CHOJ11])	Adversary registers or fakes a new identity to restore trust after attacking.	Low initial trust, Certification authorities, Web of Trust
Sybil attack ([DOUC02], [GOLL04], [CHOJ11], [MASH11], [ZHAN11], [ALKA12])	Adversary assumes multiple identities in order to launch a collusion or newcomer attack.	Physical verification[74], Adversarial parsimony principle[75] ([GOLL04])

[70] Here: internal attacker that possesses one or multiple agent identities.

[71] Especially the use of intermediary trust propagation allows a group of agents to arrive at a common objective reputation for the adversary, without losing too much trust within the group.

[72] The attacker will harm themselves, and, since the recipient will also communicate with benevolent nodes, most likely more than the victim (which also prevents further attacks). Of course, this argument is based on the attacker not being able to perform a newcomer attack.

[73] Only if there is a benevolent group that is large enough to detect the adversary group and to override their claims.

[74] I.e., investigating whether physical entities that correspond to the logical nodes (e.g. vehicles) actually exist.

[75] When in doubt, a decision-making entity should pick the explanation that involves the fewest malicious entities.

Denial of service (DoS) ([CHOJ11])	Adversary floods the system with unnecessary messages, blocking important communication	Detection in short-term memory

Figure 3.2.3-II: Possible attacks on trust systems and countermeasures

Depending on the structure of the underlying systems, various other attacks on the trust system might be possible, for example, partitioning of the network which effectively partitions the trust system as well.
The system to be introduced in section 3.5 will also be examined theoretically with regard to resilience against the listed types of attacks.

3.2.4 Economic Trust Management

Trust relations often play an important role when monetary transactions are involved; for example, in two-part-transactions, one trading partner has to trust the other to pay, or to provide the sold asset. It would seem sensible to directly relate trust dynamics and economic exchange. Trust can even be regarded as a currency, when entity trust in an agent (i.e., its influence) is exchanged for goods (system resources) or services (cooperation).
To begin with, we investigate the direct exchange in a two-person-interaction, without including trust propagation and third parties. Exchanging trust is simple, in contrast to a usual currency system; there must neither be a physical good transferred, nor third entities present to witness the transaction. If one agent observably increases another agent's utility, the second will naturally increase its entity trust towards the first; if utility is decreased, trust is decreased too. Being aware of this, the first agent can spontaneously help the other in expectation of a future reward; analogically, it will refrain from harming the other in order not to lose a potential cooperation partner. Trust therefore enables the emergence of reciprocal altruism in such a system.
Agents that trade with trust in this manner naturally employ a tit-for-tat strategy: If the respective other cooperates, they will respond with cooperation too (since entity trust was raised to the necessary level); if the other does not cooperate (or betrays), further cooperation is refused. This concept has some advantages. First of all, both entities can valuate the same exchanged asset arbitrarily and different; however, to achieve a long-term stable trade relationship they need to become even in both viewpoints over multiple transactions. Secondly, other than in a classical tit-for-tat strategy, the keeping of a trust value allows this long-term view and the representation of a temporary debt. A series of computer tournaments organized by R. Axelrod ([AXEL84] p. 30) in the 1970s demonstrated, empirically, that "tit for tat" is the optimal strategy in competition scenarios akin to the iterated prisoner's dilemma[76] (IPD).

[76] At least in populations of agents designed by professional game theorists for exactly that purpose. Newer simulations from the Axelrod project on GitHub include different populations and different dominant strategies.

A major disadvantage of most tit-for-tat-like strategies is the potential creation of a catastrophic feedback loop in minor error cases. Such effects could not be observed in Axelrod's tournaments[77]; they would occur in more complex systems where an agent could misinterpret the actions of its peers. In the iterated prisoner's dilemma of two agents A and B, suppose that both agents cooperated on the first turn. However, due to some kind of communication error, B interpreted A's action as betrayal. B, employing "tit for tat", would naturally retaliate by betraying in the next turn, while A still cooperates. A notices the (from their point of view uncalled for) betrayal and retaliates, continuing the cycle[78]. In systems and strategies where agents are prompted to give a retaliation that is harsher than the perceived offense, this process is accelerated; real-world examples of this could be trade embargos, arms races or even the creation of wars. In a trust system, this would merely correspond to a complete loss of trust between both agents; nevertheless, the resulting loss in efficiency should not arise from a simple communication error (or, possibly, brief delusion by malicious agents). Measures to prevent such loops include:

(3.2.4A) Measures to prevent vicious cycles in the IPD:

(1) Damping
Reducing the effects of betrayal on trust, so that a "retaliation" is always slightly weaker than the perceived offense. Agents would aim to be a little friendlier than their peers. If the sequence of trust losses converges, a (theoretically infinite) distrust loop is avoided.

(2) Random cooperation
Essentially the opposite of JOSS. By occasionally attempting cooperation with distrusted peers (a *peace offer*), agents can break negative feedback loops.

(3) Forgetting
In the iterated prisoner's dilemma, an agent could choose to attempt cooperation after being stuck in the loop for a fixed number of iterations. Correspondingly, agents in a trust system could slowly increase trust in others over time (up to some neutral level), allowing distrusted agents to recover.

System and agent designers must implement such measures carefully, since they easily create vulnerabilities to attacks. On-and-off and newcomer attackers can often ignore the long-term consequences of their actions, so agents should not be too forgiving in their response to early betrayal. Defensive strategies can often be countered by negatively mirroring them, hoping to hit an opponent's occasional cooperation with occasional betrayal; being more forgiving allows others to be more abusive.

This simple principle of using trust for reciprocal exchange can also be observed in social and small economical contexts. Humans naturally help one another, often expecting a

[77] However, a similar effect was observed when "tit for tat" played against JOSS, a modified "tit for tat" that has an additional random chance of betrayal in every round.

[78] In the iterated prisoner's dilemma, this does not terminate cooperation yet; every subsequent turn contains one cooperative action and one betrayal. It needs a second error to produce continuous betrayal on both sides ([AXEL84] p. 37).

return of the favor at some point. A local grocery store owner might allow their customers to chalk something up, but only to a certain extent before refusing service.

At larger economic scales, more complex concepts of trust exchange prevail. Trust propagation mechanisms allow a group of agents (a society) to have a common concept of debt; for example, credit cards can be used internationally. While a trading partner might not trust us to reimburse them without a direct exchange of currency, they trust the credit card company, which in turn trusts us. The credit card serves as a certificate of that trust, which corresponds to intermediary trust propagation in our model. The existence of trust propagation allows entity trust to be used as a universal currency and as a tool for resource management in an open MAS. A simplified example scenario is the following: a group of agents share a common communication channel, for example, the mesh network of their individual nodes. They usually treat all packets equal and forward them with equal priority. Now agent A needs to make an urgent transaction with agent C. A therefore requests temporary priority for its messages, by marking them as important. Agent B, the sole hop between them, complies with the request, but lowers its entity trust in A. This distrust is spread in the group by means of trust propagation. Later, A helps another agent D with a similar problem, which causes D to increase trust in A; this increase in trust is also disseminated. When A's initial influence is restored, it has given enough resources back to the system, so that the other agents regard the exchange as fair. This principle allows for a fair distribution of resources without a central authority, even if all agents have different needs.

A similar concept (for MANETs, which can be regarded as a special case of open MASs) has been introduced in [BUTT01]. Instead of using the concept of trust, the authors suggest the use of a virtual currency called *nuglets*; the managed resource is the communication channel (packet forwarding). Nuglets behave just like a real currency; agents spend trust when requesting packet forwarding, and gain trust when forwarding packets for others. Since the system is decentralized, no central authority can control the currency, and the prevention of nuglet forgery or duplications is a complicated security problem. In [BUTT01] (cf. p. 10), this is solved by suggesting the implementation of a tamper-resistant security module in each agent, essentially a *secure element*. However, this simply creates a new central authority (i.e., single point of failure): the manufacturer of the module.

The use of entity trust and trust propagation (in the form which is discussed in this thesis) for resource management circumvents these issues. Since an agent's influence is only controlled by other agents, it can not "forge" trust for itself. It could forge trust for others, but, with regard to entity trust, this simply means providing more resource to its accomplices at its own expense; with regard to subjective (and thus also objective) reputation, this is a tampering attack (see section 3.2.3), which is limited by trust coherency solving[79]. A sufficient majority of agents could still oppress a minority and control all resources (large-scale collusion attack). This is an intrinsic problem of decentralized open MASs and can not be prevented. If an agent is surrounded only by malicious agents, and has no other means of communication, the malicious group can always prevent it from communicating.

[79] The same applies for intentional trust "destruction" (Unfair negative feedback).

Requiring tamper-resistant pre-manufactured modules installed in all agents is equal to requiring that there are no malicious agents, in this regard.
Of course, (semi-)centralized systems are not affected by such issues. The system proposed in [MART00] features a so-called *watchdog* entity. The watchdog entity listens to all communications, even if they do not involve it[80]; it can then recognize misbehaving nodes and issue recommendations to avoid them. Another decentralized approach could be the use of a decentralized virtual *cryptocurrency* such as Bitcoin, although such a system would still be vulnerable to majority collusion attacks (cf. [NAKA08] p. 3).

3.2.5 Where Should Trust Be Managed?

The trust systems referenced in this thesis can generally be grouped into two fundamentally different styles of architecture: *centralized* and *decentralized* systems. Their key difference is the localization of trust management.
Centralized trust systems (CTS) feature a singular entity (also called the *center*), which acts as the sole authority on all trust ratings. Operation of the system is based on a high collective trust towards the center. This does not require that the center observes and assesses the trustworthiness of all other agents; it can also accumulate ratings from peers (cf. [JOSA07] p. 18). For example, the eBay reputation system could be regarded as a trust system. Despite the fact that users rate each other, the system is centralized, since users base their decisions on the total score (ideally, objective reputation) that is accumulated by the service[81]. The center is often not the only specialized entity in a centralized system, a common addition is a hierarchical structure of subordinate centers. Variants that feature multiple centers are often referred to as *semi-centralized systems*.
Decentralized trust systems (DTS) are characterized by the lack of such a central entity. There are usually no special roles; all agents are equal to the trust system and trust propagation can emerge at any communication path in the network. Operation of the system is based on a ubiquity of communication partners and, therefore, opinions. Since there is no central ordering influence, global structures can only emerge from the accumulation of individual behavior in local subsystems.
There are several distinctive advantages and disadvantages to both centralized and decentralized systems, which often depend on specific additional requirements. Frequently, another determining factor is the underlying network topology, or infrastructural limitations. The lack of communication paths can often prevent the suitability of one architectural style. A general discussion of the aforementioned advantages and disadvantages will be the subject of the following chapters 3.3 and 3.4.

[80] This might not always be possible: For example, in the usecase of MANETs, this requires a *promiscuous* mode of the wireless interface (cf. [MART00] p. 256).
[81] As a counterexample, this does not apply for the review system. Users interpret recommendations of trustworthiness that are contained in individual reviews (subjective reputation). Although communication is relayed via the server, this can be regarded as direct trust propagation between two users. However, one could also argue that the system is not perfectly decentralized, since eBay (the center) potentially has the power to manipulate reviews.

3.3 Centralized Trust Management

3.3.1 Centralized Trust Systems

In trust propagation protocols of centralized trust systems, [ABDU97] identifies a common trusted intermediary as the key component. This intermediary, also called the *trusted authority* (TA), is used to form trust relationships between the other agents (cf. [ABDU97] p. 49). In a fully centralized system, the center is the only TA; every recommendation that other agents receive must have been issued by it. As a necessary condition, the other agents (users) do not place any referral trust in each other, at least not within the trust system. A remaining variable factor is the existence of recommendations sent by agents (assessing the trustworthiness of other agents) to the center. If agents rate each other in this way, and the TA objectively computes an average of these trust values (i.e., its valuation of subjective reputation is identical with objective reputation), the resulting system is called a *centralized reputation system* (cf. section 3.1.9). [JOSA07] lists two fundamental aspects of such systems: *centralized communication protocols* which allow for trust propagation between users and the TA[8283] and a *reputation computation engine*, the component utilized by the TA to compute reputation values.

In centralized trust systems where users do not rate each others' trustworthiness, the TA issues purely subjective recommendations. These recommendations are often static, or have a long period of validity, since there is no continuous influx of potentially altering recommendations. They can take the form of signed certificates, enabling intermediary trust propagation without further communication with the TA (now also a certificate authority). It is a common misconception that a certification-based PKI can only provide identity management. In fact, a CA can certify any subjective statement, for example, in the form of: "This entity can be trusted for that purpose". For instance, the X.509 v3 standard's certificates can carry such information in an extension field (cf. [RFC5280]).

A semi-centralized trust system contains multiple TAs; they often form a hierarchical structure of CAs as depicted in Figure 3.1.7-VII. Another option is to include multiple root CAs (for example as a result of merging two PKIs) using techniques such as cross-certification.

In a CTS with a single root TA, the center essentially controls the whole system, allowing for clear definitions of global behavior and good prediction of scenarios in a global view. They are most suitable when entity trust is rather static, since trust propagation can be difficult[84] (see section 3.1.7), and independent of the trustor, since all users have to accept the same reputation value (or the TA's opinion) for a single trustee. Trust relations between any pair of users are implicitly established by their certifications, which makes centralized trust systems ideal for random, unpredictable interactions between users. They also provide easily implementable security and privacy. Key management for secure communication can

[82] Referred to as *central authority* or *reputation center* in [JOSA07].

[83] These protocols are the centralized equivalent of the trust propagation protocols discussed in 3.2.

[84] Unless the trust system is a reputation system where interaction requires communication with the center anyway (example: online shops with user reviews).

be added to the center (since it already needs to be fully trusted) and identities can be protected using pseudonyms. Each user can store a sufficiently large set of certificates (for offline interaction), or generate a partially random identifier for each interaction, encrypted with a CAs public key. When the communication partner requests a recommendation for the identifier from the center, the CA can then derive the original identifier, but the communication partner is unable to track the user's behavior.
Since all recommendations received by agents are issued by the TA and individual agents hold close to no power, CTSs also offer excellent protection against most insider attacks, for example tampering attacks. In reputation systems, a majority of users could still suppress a minority using negative recommendations; however, as we will point out in section 3.4.2, this applies to decentralized systems as well. Trust coherency solving can not be applied in a CTS; it would produce no useful results, since all users have to put complete trust in a TA and are not allowed to reduce it as a result of trust propagation.
To sum up, a CTS should be implemented if strong requirements on the system's global behavior are given, availability of the TA is guaranteed[85], trust can be regarded as objective, or privacy between users has higher priority then privacy towards the TA.

3.3.2 Vulnerabilities

The structuring influence of a central authority can also be to the detriment of a trust system. The number of possible paths along which trust can be propagated is limited by the valency of the central node; communication with the TA can easily become a bottleneck. Any necessary security features that can be removed from the nodes are simply shifted onto the central authority. Blocking a few communication paths can easily incapacitate a subsystem, or even reenable insider attacks (the trade between availability and consistency discussed in section 3.1.7). If an attacker gains temporary control over the TA, they can arbitrarily recommend (certificate) allied malicious nodes and carry out global attacks. The TA effectively becomes a Single Point of Failure (SPOF). Semi-centralization can render the issue more critical; completely trusting one entity is dangerous, but completely trusting a multitude of entities is even worse. In systems with cross-certification, the security of any certificate is reduced to that of the least trustworthy CA, i.e. the weakest link in the chain (cf. [GUTM11]). Security breaches of influential CAs have occurred in the past, leaving entire PKIs vulnerable. Two prominent examples of this are the breaches of the Comodo CA in March 2011 [COMO11] and the DigiNotar CA in July 2011 [PRIN11]. The latter led to man-in-the-middle attacks using fake *.google.com certificates in Iran and to the subsequent bankrupt of DigiNotar BV. Centralization of a system is often a difficult decision to make; while the number of risks is reduced, their magnitude greatly increases. As past experience shows, rare but severe risks are much harder to manage.
Furthermore, the operation of a CTS requires constant availability of the TA, resulting in instability against network partition. The static nature of a CTS complicates adaption to dynamic or complex trust relations. Each change of a trust value influences the entire

[85] Or temporary decommissioning of the user nodes is tolerable otherwise.

system. In addition, trust is no longer a subjective value; all users share the same entity trust towards an individual. Isolated clusters, which frequently emerge in decentralized systems, often reflect preexisting underlying structures, yet can not be represented in a CTS. Users only know the global assessment of the others' behavior, but not the local; this enables attackers to cause damage within a small group while being covered by positive recommendations from the unsuspecting rest. Counteracting this with harsher punishment for few negative recommendations has the downside of becoming more vulnerable to tampering attacks.
While the general availability of trust relations between all users is ideal for random interactions[86], it can have a negative impact in systems where users mostly act in small peer groups. Every unused trust relation adds an unnecessary risk; as soon as one malicious agent tricks the CA into recommending it as trustworthy (or turns malicious after recommendation), it can attack the entire system, even the majority of agents that should not have the need to trust it.
If agents have largely different local views and peer groups, they will most likely also acquire different impressions of the same cooperation partner, inconsistent with the opinion that the TA imposes upon them; trust accuracy is reduced. Provided that the agents, or agent designers, are aware of this, trust in the TA is lost[87]. As [ABDDU97] puts it:

(3.3.2A) Credibility loss of a CTS TA:
"[…] a TA can never be a good enough 'authority' (or recommender of trust) for everyone in a large distributed system. Its credibility depletes, and its recommendations increase in uncertainty, as its community of trustees grows." ([ABDU97] p. 49)

Of course, the foundation of any CTS is the complete collective trust towards the TA. In an open MAS, the establishment of a TA that can fulfil this requirement is not always possible if the set of agents or of agent designers is predetermined (they might not be able to agree on a TA). As previously stated, this can not simply be solved by adding alternative TAs.
One of the main arguments for the implementation of a CTS is often privacy. But as with security, the risks are not eliminated and instead concentrated on a single entity which holds full knowledge and power. Especially if the agents are or are owned by humans, they might prefer multiple members of their peer group knowing about a small subset of their actions (as it naturally is the case in everyday life) to a single entity knowing about all of their actions (which fiction frequently depicts as a dystopic scenario).
In conclusion, CTSs should not be implemented when the individual agents' local views greatly differ from the global view (complex internal structure), entity trust is strongly subjective (only locally observable behavior, or no consistent concept of cooperative behavior), the underlying network is incompatible (frequent partitions) or when agents or agent designers can not agree on a common completely trusted authority.

[86] This implies that the local view of an individual agent is similar (ignoring scale) to the global view.
[87] This loss is usually not represented within the trust system.

3.4 Decentralized Trust Management

3.4.1 Decentralized Trust Systems

Many of the papers referenced in this thesis propose decentralized trust systems[88]. Unlike in a CTS, agents are usually homogenous and do not possess any special roles or hierarchical positions. This lack of a central authority leaves the agents responsible for calculating their own trust values and organizing the system (self-organization), a “bottom-up” approach, in contrast to a CTS’s “top-down” approach (cf. [THEO06] p. 319). Compared to a CTS, local requirements are easier to meet (for example, subjective trust coherency), while global behavior is harder to predict[89]. Agents often have no concept of a global view; they operate solely on their local view. Any global structure or behavior is an emergent property of their individual behaviors; not only the underlying MAS, but also the trust management is fully distributed. As a consequence of this, DTSs are highly robust; any agent can be removed from the system without hindering its operation. In addition, insider attacks (such as the tampering attack) can only affect a small group of agents, namely the local environment of the malicious node. Trust propagation can then take multiple paths in parallel to recover from the attack, inducing a collective loss of trust towards the attacker and restoring trust in framed benevolent nodes.

In a decentralized reputation system, agents act upon subjective instead of objective reputation. According to [JOSA07], the fundamental aspects of a decentralized reputation system are a *distributed communication protocol* and a *reputation computation method*. The distributed communication protocol allows participants to exchange ratings (recommendations), which they combine in order to compute their own subjective reputations. This combination operation is handled by the reputation computation method. An important aspect of a DTS is that each agent can have a different reputation computation method. The advantage of this is that an agent’s policies “[…] need not be communicated, so there is less ambiguity and no effort involved in trying to understand them.” ([ABDU97] p. 51) A DTS can operate without any specification of agent behavior as long as they understand the communication protocol, although an “ideal agent behavior” specification could improve the system as suggested in section 3.2.1. Unlike in a CTS, different agents can put a different amount of trust in the same peer and are not restricted to an averaged (or completely subjective) value provided by a TA; consequently, DTSs do not suffer the trust accuracy problems of CTSs pointed out in section 3.3.2. Since there is no need for unity[90], open MASs with a lot of strong external authorities with different requirements, for example, agent designers, can be supported by a DTS, but not by a CTS.

[88] Notable approaches include [ZIMM95], [BLAZ96], [ABDU97], [BUTT01], [MICH02], [SABA02], [KAMV03], [ZHAO04], [DOTZ05], [REPA06] and [SURY06].

[89] An important exception are patterns that emerge from subpatterns which are clearly defined at agent-level; for instance, objective trust coherency emerges from subjective trust coherency.

[90] Especially not for appointing a single entity that everyone can put complete trust in.

Moreover, any CTS can be regarded as or nested in a DTS, provided that the necessary communication paths exist. In our open MAS model, this permits agent designers to implement their own CTS for trust propagation among their agents. As agents have full control over their own interpretation of trust computation; this is legal in a DTS, although it could conflict with the respective ideal agent specification.

Agents only need to put trust in agents in their local environment, i.e., their peer group, which is constituted of frequent communication partners. Therefore, a DTS does not contain more strong trust relations than necessary. Weaker trust relations (this could be regarded as a “ready for connection” state) are established via trust propagation. This is especially efficient when interaction is bound to distance in an underlying topology and agents are more likely to communicate when their distance in the resulting trust graph is lower.

While simple global patterns can emerge from local mechanisms, the global structure can also become highly complex, making the system very adaptive, especially to complicated trust networks in subsystems and to rapid changes. Accordingly, ad-hoc networks, being based on highly dynamic and rapidly changing topologies and often being bound to a spatial topology, benefit greatly from an inner DTS; hence the decentralized approach has been most widely implemented and put into use in such systems (cf. [THEO06] p. 319).

Compared to a CTS, a DTS does not require as clear trust semantics. In a CTS, propagated trust values directly affect the actions of agents and grant access rights, whereas in a DTS, each agent can run its own computation of entity trust based on received trust statements. An agent might regard its peer group as too credulous and decide to reduce entity trust derived from recommendations. With trust coherency solving, there is a positive feedback for the automatic development of a common interpretation of trust values, as agents that deviate from the norm issue inaccurate recommendations and temporarily lose trust. The semantics of trust values might drift, but the system remains operational as the agents adapt, provided that they can interpret entity trust as relative to their long-term environment[91]. An illustrative example of this are ratings provided by humans. Throughout the history of such a rating system, the density distribution of ratings can change; juries in contests might become reluctant to give impolitely low ratings, or feedback effects cause voters to depart from the intended interpretation of the ratings (tactical voting). Even so, recipients in a DTS can adapt and arrive at the same interpretation as the recommenders. Nevertheless, a basic default interpretation[92] is necessary to promote convergence and to prevent chaos.

If an open MAS has few requirements on global behavior, the underlying topology is dynamic and complex, conflicts between agent designers or agents exist, or clear trust semantics are hard to define, the implementation of a DTS is indicated.

[91] Even a non-linear example of this is observable in online markets; buyers have adapted to regard 5-of-5-star average ratings as worse than 4.9-star ratings, since the former usually indicates that there have been too few ratings to produce a good average.

[92] This is another example of a tie-breaking rule in an MAS.

3.4.2 Vulnerabilities

Lacking a centralized component that manages trust, a DTS has to shift the burden of interpreting and combining trust recommendations onto the agents. The resulting disadvantage is that more responsibility and expertise is required from these agents (cf. [ABDU97] p. 51). In an open MAS, if too many agent designers fail to implement sustainable trust policies[93], there will be low average entity trust between agents and cooperation is obstructed, despite the fact that the agents, or agent designers, do not wish to exploit one another.

While a DTS promises a higher availability of trust ratings, this comes at the price of consistency. As pointed out in section 3.4.1, agents usually do not need to have any consistent or even useful opinion of far distant peers, but this can be abused by malicious agents if the system lacks trust propagation. By simply changing its local environment, i.e., moving, an agent can switch between peer groups to dodge the penalty of trust loss after attacking. In order to prevent this, trust propagation must be strong and fast enough, so that a bad reputation follows an attacker. This can be combined with a low initial trust for unknown agents; however, this also affects trustworthy agents that travel long distances. Trust propagation must therefore also facilitate that good reputation follows such an agent.

On the other hand, too strong or too frequent trust propagation can also have adverse affects (see section 3.2.3). Implementation of a DTS accordingly requires careful adjustments of many parameters to create a stable system.

A particular danger of overpropagation is unnecessary clustering. Superfluous trust relations expose the system to the same risks as CTSs and single points of failure are created (cf. [PERR13]). This effect can be contained if propagated entity trust is sufficiently diluted along trust paths, resulting in small, fuzzy peer groups.

Since there is no central controller, global behavior only emerges from the local behavior of (homogenous) agents and is subject to feedback effects, as the agents react to their environment. This makes the system's global behavior harder to control and to predict. If there are strong global requirements on a system, such as the collaborative fulfillment of a single task, a CTS would be more appropriate. The same applies for the distribution of resources. A central authority can easily distribute resources among the agents; resources in a DTS that are initially shared by all agents complicate this process. In economic systems, the problem of a decentralized distribution of resources is solved using currencies; a similar solution can potentially be applied to DTSs (see section 3.2.4).

[BISM12] proposes a central misbehavior evaluation scheme for VANETs, citing a lack of power in detecting Sybil attacks with local misbehavior evaluation, a key component of a DTS. In fact, this is not limited to VANETs; a general vulnerability to Sybil attacks is a frequent point of criticism for DTSs. According to the paper that coined the term "Sybil attack", "If distinct identities for remote entities are not established [by a certification authority], [peer-to-peer] systems are susceptible to Sybil attacks" ([DOUC02] p. 5). While a

[93] The system designer can remedy by specifying ideal agent behavior.

decentralized identity management can prevent the forgery of new trusted identities[94] and therefore newcomer attacks[95], it can not prevent the following attack:

(3.4.2A) Single-target Sybil attack on a DTS:

(1) As the adversary, control a single malicious node that is identified by the DTS.
(2) Using only the last assigned identity, gain trust through cooperation or other "legal" means.
(3) Without using any characteristic of already assigned identities, reapply for identification of the node, which is still possible without an underlying centralized identity management[96].
(4) Repeat steps 2 and 3 until you control n node identities.
(5) Construct a situation where your node can communicate with a target node, but the target node can not detect the overlapping of your identities[97].
(6) Using any kind of multiplex, initiate one communication session with the target for each owned identity. The combined influence now allows you to continue with strong collusion or tampering attacks.

This kind of attack is similar to a collusion attack, the difference being that in (3.4.2A), coordination of the attack[98] and gaining trust (step 2)[99] is easier, while collusion attackers have the advantage of higher mobility. Another variant is that the adversary performs an improved on-off-attack by always switching to the locally most trusted identity. Sybil attacks with a single malicious node are also much cheaper to perform than collusion attacks or Sybil attacks with transferred identities (from multiple nodes) and pose a serious threat to systems such as VANETs (cf. [GROV11] p. 152). We conclude that a DTS at risk from Sybil attacks should include a centralized identity management. This seems somewhat contradictory, but centralized identity management does not have all the drawbacks of centralized trust management, even in an ad-hoc environment. For example, an X.509 PKI could be utilized, with the CA only certifying unique identity, not trustworthiness (no trust accuracy issues), preventing the creation of duplicates, for example by observing node creation. Unlike in a CTS, revocation of these certificates would rarely be necessary; trust can be highly dynamic, but identity is never lost; that is, unless the private key is stolen.
Theft or malicious transfer (Sybil attack at the cost of acquiring an additional node) is still possible; however, this also applies to almost every CTS. It can be prevented by introducing secure elements, or additional characteristics that can be used to identify a node[100].

[94] For example, using a web of trust.
[95] Against nodes whose policies specify to not trust any nodes not identified by the system.
[96] This is why participants at a key-signing party present identity documents.
[97] In a VANET, this could simply mean to hide behind a corner reporting multiple fake positions.
[98] Since a single node controls all identities.
[99] Collusion attackers risk that not all conspirator nodes are equally / sufficiently trusted and a distrusted node pulls the other members down, whereas in a Sybil attack, all identities gain trust with the same tactic in the same environment (step 2).
[100] An example of such a characteristic is the photograph on a passport.

In a DTS, ensuring privacy between users, especially pseudonymity, is more complicated than in a CTS[101]. Local observers can keep track of an agent's interaction history and the agent can not change its pseudonym without losing its reputation. On the other hand, [GOLL04] even argues that a DTS offers better privacy, since no privacy sensitive data flows to a centralized location. If pseudonymity is nonetheless required, a DTS with high agent mobility could allow its agents to transfer reputation to a new pseudonym with a broadcast message to their local environment. An agent then only loses the entity trust from remote peers, which is relatively outdated anyway, does not affect current operation and can be easily regained via trust propagation. The downside of this approach is that local peers could link the identities, but unless they collude with many remote nodes, they can not track interaction history[102]. In order to hide the pseudonym changes from potential eaves-droppers (that are not trustworthy nodes), the transfer messages could be encrypted.
Lastly, since all agents have more or less the same power, a colluding (global) majority could gain control over the trust system, if agent designers did not react to such a development.
Decentralized trust systems should not be implemented if strong requirements on global behavior exist, sufficient trust propagation is not possible, or the trading of influence for pseudonymity is unacceptable.

3.5 Example: Generic Trust System

3.5.1 Generic Trust Propagation Protocol

In this section, we draft an example trust propagation protocol to meet the requirements identified in 3.1-3.4: the Generic Trust Propagation Protocol (GTPP). It is agent-centric and most suited for decentralized systems, but also allows centralized subsystems to be embedded in a DTS. Figure 3.5.1-I lists the 4 message types that make up the GTPP; Figure 3.5.1-II lists constraints on valid messages that are not included in the ASN.1 notation. Depending on the domain of application and the underlying network structure, additional fields might need to be included in the messages.

[101] The Web of Trust for example leaks private data such as communication partners (cf. [PERR13]).
[102] And if they did, no anonymity or pseudonymity scheme could provide privacy.

```
GenericTrustPropagationProtocol DEFINITIONS ::= BEGIN
  RecommendationMessage ::= SEQUENCE {
    timestamp UTCTime,
    id        MessageID,
    trustor   EntityID,
    trustee   EntityID,
    domain    DomainID,
    value     TrustValue,
    inquiry   MessageID OPTIONAL,
    inquirer  EntityID  OPTIONAL,
    citations SEQUENCE OF RecommendationMessage,
    signature Signature
  }
  InquiryMessage ::= SEQUENCE {
    expiry    UTCTime,
    id        MessageID,
    inquirer  EntityID,
    trustee   EntityID,
    domain    DomainID,
    degree    INTEGER(0..MAX),
    signature Signature
  }
  TrustedMessage ::= SEQUENCE {
    timestamp UTCTime,
    id        MessageID,
    trustor   EntityID,
    value     TrustValue,
    message   CHOICE {
      trustedMessage TrustedMessage,
      plainMessage   BIT STRING
    },
    signature Signature
  }
  TransferMessage ::= SEQUENCE {
    timestamp    UTCTime,
    id           MessageID,
    oldTrustee   EntityID,
    newTrustee   EntityID,
    oldSignature Signature,
    newSignature Signature
  }
END
```

Figure 3.5.1-I: Message data structures of the Generic Trust Propagation Protocol (GTPP) in ASN.1 notation [ITUX680]. Figure 3.5.1-II describes additional constraints.

RecommendationMessage

(1) The combination of *trustor* and *id* must be unique over messages.
(2) The fields *trustor* and *trustee* must not have equal values.
(3) The fields *inquiry* and *inquirer*, if present, must correspond to the *id* and *inquirer* fields of a previous InquiryMessage.
(4) Each element of *citations* must contain the same value of *trustee* as the containing message and must have a unique *trustor* within the message.
(5) The field *signature* must contain a valid signature of the sender identified by *trustor* over all other fields.

InquiryMessage

(1) The combination of *inquirer* and *id* must be unique over messages.
(2) The field *signature* must contain a valid signature of the sender identified by *inquirer* over all other fields.

TrustedMessage

(1) The combination of *trustor* and *id* must be unique over messages.
(2) If *message* contains another TrustedMessage, it must contain a different value of *trustor*.
(3) The field *signature* must contain a valid signature of the sender identified by *trustor* over all other fields.

TransferMessage

(1) The combination of *oldTrustee* and *id* must be unique over messages.
(2) The fields *oldTrustee* and *newTrustee* must not have equal values.
(3) The field *oldSignature* must contain a valid signature of the sender identified by *oldTrustee* over all prior fields.
(4) The field *newSignature* must contain a valid signature of the (same) sender identified by *newTrustee* over all other fields.

Figure 3.1.5-II: Additional constraints on GTPP messages.

The message type RecommendationMessage enables push propagation. Agents can identify themselves as the trustor in a trust relationship and issue an associated recommendation that corresponds to the trust statement (see section 3.1.8) of $(trustor, trustee, domain, value, timestamp)$. In addition, the sender of a RecommendationMessage can include related recommendations to justify their statement in the field *citations*. This allows for intermediary trust propagation, but also forces the intermediary to add their own opinion. The sending agent can therefore not use intermediary trust propagation for tampering attacks without risking distrust towards itself, should the recipient detect[103] the attack. By

[103] I.e., detect a large difference from its own or other's opinions.

signing the entire message, the sender also confirms the reception and processing of all nested RecommendationMessages, which increases data trust in them[104]. RecommendationMessages can be uniquely identified by the MessageID *id*, generated by the sender/trustor, in combination with the trustor's identifier. Note that this message type contains no reference to a period of validity. The corresponding semantics is that the trust statement is only fully valid, i.e., correctly describes the trust relation, at the time of issuance (*timestamp*). It is up to the recipient to decide when a statement lies too far in the past to be considered significant for their decision-making. Instead of using a fixed period of validity, the recipient could also account for the time elapsed by gradually decreasing the certainty of the derived entity trust.
The second message type, InquiryMessage, offers a mechanism for pull propagation. Again, messages are uniquely identified by *id* and the sender (*inquirer*). By sending such a message, the inquirer requests a RecommendationMessage as a response, with identical trustee and domain of action. The sender determines the period of validity (ending at *expiry*); messages should only be responded to if the inquirer can receive the response within this period. Responders can refer to the original inquiry in the *inquiry* and *inquirer* fields of their RecommendationMessage to increase data trust[105]. As mentioned earlier, RecommendationMessages can contain additional nested recommendations. Naturally, nested responses are more desirable for the recipient as they provide more trust-related information and also further protection against attacks, because the intermediaries confirm the nested recommendations. On the other hand, the preparation of a nested response can be time-consuming and costly, since the intermediary might need to start own inquiries to acquire enough usable justifications. To measure the trust related information contained in a nested recommendation, we propose a metric called the *degree* of a recommendation.
It is computed as follows:

(3.5.2A) Degree of a recommendation:
Let any set of recommendation S represent a recommendation containing all recommendations that are elements of S. The degree of S is defined by the following recursion:

$$size(S) = 1 + \sum_{r \in S} count(r)$$

$$degree(S) = \sum_{r \in S} count(r) + degree(r)$$

An example tree of nested recommendations is illustrated in Figure 3.1.5-III. Each addition or confirmation of a recommendation increases the degree of the outer recommendation by 1.

[104] It proves that the new recipient is not the sole recipient of these messages and a tampering attack would come at a higher cost.
[105] By demonstrating that the current propagation of trust was initiated by the recipient and not them; therefore, they did not have the freedom to chose the trustee to recommend (which is inconvenient for attackers).

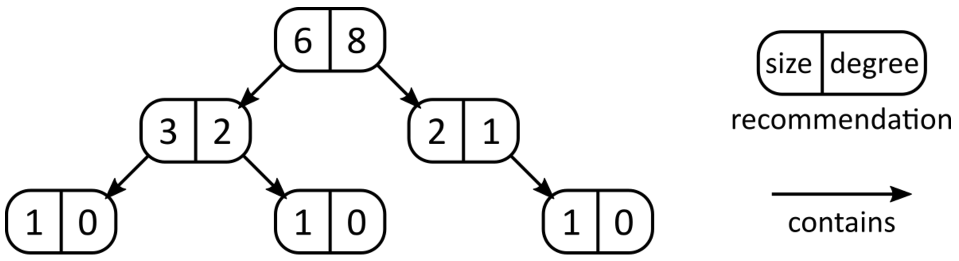


Figure 3.5.1-III: Degrees of nested recommendations.

For pull propagation, an inquirer can specify the desired degree of responses via the *degree* field of an InquiryMessage.

[DOTZ05], [REPA06] and [CHEN10] employ a technique called *opinion piggybacking* for the communication of data trust or relevant entity trust in ad-hoc networks. When a message is forwarded, the recipient is often interested in additional opinions on its trustworthiness. The forwarding entity can add its own opinion without creating the overhead of additional transmissions. To utilize this mechanism, the GTPP includes the message type Trusted-Message. The message contained in its field *message* can be any message of a higher application layer, but also another TrustedMessage; multiple opinions can be attached to an opinion that is forwarded via multiple hops (cf. [DOTZ05] p.1). Since the messages are nested, this approach carries more trust-related information with the same network load, compared to approaches where opinions are simply concatenated.

Lastly, the GTPP contains the message type TransferMessage to enable trust-preserving pseudonym changes as discussed in section 3.4.2. An agent can send a TransferMessage to select trusted peers that replace its identifier with a new one internally. General privacy is preserved, as long as the agent can rely on the recipients not propagating the information and no other agent being able to deduct a link between the old and new identities.

While all messages that are part of the GTPP can only have one sender, identified by the fields *trustor*, *inquirer* and *oldTrustee*/*newTrustee*, respectively, they can have an arbitrary number of recipients. For example, voluntary[106] push propagation messages should usually be broadcast, whereas transfer messages should only be sent to trusted peers.

3.5.2 Ideal Agent Behavior

As argued in section 3.2.1, in designing a trust system we should specify guidelines for the implementation of an ideal agent, especially for how it should interpret, propagate, and act upon trust.

The purpose of trust in an MAS is to defend against insider attacks by recognizing agents that do not aim to maximize global utility and instead try to abuse the help of cooperating agents for their own benefit (see section 3.1.6). Both the trust propagation protocol and the ideal agent specification are rules demanded by (3.1.6E3) and must therefore be universalizable in compliance with (3.1.6E1). As a result, a benevolent (and hence

[106] Here: not responding to an inquiry.

trustworthy) agent (or agent designer) that acts according to (3.1.6D) has no reason to deviate from these rules. This allows us to use compliance with the specifications as a condition for, and indicator of, trustworthiness. An agent should interpret entity trust as the assumption of conformance to the ideal agent specification (including the GTPP) and fair use of system resources. It should keep track of entity trust by deriving and remembering trust information from interactions, and recognizing a local peer group of frequent collaborators. Not following the protocol, for instance by never responding to inquiries, causes other agents to distrust the agent, and a loss of influence; this also provides an incentive for every agent to behave according to the specification.

The goal of the agents' trust propagation is to increase objective trust coherency, or equivalently, increasing each other's subjective trust coherency to reciprocally improve their own trust accuracy. Whenever current events reduce trust coherency, the ideal agent should be prompted to produce relief by means of broadcasted push propagation. This can be implemented by not only storing the current entity trust value for a trust relationship in the trust table, but also the last propagated value. When the current value is altered, the agent can compare the new value to the last propagated and use the difference as an indication of whether the new value should be propagated (since the old value, as seen by others, is now inaccurate). For example, when the agent detects an attack, it immediately loses a lot of trust in the attacker, so it should recommend it as untrustworthy in order to warn its peers, by broadcasting a RecommendationMessage. As another example, the agent might often grant an advantage to a second agent, but gets nothing in return. With each interaction, it loses a little trust towards the second agent, which has no immediate consequences[107]; however at some point, the accumulative distrust causes the agent to rate the second agent negatively[108]. This also benefits the resource use for trust propagation by optimizing it according to distance in the trust graph, which correlates to importance of the trust value. When an agent propagates trust, the effect is diluted and subsequent agents in the propagation chain will derive small enough changes to forego further trust propagation. That is, unless they receive an independent similar recommendation from a different rater, which should naturally increase the range of trust propagation.

Additionally, when the required resources are currently not in use and a trustee is within the agent's current peer group, it should also occasionally broadcast RecommendationMessages to make up for the decay of certainty of aging trust statements[109], maintaining a high local trust accuracy so that no further trust propagation is needed if an immediate cooperative maneuver suddenly becomes necessary.

Push propagation should be initiated by an agent sending an InquiryMessage when it lacks sufficient trust accuracy to make trust-based decisions, for example when it encounters a previously unknown agent. When an agent receives an InquiryMessage, it should respond with a corresponding accurate RecommendationMessage. If the message was broadcast to

[107] If it were otherwise, long-term cooperation would be nearly impossible.

[108] And also, as we pointed out in section 3.2.4, to cease granting favors.

[109] If the other agents only have very old trust statements that make up their knowledge of one another, they can not sufficiently rely on their accuracy when choosing cooperation.

a large number of agents, it should wait for a random short amount of time and cancel the operation if it receives another agent's response, so that there is no unnecessary overhead created. If the InquiryMessage specifies a non-zero recommendation degree, the agent should eventually start another inquiry to collect trust statements as justifications, but at first issue a recommendation with the current maximum possible degree, to enable third agents to generate a response. An agent can always respond with a higher degree to minimize the risk of losing trust, as long as this does not waste too much system resources.

To process incoming recommendations, the ideal agent should use trust coherency solving (see section 3.1.7). The trust system should define a combination operator on its trust space, that calculates new entity trust for both trustee and trustor of a recommendation, while preserving trust coherency, i.e., the fulfilment of the corresponding constraint. The received trust value will often largely differ from an agent's original opinion, for example, if the recommendation is an unexpected warning resulting from a detected attack. If this is the case, application of the combination operator will create significant distrust in the recommender. This is a necessary temporary evil; it protects the agent from tampering attacks. If the suspicious trust statement was accurate, affirmative statements from other agents will restore trust towards the recommender. Trust coherency solving should never produce more distrust than necessary for enabling the desired entity trust towards the recommended trustee.

Up to now, in discussing trust coherency solving we ignored the domains of action for simplicity reasons. While the domains of action of the involved trustor-trustee and recommender-trustee relations must be identical, the affected trustor-recommender relation has referral character and can be different. This can be formalized with an endomorphism $r: D \rightarrow D$ that maps a domain to its corresponding referral domain[110].

The concrete definition of the combination operator is implementation-specific. The trust system may require multidimensional trust values, for example, for representing certainty independent of trust and distrust (cf. Jøsang's Subjective Logic). Besides adapting the original entity trust value, the combination operator should increase certainty when receiving similar recommendations in such a system. The weighting of the combination operator, i.e. whether it tends to produce strong distrust in the recommender and low trust adaption (*skeptic* operator) or weak distrust and high trust adaption[111] (*credulous* operator) essentially defines the magnitude of trust propagation in the system.

In systems where the agents need to manage and distribute system resources, the ideal agent should use entity trust as a currency by increasing entity trust in others when receiving access to resources or services and decreasing trust when providing them (see section 3.2.4).

Agents need to act upon trust, not only when selecting cooperation partners, but also in general decision-making; for example, agents should decrypt equal-priority messages from

[110] Note that a domain can be its own referral domain, for example when there is only one generic domain or only one generic referral domain.

[111] Of course this also depends on the initial trust in the recommender.

more trustworthy peers first, as the risk of subsequently discarding the message due to low data trust is less likely. For a characterization of trust-based behavior, see Figure 3.2.2-III.
Open MASs featuring agents that are or are associated with humans usually need to fulfil special requirements with regards to privacy. The implementation of privacy-protecting behavior is facilitated by the message type TransferMessage of the GTPP. When an agent wishes to switch to a new pseudonym, it should select a set of trusted peers that can vouch for it, and send them an encrypted TransferMessage after switching to the new identity. The agent can repeat the same TransferMessage when necessary (for example, after switching the local environment), but it should not act within the trust system using an old identity again. Accordingly, recipients of a transfer message should copy entity trust from the old to the new identity and lose all entity trust towards the old identity[112]; the latter is necessary to prevent Sybil attacks.

3.5.3 Properties

[ADAM05] lists the following requirements for their trust management system, based on the Interpersonal Trust Model described in [ABDU00]:

(3.5.3A) Trust Management System Properties (cf. [ADAM05] p. 321):

(1) Trust [is] context dependent.
(2) Trust [has] positive and negative degrees of trustworthiness.
(3) Trust [is] expressed in continuous values, as described by Marsh [MARS94].
(4) Trust [is] based on experiences and observations between individuals.
(5) Trust information [is] exchanged between nodes.
(6) Trust [is] subjective. Nodes [calculate] different reputation values for the same observed node.
(7) Trust [is] dynamic and [is] modified, in a positive or negative direction, based on new observations and reports.

It is easy to see that the herein described system fulfills all of these requirements. We add the following properties expressed by the system:
(3.5.3B) Trust Management System Properties (cont.)

(8) Trust converges to coherence, i.e. meets common "transitivity" requirements without contradictions as a result of trust propagation.
(9) Trust can be used as a currency for inter-agent transactions.
(10) The system protects the individual against tampering attacks by implicitly producing distrust towards unfair recommenders.

Corresponding to the differentiation pointed out in section 3.1.3, this model of trust management does produce clear assessments of (un-)trustworthiness, but not of benevolence / maliciousness. It should be noted that the system therefore can not serve as an attack

[112] Not only reverting to the default state of general distrust against unknown identities, but also marking the old identity as destroyed; the agent should not be able to gain influence under that identity again.

detection scheme. It rather allows for the exclusion of unreliable cooperation partners and forms a basis for trust-related utility transactions. An attack detection system can be built on top of the trust system and process trust information as well as other sensory data.

3.5.4 Implementation

To implement the aforementioned mechanisms, an agent could incorporate an internal trust system as depicted in Figure 3.2.2-V. Naturally, an agent designer does not have to utilize any of the non-functional concepts described in this section; they could even choose to embed a centralized subsystems including a fully trusted central entity that manages recommendations (certificates) for their own agents; this would be fully compliant with the ideal agent specification, since agents are "responsible for their own fate". Trust can be gained via trust propagation and external sources such as previous interaction, and possibly via categorical trust such as "fully trust all other agents designed by the same agent designer". The internal policies that define the respective impact of these sources may arbitrarily differ among agents. Non-universalizable behavior such as overreaching utility gain in transactions, or unfair ratings, is not explicitly forbidden (this would usually be impossible to enforce in an open MAS), but gradually punished with distrust from others instead.

Therefore, for an agent designer, the only strong requirement is to integrate their agents with the trust system and the underlying protected system. Agent designers are advised to follow the specifications issued by the system designer.

For a system designer, implementation of the aforementioned trust system requires the specialization of the GTTP and the corresponding ideal agent behavior (3.5.2) in accordance with the logical environment. In order to that, they must:

(3.5.4A) Trust system specialization:

(1) Choose an adequate trust space T along with a corresponding interpretation, a trust value metric (see section 3.2.2) and an encoding of trust values.
(2) Define a trust combination operator $\circledast: T \times T^2 \to T^2$ on the trust space that produces / increases subjective trust coherency[113] and meets their additional requirements[114].
(3) Define sets of entity, domain and message identifier types (EntityID $= E$, DomainID $= D$ and MessageID) including encodings.
(4) Define the referral domain mapping $r: D \to D$.
(5) Specialize the ideal agent behavior according to the environment[115] and issue a specification.

[113] For all practical purposes, it does not need to produce a stable state within a static system, as long as it converges to a state where fluctuations are within an acceptable margin.

[114] These include requirements on the speed, impact (relative to experience-based trust) and balance (how much the recommender is distrusted) of trust propagation.

[115] For example, in a VANET (fictional): "Distrust when peer drives on the wrong side" or "Trust when member of police force".

We implemented a small generic library to aid in the implementation of trust coherency solving and GTPP simulation; its sources are attached in the GenericTrustSystem project. We also began the implementation of a research tool that enables the investigation of suitable trust combination operators by performing trust coherency solving (see Figure 3.5.4-I). Unfortunately, due to time constraints, we could not complete it for this thesis; we will nevertheless continue development. For the interested reader and developer, its sources are also attached in the TrustCoherencySolver project.

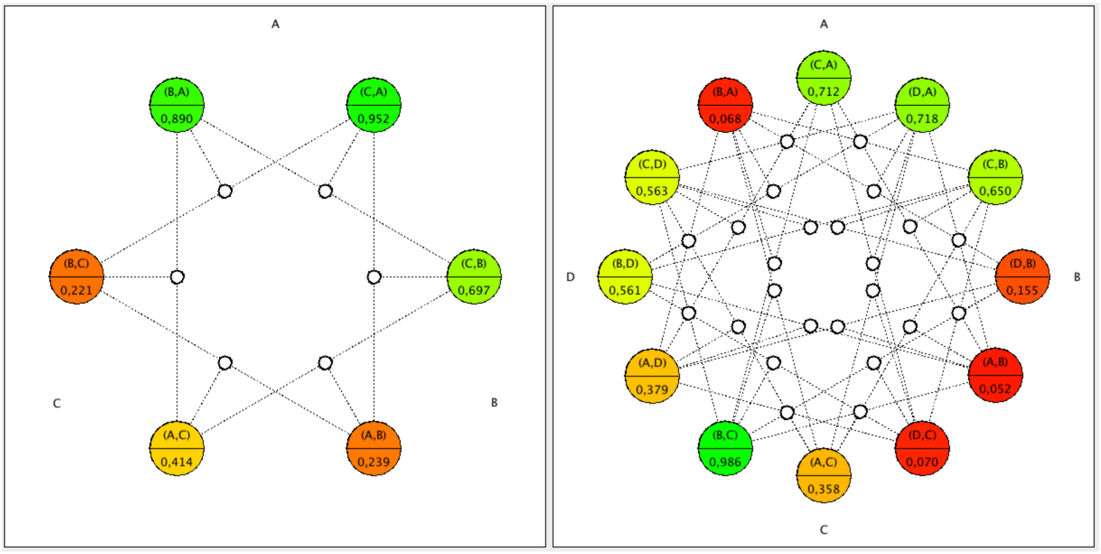


Figure 3.5.4-I: Screenshots of the TrustCoherencySolver UI for 3 and 4 nodes

In the following chapter, we will assume the role of a trust system designer for a VANET (Vehicular Ad-Hoc Network) and perform the above operations to create an informal draft, but not a complete formal specification. The system and attacker models will be simplified to not shift the focus off the trust system design. We will then implement a subset of the specified functionality by extending an existing VANET simulator, run some exemplary scenarios, and analyze the results.

4 DTM in V2X communications

4.1 Motivation

For demonstration and further investigation of the concepts proposed and discussed in section 3, we are in need of an environment where our generic trust system (see section 3.5) could be employed in a real-world application. Since we are only interested in the development of a trust system, there should be an already implemented or at least theoretically developed underlying open MAS that we can with integrate it with. Completely centralized trust systems are already omnipresent and have been intensively researched, hence we would prefer an underlying system that is largely decentralized.

As of now, current traffic systems could be regarded as an open MAS[116] that is functionally decentralized[117] and features rules and regulations that limit the agents' operations. Centralized attack prevention exists[118], but there is no strong integrated trust system. In fact, there is currently no need for such a trust system, since trust-related attacks either require relative huge effort[119] or are easily detectable and ineffective due to lack of reliance[120].

As mentioned in section 2.3.2, the advent of V2X communication will change the current state of trust relations within this traffic system. In a VANET, trust-related attacks are much cheaper (e.g. delusion or DoS attacks), and, with the current trend towards autonomous vehicle services, the agents become susceptible to software malfunction[121]. It seems unlikely that no hacker will ever seize of control of a node or create a fake trusted identity, and vehicles have already been recalled due to delayed detection of software malfunction in the past, so the system must still be regarded as an open MAS.

As a result, future V2X systems will require the implementation of distributed (but not necessarily decentralized) trust systems, where vehicles report[122] and receive trust-related information. Since trust can not be regarded as completely static[123] and a multitude of

[116] Drivers must pass tests to enter the system, but are from then on largely autonomous.

[117] The problem "Move goods and passengers to their destinations without collisions" is solved by the set of all drivers.

[118] The police force as a singular entity observes the agents and issues traffic tickets.

[119] For example, stealing and repositioning a road sign.

[120] For example, the malevolent misuse of turn signals.

[121] Not only to failure of their own system but also to failure of other nodes, by relying on communicated information.

[122] For example, suspicious behavior.

[123] New vehicles enter the system and existing components might be hacked or prove faulty.

manufacturers can produce agents, the trust system must include both a mechanism of trust propagation and a common specification of ideal agent behavior (see Figure 3.2.1-I). The concept proposed in section 3.5 meets these requirements.
For the aforementioned reasons, we will attempt to demonstrate and investigate the application of our generic trust system in a VANET.

4.2 System Model

Because there is no large-scale VANET available that we could integrate our trust system with, we need to acquire a (necessarily simplified) simulation of a VANET, which in turn requires a simplification of our system model of a VANET, and the selection of simulation software prior to the concrete specification of said model. We filtered the simulation software found by means of online research according to various criteria, including the possibility of close-to-real-time rendering of the current state on a conventional PC, and the necessary effort to integrate a trust system (especially source access). The simulator that most closely matched our criteria[124] was VANETsim, an open-source simulator written in Java and introduced in [TOMA14]. Its source code and documentation can be accessed at https://github.com/VanetSim/VanetSim. While this simulation is good at approximating the dynamics of local node environments, it lacks multiple aspects of realistic driving behavior[125]. As a consequence of this, we reduced our model of message exchange to messages that do not affect driving behavior.
A VANET in the context of our model (V2V-only) consists of a set of nodes (vehicles) that move, according to usual traffic regulations, within a 2-dimensional map, that corresponds to realistic road infrastructure. VANETsim does not simulate the so-called “urban canyons” or any other physical intricacies of the wireless medium; we therefore define that every pair of nodes that are within a fixed distance of each other will be considered as *connected* and able to communicate (Ad-Hoc connection). Each node can be assigned a *utility* value (real number) that it aims to increase. Nodes that are connected to each other exchange messages at regular intervals. During message exchange, both nodes independently select an *attitude* on a scale between cooperation and betrayal (allowing for the simulation of trust-based behavior and attackers) and can also transmit a single GTPP message[126]. When two nodes cooperate, both of their utility values increase by a value dependent on their local environment ($\propto k^{-1}$ for nodes surrounded by k other nodes at flawless operation conditions[127]). This approximates the results of the randomly distributed long-term operation of the more complicated actual VANET usecases. Betrayal by both nodes has no specified effect[128]. If one node betrays and the other cooperates, the traitor gains a great

[124] This is mostly because it focuses on simulation of the application layer, where our trust system would be located.
[125] Most important for our considerations, maneuvers such as overtaking.
[126] This simplifies the simulation, but also leaves us unable to simulate (D)DoS attacks.
[127] So that under normal conditions local average trust is position-independent.
[128] We do not aim to simulate interaction between attackers.

amount of utility while the cooperator loses an even larger amount. Utility per node is capped at an upper maximum value that represents flawless operation of the VANET.
By default, there are no fractions and all nodes are benevolent and utilitarian (they have ideal agent character), which implies that they will always fully cooperate with other benevolent nodes to maximize global utility (sum of all node utilities) when the system is in a state of flawless operation. The benevolent agents learn from experience (i.e., adjust trust values) and make honest reports (i.e., issue recommendations that are independent of recipient and trustee identity[129]).

4.3 Attacker Model

We define an attacker, or adversary, as a singular node (inside attacker) that is exempt from the default node behavior and aims to maximize only its own utility; we will therefore not be able to simulate Sybil attacks. As pointed out in 4.2, attacks should not affect driving behavior; nevertheless, they should correspond to somewhat realistic attack scenarios. From the lists in section 2.3 and section 3.2.3, we derive the following set of adversary types:

Identifier	**Name**	**Simulated scenario**	**Attacked system**
A_1	Exploiter	Will chose minor betrayal instead of cooperation, to simulate not returning resources and services (Trust as a currency).	VANET
A_2	Spammer	Will send a message that should be displayed to the driver, claiming complicated hazard information. Message contains spam instead (Bogus information, major betrayal). Can be detected immediately[130].	VANET
A_3	Slanderer	Will broadcast unfair negative trust information for specific targets (Tampering attack).	Trust system
A_4	Colluder	Will broadcast only positive information for other colluders (Collusion attack). Should always be combined with another attack type.	Trust system

Figure 4.3-I: Simulated adversary types

All of these can be combined with betrayal (B_1), on-off (B_2) and newcomer (B_3) attack behavior (see Figure 3.2.3-II).

[129] Still dependent on the interaction history, of course.
[130] Within the simulation model, ignoring the driver's reaction time.

4.4 Trust Model

Our trust model is derived from the generic trust system proposed in section 3.5. To specialize and implement it in the context of a VANET, we need to follow the steps outlined in (3.5.4A):

(4.4A) Trust system specialization for our simulated VANET:

(1) For simplicity of implementation and visualization, we chose a one dimensional trust space of $T = \{x \in \mathbb{R} | 0 \leq x \leq 1\}$ with a linear interpretation of “Do not cooperate at all” at 0 and “Choose maximum possible cooperation” at 1, for benevolent nodes. The trust value metric is the absolute difference that we will refer to as d_t. Encoding will be specified in section 4.5.

(2) We chose the experimentally selected trust combination operator of

$$v_{RT} \circledast (v_T, v_R) = (v'_T, v'_R) \text{ with}$$
$$v'_T = \begin{cases} v_T + p v_R (v_{RT} - v_T) \; if \; v_T \geq v_R \\ v_T + q v_R (v_{RT} - v_T) \; if \; v_T < v_R \end{cases} \text{ and}$$
$$v'_R = v_R - q v_R d_t (v_T, v_{RT}),$$

v_T / v'_T being the old/new trust value of the trustor towards the trustee, v_R / v'_R being the old/new trust value of the trustor towards the recommender, and v_{RT} the recommended value. p and q are control parameters for the propagation strength of negative and positive recommendations, respectively. We will demonstrate the effects of this operator in the following sections.

(3) The set of entity and message identifiers will be defined in section 4.5. We use a single generic domain of action Ω for simplicity reasons (generalized trust). Encodings will be specified in section 4.5.

(4) As a consequence of (3), $D = \{\Omega\}$ and $r(\Omega) = \Omega$.

(5) Ideal agent behavior was outlined in section 4.2 and will receive more concrete specification in section 4.5.

4.5 Draft

We extend the VANETsim (branch “master” on GitHub, commit of 2014-04-01) to include a simulated trust system. VANETsim offers the creation of reproducible scenarios that define an initial state of the simulation, as well as events. To preserve backwards-compatibility, we plan not to modify the scenario format, and instead build another system of so-called “trust simulation scenarios” on top of it.

VANETsim already offers a lot of additional features, including the simulation of attacks, but they are not adapted for integration with a trust system. To reduce complexity, we deactivate everything except for the traffic simulation in our test scenarios. Instead of using the built-in attacker management, we extend ordinary *Vehicle* nodes by attaching an internal trust system to them and assigning them a role within the current scenario that determines their behavior. We add a new class *TrustSimulationScenario* to the package *vanetsim.scenario*, which manages the currently active trust simulation scenario and the corresponding roles.

We define the following roles to implement the adversary types defined in 4.3-I:

Name	Attacker type (4.3-I)	Special behavior	Internal identifier
Benevolent	—	—	ROLE_BENEVOLENT
Exploiter1	A_1	Never cooperates.	ROLE_EXPLOITER1
Exploiter2	A_1	Betrays by always providing slightly less utility than expected.	ROLE_EXPLOITER2
Spammer1	A_2B_1	Will simulate bogus information attack by greatly reducing utility for its local environment on activation. Can be assigned to same node again to trigger multiple attacks.	ROLE_SPAMMER1
Spammer2	A_2B_2	Will continuously perform bogus information attack on all Spammer2Victims.	ROLE_SPAMMER2
Spammer2Victim	—	—	ROLE_SPAMMER2VICTIM
Slanderer1	A_3	Will continuously provide negative recommendations for all Slanderer1Victims.	ROLE_SLANDERER1
Slanderer1Victim	—	—	ROLE_SLANDERER1VICTIM
Colluder1	A_1A_4	Will continuously provide positive recommendations for all other Colluder1s. Otherwise behaves like Exploiter2.	ROLE_COLLUDER1

Figure 4.5-I: Roles within a trust simulation scenario

Benevolent nodes assume the following ideal agent behavior:

(4.5A) Benevolent node behavior:

(1) For local trustees with entity trust > *baseTrustCooperation*, cooperate minimally, increase granted utility linearly with entity trust.
(2) When receiving utility, adjust entity trust based on the difference to the expected result. Generally, increase trust slightly for every positive interaction.
(3) For local trustees with entity trust > *baseTrustPropagation*, accept their recommendations and interpret them using trust coherency solving.
(4) When an own trust value has changed by more than *maxTrustDistanceLocal*/-*Global*, propagate the new value by broadcasting a *RecommendationMessage*.

Other nodes override this behavior as specified in the column “Special behavior” of Figure 4.5-I.

In order to provide the *Vehicle* nodes with an internal trust system that meets the requirements of section 4.4, we add the package *gts.simple* to the GenericTrustSystem library. The individual requirements shall be implemented by the following classes:

Requirement	**Implementing Class**
One-dimensional trust space [0,1] with the absolute difference as metric	*OneDTrust*
Trust combination operator as defined in (4.4A2)	*SimpleOperator*
Single generic domain of action Ω	*SimpleDomain*
Internal trust system	*SimpleNode*

Figure 4.5-II: Classes implementing the internal trust system of a *Vehicle*

For trust-related communication between nodes, we define the class *Recommendation-Message* to encode the eponymous message type of the GTPP. Message identifiers are integers and encoded as *int* values. Other identifiers are encoded as references to the respective objects. Trust values are encoded as instances of *OneDTrust* (value class).
An early prototype showed severe performance issues with pull and intermediary trust propagation[131]; therefore, we choose not to include nested recommendations and *Inquiry-Messages*. Instead, frequent random push propagation compensates for the lack of pull propagation (cf. Figure 3.2.2-I).
We implement the single trust simulation step described in section 4.2 via the sequence illustrated in Figure 4.5-IV, that is executed once per node each step. The active trust scenario will be chosen at compile time to reduce unnecessary modifications of the VANETsim UI.
The user can assign roles by clicking on the point representations of vehicles on the map. The role assigned depends on the active scenario and the number of assignments, as specified in Figure 4.5-III.

Scenario	**Click #**	**Assigned role**
scenario1	≥ 1	Exploiter1
scenario2	≥ 1	Exploiter2
scenario3	≥ 1	Spammer1
scenario4	1	Spammer2
	≥ 2	Spammer2Victim
scenario5	1	Slanderer1
	≥ 2	Slanderer1Victim
scenario6	≥ 1	Colluder1

Figure 4.5-III: Role assignment in trust simulation scenarios

[131] These would not appear if the system was actually distributed and we implemented more sophisticated agent behavior that includes discarding inquiries.

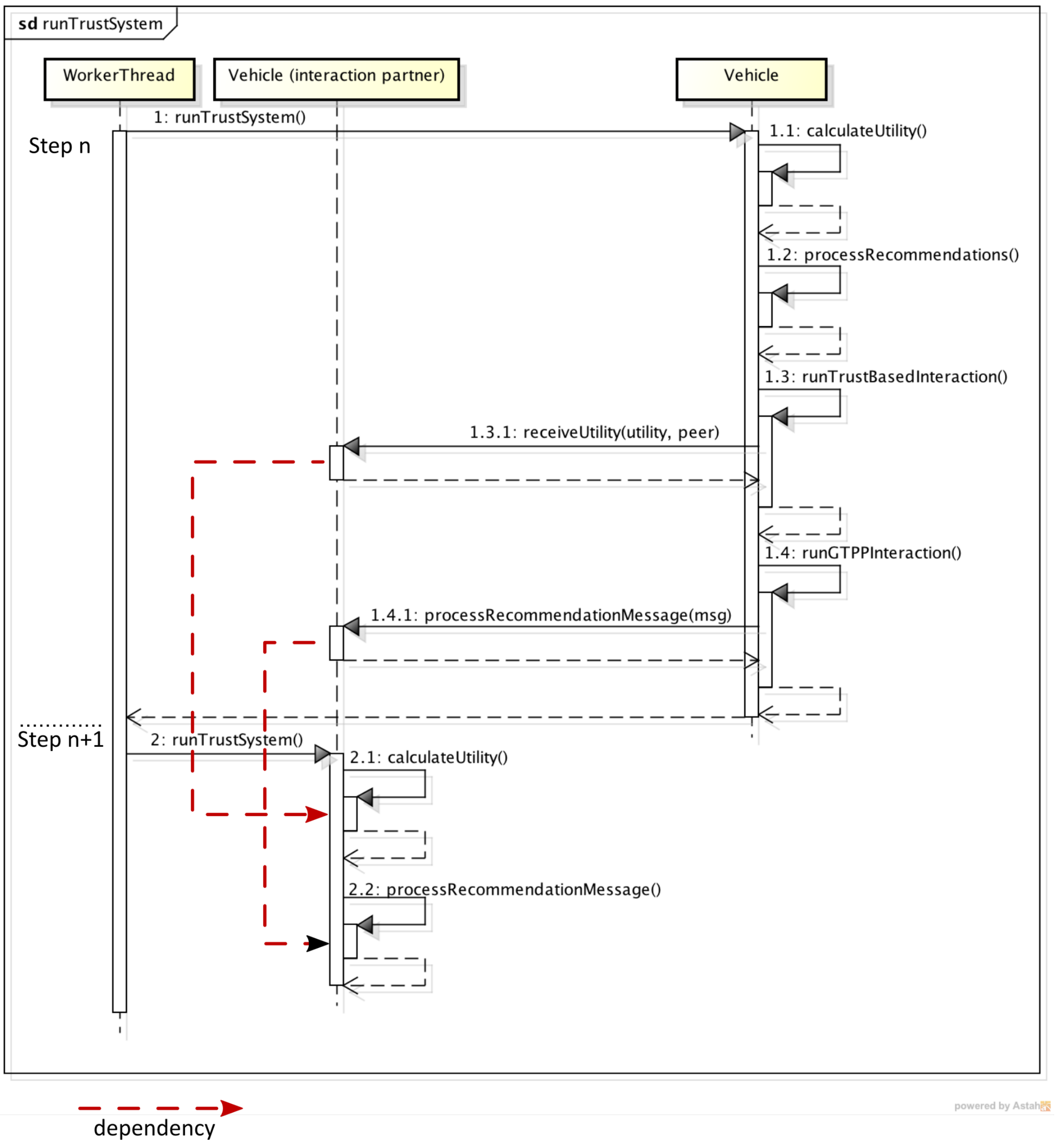

sd runTrustSystem
WorkerThread
Vehicle (interaction partner)
Vehicle
1: runTrustSystem()
Step n
1.1: calculateUtility()
1.2: processRecommendations()
1.3: runTrustBasedInteraction()
1.3.1: receiveUtility(utility, peer)
1.4: runGTPPInteraction()
1.4.1: processRecommendationMessage(msg)
....
Step n+1
2: runTrustSystem()
2.1: calculateUtility()
2.2: processRecommendationMessage()
dependency
powered by Astah

Figure 4.5-IV: Partial sequence diagram of two trust simulation steps

We also need to visualize trust relations, so that the user can observe the state of the trust system. Visualizing the huge amount of global trust relations would overcomplicate interpretation, hence we show only those that affect cooperation, i.e., relations between connected vehicles (local).
A trust relation is visualized as the half of the edge between trustor and trustee that is adjacent to the trustor, and colorized to indicate the level of trust. We also visualize a vehicle's current local influence through the color of its point representation on the map. An example of trust visualization is shown in Figure 4.5-V.

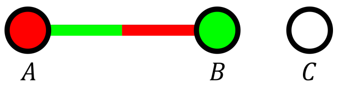


Figure 4.5-V: Example of trust visualization. A completely distrusts B, B completely trusts A, A has minimal local influence, B has maximal local influence, and C is isolated.

The aforementioned UI modifications are implemented by modifying the classes *Mouse-ClickManager* and *Renderer*. Altogether, the following changes are introduced to the VANETsim class structure:

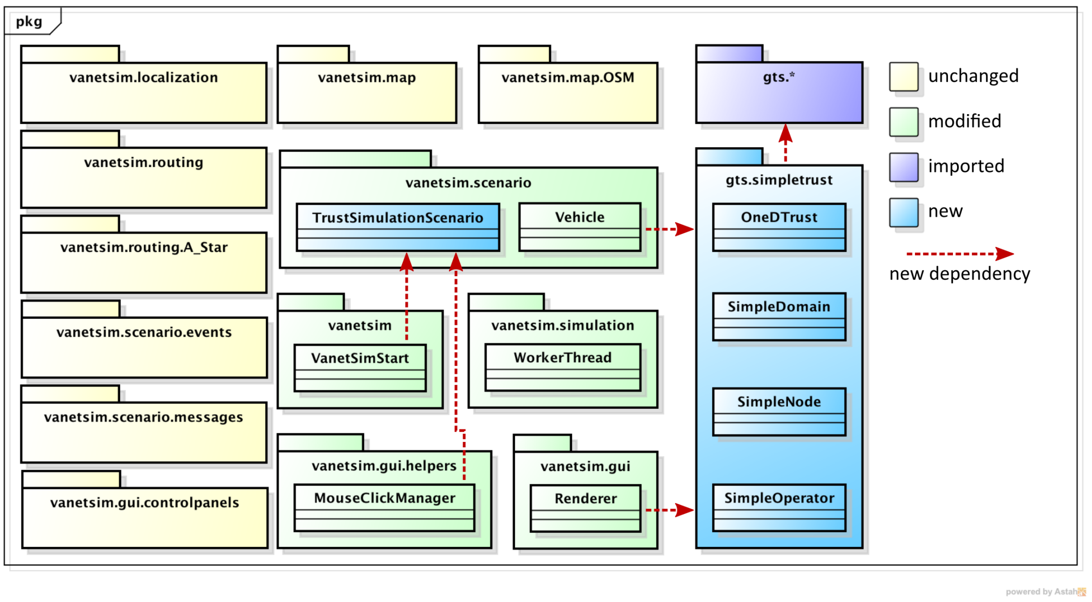


Figure 4.5-VI: Changes to the VANETsim class structure

4.6 Implementation

We implemented the trust simulation within the Eclipse project in the attached folder "VANETsim", closely following the draft from section 4.5.
To provide a concrete and reusable simulation environment, we used the VANETsim to generate a map and scenario file, based on a map of a small region of Hamburg, Germany, that was downloaded from the OpenStreetMap project (www.openstreetmap.org). The map file can be found under "VANETsim/TrustSimulation/Hamburg_KleinFlottbek.xml" and the scenario file (500 vehicles simulated) under "VANETsim/TrustSimulation/Hamburg_KleinFlottbek_TrustSim500.xml".
We adjusted the simulation parameters until the desired effects could be observed. We chose the following values:

- *baseTrustCooperation*: 0.4
- *baseTrustPropagation*: 0.5
- *maxTrustDistanceLocal*: 0.8
- *maxTrustDistanceGlobal*: 0.16
- *SimpleOperator.p*: 0.1
- *SimpleOperator.q*: 0.08

It should be noted that these "desired effects" (and therefore also adequate parameter values) differ from the effects that would be desirable in a real VANET. For example, trust loss as a consequence of minor egoistic behavior (Exploiter2) should be much slower in reality. However, this would require that the user of the simulation runs it for the same, long time period[132]. We therefore accelerated the detection of attacks by adjusting the parameters.
VANETsim uses multithreading, with each simulation thread managing a single rectangular region of the map. To avoid complications with trust propagation at the borders between regions, we modified the simulation to use just one single small region.
We then ran all of the trust simulation scenarios specified in Figure 4.5-III and analyzed the observed results.

[132] On a 2011 MacBook Air, we could not run the simulation much faster than in real-time.

4.7 Analysis

4.7.1 Predefined Scenarios

The following six figures (Figure 4.7.1-I to 4.7.1-VI) contain screenshots of the VANETsim GUI with integrated trust simulation, and illustrate the system's reactions to the scenarios defined in section 4.5.

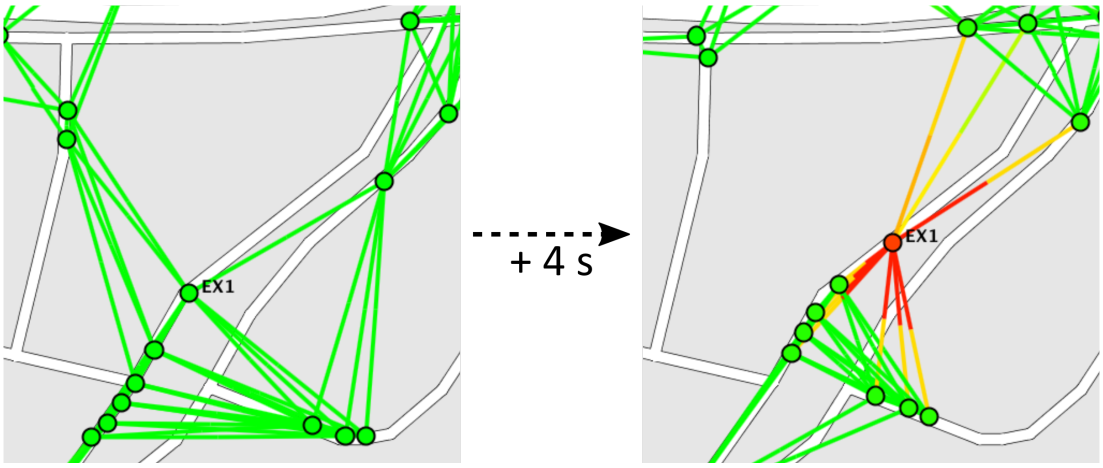


Figure 4.7.1-I: Scenario 1 (Exploiter1). The adversary node quickly loses all local influence. Nodes directly outside of its local environment are already prepared by means of trust propagation and are already careful (low trust) before the adversary could harm them.

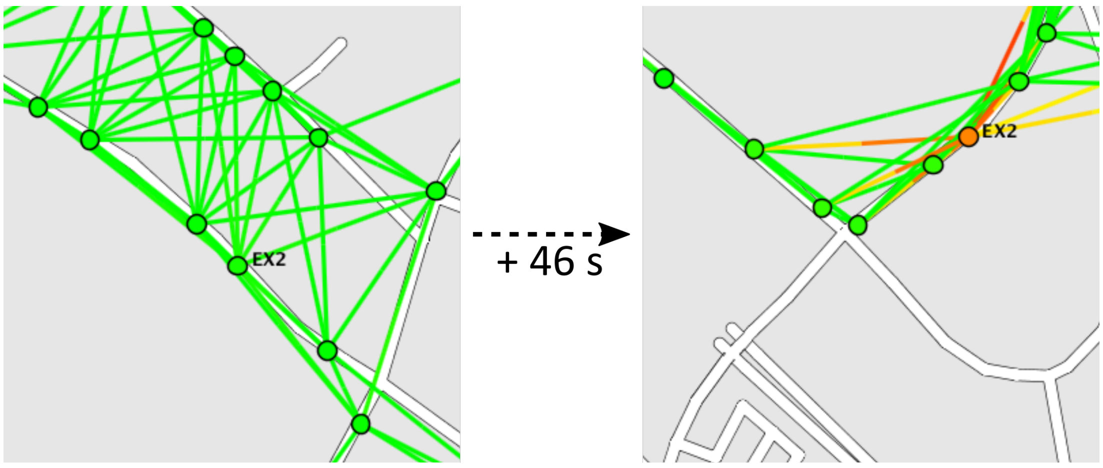


Figure 4.7.1-II: Scenario 2 (Exploiter2). The adversary node slowly loses local influence. Initially, other nodes would still cooperate with it, since they expect that it will later reimburse the system through increased cooperation. This does not happen, so after a while, cooperation with the adversary node is shut down.

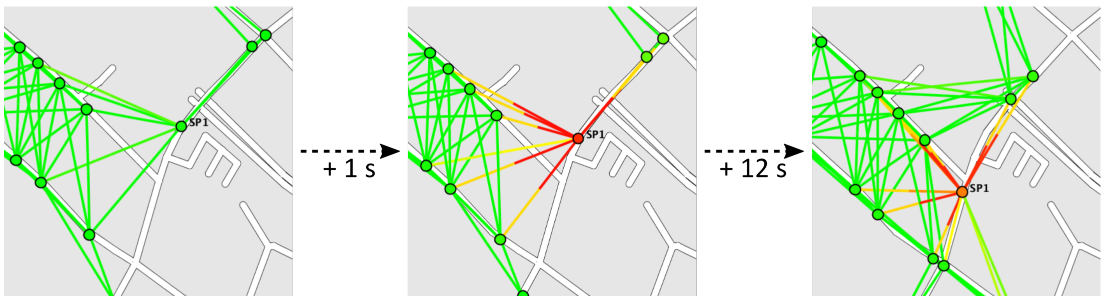


Figure 4.7.1-III: Scenario 3 (Spammer1). The adversary node immediately loses all local influence. Since it then becomes cooperative again, trust is restored after a while.

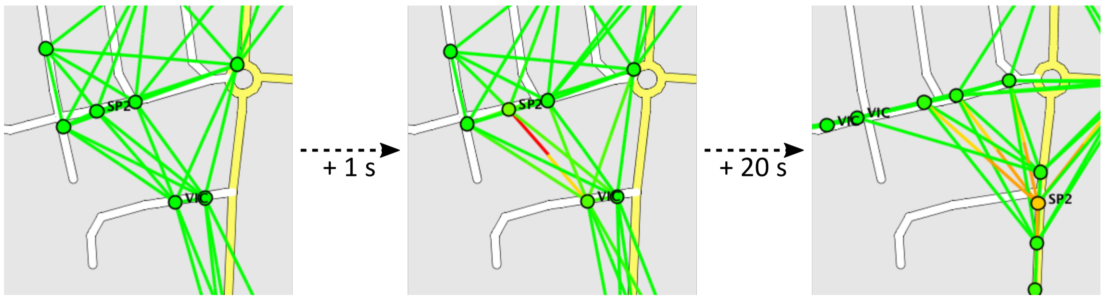


Figure 4.7.1-IV: Scenario 4 (Spammer2). The adversary node immediately loses the trust of its victim, but it can still interact with other local nodes. After later attacking other nodes, it loses local influence, even though the initial victim has moved away.

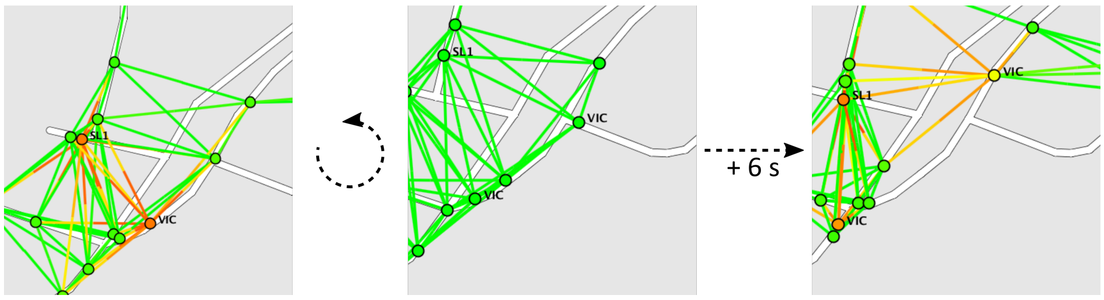


Figure 4.7.1-V: Scenario 5 (Slanderer1). After the initial attack on a single target, both the adversary node and its victim lose local influence (although the attacker can not maintain this state). After an initial attack on two targets, the victims lose less influence than the adversary and can recover much quicker.

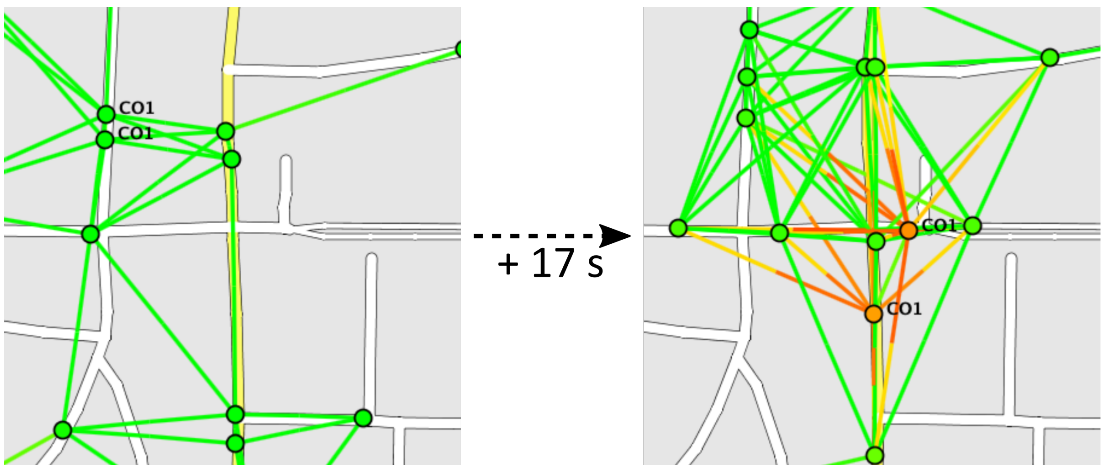


Figure 4.7.1-VI: Scenario 6 (Colluder1). Like in scenario 2, the adversary nodes can still cooperate with their local environment for some time. Nevertheless, their overly positive recommendations do not prevent their discovery; in fact, they accelerate it due to the distrust created by inaccurate recommendations.

4.7.2 Observed Effects (Scenarios)

4.7.2.1 Bubble of Distrust

Trust propagation surrounds an adversary node with a local bubble of distrust that is slightly larger than its local environment (see Figure 4.7.1-I). When the adversary moves, newly connected peers are already prepared; it is surrounded by a "bubble of distrust". A benefit of the trust system's decentralization is expressed here: remote nodes do not need to be affected when trust changes, whereas the local environment can be highly adaptive. This reduces propagation overhead while preventing attackers from "escaping" their bad reputation by moving.

4.7.2.2 Trust as a Currency

Temporarily taking an unfair amount of system utility does not exclude a node. It could still restore its status by means of increased cooperation (trade). Until then, its is slightly distrusted (debt). If it continues to reduce system utility, its influence eventually collapses to a point where it would need to provide much more utility than gained to recover (see Figure 4.7.1-II and -VI). This simplifies resource management (individual nodes are not "on a tight budget" and can have different demands) and also enables special usecases. For example, a vehicle in a hurry could ask its peers to assist it by giving way. If all vehicles do this at the same frequency, the system can operate; if they do not, the vehicles that request more can reimburse them through increased assistance with cooperative maneuvers etc.

4.7.2.3 Decentralized System Memory

Despite moving away from its initial victims, adversary nodes can not escape the accumulation of bad reputation. This is because the system memorizes their status in a decentralized way, and the "memory" is preserved through trust propagation. As shown in Figure 4.7.1-IV, the adversary node is not surrounded by a strong bubble of distrust, and its environment changes, but the doubt created by the (alleged) attack allows its peers to distrust it after they gain learn about a second attack.

4.7.2.4 Restoring Trust

Nodes that are recognized as untrustworthy by their local environment can recover by cooperating (see Figure 4.7.1-III). That this is also possible despite the fact that the node's behavior is clearly malevolent (Spammer1), is due to the indifference of the trust system towards the cause of distrust. As argued in section 3.1.3 and 3.5.3, this task should be undertaken by a special attack detection system. For example, a victim's internal attack detection system might forward the non-repudiable attack message to a peer (as proof of the attack). The peer's attack detection system could then inform its trust system to completely distrust the attacker without distrusting the victim, while informing authorities.

4.7.2.5 Power of Majority Tampering Attackers

An adversary node, or a group of them, can temporarily destroy a victim's influence at the price of their own influence, provided that they initially hold more influence than the victim(s) (see Figure 4.7.1-V). However, the attack can not be maintained if the victim does not actually misbehave. Further broadcasts of negative recommendations will be less effective since the adversary is already distrusted.

4.7.2.6 Race Conditions

In Figure 4.7.1-V, which of the victims shown in the third screenshot recovers much faster than the other is determined by a race condition; they are attacked almost simultaneously. The second attack has much less impact since the attacker is already distrusted.

If we had implemented a short-term memory component, the other peers could reevaluate both negative recommendations together when they receive the second one. As a result, trust in the first victim would be restored (since the first recommendation would have less impact) and it would also recover faster.

4.7.3 Other Observed Effects (Different Simulation Parameters)

4.7.3.1 Distrust in Relative Outsiders

We adjusted the simulation to include a tenfold increase in the frequency of random push propagation (simulating more pull propagation). We observed the following effect:
Occasionally, a node that entered a local environment, that had been stable for some time, was quickly distrusted by every other node (see Figure 4.7.3.1-I). This happened only when the new node had been completely disconnected (or only connected through long paths) from that environment earlier.

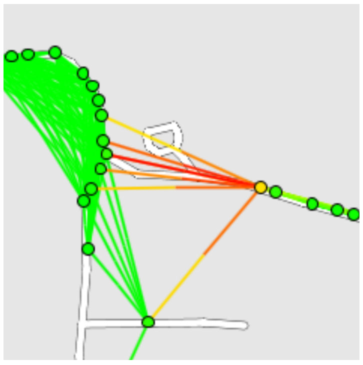


Figure 4.7.3.1-I: Distrust in a relative outsider

The reason for is effect is that all other nodes within the group gained high local influence, but the new node did not "know" them or this development and therefore put only moderate trust in them. When the new node issued recommendations, the environment reacted as if it was a tampering attacker. For the same reason, the new node distrusted the other nodes.
This problem could be easily solved by using a trust space that is more than one-dimensional and includes a dimension of certainty; for example, the opinion space of subjective logic ([JOSA01]). When receiving an inquiry, the new node could then simply issue an honest trust statement with low certainty , without receiving as much distrust.
By also including intermediary trust propagation, one could enable the other nodes to justify their recommendations with each others earlier statements, so that the new node does not suspect a tampering attack.

4.7.3.2 "Tit-for-Tat" Vicious Cycle

We removed the minimal base cooperation that benevolent nodes offered from the simulation. As a consequence of this, nodes that received asymmetric opinions of each other, for example, when only one of them observed an attack, would continue to lose all trust between them.
When the more trusted node decreased cooperation, the other node regarded this is as unfair and also decreased cooperation, causing the first node to trust it even less, and so

on. This stresses the fact that individuals in a trust system need to be slightly more benevolent than employing a “tit-for-tat”-strategy when interacting with others.

4.8 Conclusion

We implemented a drastically simplified simulation of a decentralized trust system in a VANET environment. We could observe the expected benefits of such a system and demonstrate its stability when confronted with various simple attack scenarios. Nevertheless, our simulation can only serve illustrative purposes since it does not offer a good representation of realistic behavior and decision-making in a VANET. Moreover, it does not fully meet the requirements on a trust system, even in the simplified environment. We did not simulate newcomer attacks or vehicles that only just entered the system, and instead assumed that all simulated vehicles were already in a moderately trusted state. In our simulation, we would have to chose between enabling newcomer attacks, or exposing new benevolent nodes to the risks described in section 4.7.3.1.

On the other hand, most instabilities of the system can be explained by the lack of those features we proposed in section 3.5, but were not able simulate.

We conclude that push propagation and trust coherency solving can provide a trust system with improved resistance against some attack scenarios; however, the designer of a trust system should also include means of pull (and possibly also intermediary) trust propagation, as well as a trust space that has a dimension of certainty. Agent designers should implement a short-term memory component to enable more sophisticated solving of trust coherency problems. To defend against specific types of attacks, the trust system should be coupled with an additional attack detection system.

We illustrated some of the benefits that the implementation of a dynamic and decentralized trust system could offer in a VANET. Attack prevention, detection and containment benefit from locality and subjective views. Besides that, the normal operation without malicious nodes could also be improved since the use of trust as a currency facilitates cooperative resource management and assists with cooperative maneuvers.

5 Future Outlook

5.1 V2X Integration

In section 1.1, we envisioned future applications in the Internet of Things and in V2X communications. Neither the Internet of Things nor large-scale VANETs exist today, although both are already being designed, and infrastructure is being prepared for them. We illustrated some of the benefits that the integration of a DTS in a VANET could bring, but a future implementation seems unlikely; firstly, because such a system might be incompatible with privacy requirements (see section 3.4.2); secondly, because the planning and design of V2X systems has been in progress for more than ten years and it does not include a similar component yet.

Nevertheless, one could imagine the implementation of a DTS alongside the currently planned security mechanisms. While these mechanisms already aim to provide protection against attacks, a DTS would still enable improved resource management that uses trust as a currency, and stimulate cooperation of vehicular services. A manufacturer could implement functionality without the requirement that it provides sufficient direct benefit to own vehicles; a service that is offered to other nodes (such as improved message routing) would provide indirect benefit by increasing the influence of own nodes in the trust system; these nodes would then receive increased cooperation from their peers. Decentralized trust management could also solve some challenges of V2X systems that we did not investigate in section 4. For example, the obstacle of combining the interests of all included companies and agencies into global standards would be significantly lowered, if multiple vendors could implement systems that do need not be fully compatible with each other; instead, they could form multiple subnets that only cooperate in some functional aspects at first, but are gradually integrated with the whole system as trustworthiness increases. This possibility of dynamic and fine-grained trust would also give the agent designers more freedom regarding new or proprietary features, which removes the possible stagnancy of a controlled and planned economy and instead boosts competition and progress. Lastly, the quality of vehicular agents would be easier to control, since a decentralized trust system would implicitly measure their usefulness for the entire system (via their objective reputation). Not only could this be utilized for evaluating features, but it also provides additional incentive for agent designers to improve that quality since nodes that do not increase system utility are punished by receiving less influence.

5.2 Other Potential Applications

5.2.1 Decentralized Social Network

Common concerns with “traditional” social network services are the lack of privacy among users and the handling of the massive amount of personal data by the service provider. Firstly, trust relations between users are usually binary: you are either the friend of a person or not. Some services allow their users to fine-tune which personal data is seen by which group of friends, but extensive specification of such rules is tedious and complicated, and therefore rarely employed. As a consequence, it is often recommended to users of social networks to carefully decide which personal data should be “put online”.
Secondly, the user is never in full control of their data, since it is commonly stored on a central server[133]. The service provider could theoretically sell this data to anyone; it is almost always sold to advertisers. While the user needs to agree to this by accepting Terms and Conditions, they are often driven by social pressure, since not accepting would render them unable to communicate and socialize in a digitalizing society.
A future social network could be fully decentralized and solve the aforementioned problems using trust management. Intelligent nodes (representing users) would send status updates and messages directly to other nodes in a peer-to-peer network. A remaining obstacle is the possibility that a node could never receive another node’s updates, if they are never online at the same time. A DTS could identify common trusted peers that are authorized to store and later forward personal data, or suggest new contacts to the user. There are already existing decentralized social network services such as Diaspora*. With the implementation of trust management, (preferably open-source) software agents could manage the distribution of a user’s personal data on their behalf.

5.2.2 Software Collaboration

Autonomous software services could utilize trust propagation and trust coherency solving to identify trustworthy peers during service discovery. This would be especially useful for dynamic smart environments in the Internet of Things. Future artificial intelligences could also perform complex tasks while being protected from abuse. For example, networked robots deployed in disaster areas could distinguish other helpers from looters. Other fields for networked robots that could benefit from trust management are swarm robotics and military applications ([VAND12]).
Peer-to-peer networks that include resource management could be an immediate area of application. File-sharing networks based on protocols such as BitTorrent often utilize the so-called “ratio”, i.e., the quotient of uploaded and downloaded data, to exclude unco-operative users. There are other factors that might lead to the untrustworthiness of a user, such as the distribution of spam and malware. Using trust propagation and trust coherency

[133] In the logical sense; the system is usually physically distributed.

solving, clients could manage resource distribution (network traffic) and exclusion of untrustworthy peers on their own, without the help of a central server[134].
Another example of trust-related software interaction is email. The email system has been (and still is) confronted with heavy abuse by spammers. As a consequence, it is also quite complicated to set up a new mail server, due to of the risk of being placed on blacklists. On the other hand, if mail servers are mostly owned by large corporations, those corporations possess complete control over the contained data. The whole process would be simplified if mail servers utilized some sort of trust propagation, keeping track of spam reports individually while also reacting to recommendations (warnings).
In general, any kind of open multi-agent system could employ schemes such as the one we proposed in 3.5 to manage protection and cooperation.

5.2.3 Sociological, Economical and Evolutionary Study of MASs

Trust plays an important role in social and economic systems. Trust management schemes could be utilized in the creation of such systems, similar to the subject of our investigation. However, the inverse operation should not be overlooked: preexisting multi-agent systems could be modeled based on trust systems; these models could then be used to make predictions about the future development of the systems. This is based on the assumption that trust propagation exists in these systems. By choosing an appropriate trust space and trust combination operator[135], trust-related agent behavior could be approximated. For example, one might research how an advertising character's recommendations are affected by other recommendations for different products by the same character.
Another area of research where trust models could find new applications in is the study of multi-agent organizations. Organizational teamwork includes, among others, the formation of teams, the negotiation of responsibilities, and the controlling of the team's operation. In all of these operations, a trust system could be employed; for the formation, sufficient trust between the forming agents is necessary; for negotiation, trust as a currency could be utilized to trade tasks, access to resources, or claims on another agent's work effort; and lastly, for controlling, agents could utilize the trust system to keep track of each other's performance and cooperation within their respective fields of duty. After convergence of the trust graph, the resulting network can be utilized to optimize or rebuild the organizational structure.
In the field of evolutionary computation, trust propagation and coherency solving could be utilized to include an agent's effectiveness in cooperative tasks into a fitness rating, while at the same time preventing the domination of agents that simply provide unfair negative recommendations for others.

[134] As pointed out in 3.3.2, a centralized TA does not only raise privacy concerns, but could also be unable to represent complex trust relations; users might have different perceptions of which files should be removed from the network.
[135] Which might operate on more arguments than those we investigated, to solve higher-level constraints.

6 Conclusions

In retrospect, we mostly reached the goals we set ourselves in section 1.2.
In chapter 2, we provided the reader with a short introduction to V2X communications, and to its remaining obstacles which we partially aimed to solve.
In chapter 3, we gained a deeper understanding of trust in the context of multi-agent systems through surveys of existing literature as well as original research, which culminated in the design and informal specification of a generic abstract trust system (GTS). Some interesting results we gained along the way include:

- That the need for entity trust can be derived from the rational considerations of an open open multi-agent system's system designer (section 3.1.6)
- That the traditionally separated concepts of transitive trust propagation (adaption) and protection from slander attacks can be unified into a single concept (trust coherency problem) and operation (trust coherency solving), which can also replace inconsistent definitions of trust transitivity (section 3.1.7)
- That trust propagation should be further differentiated and specified in two central documents that form the foundation of an open MAS (section 3.2.4)

and

- That trust can be used as a currency to enable dynamic and adaptive resource management even without communication (section 3.2.4).

In chapter 4, we designed and implemented a simulation software that we also used to investigate the properties of our system in V2X communications. However, we were not able to simulate the full functionality of the GTS; despite that, we could observe most of the desired effects.
Lastly, in chapter 5, we listed some potential future areas of application for decentralized trust management systems.
To conclude: the research and design based on symbolic representations of trust in multi-agent systems remains an interesting and maybe underrepresented field, whose importance is nonetheless growing with the approach of the Internet of Things. Decentralized trust management can solve many of the problems that were encountered with traditional security systems in dynamic environments.

7 Acknowledgements

At this point, I would like to express gratitude to my fellow colleagues at NXP Semi-conductors who gave me the opportunity to conduct the research I summarized in this paper. They also provided me with an insight into the field of V2X technology and feedback to my ideas. I would also like to thank my thesis supervisors Prof. Dr. Christoph Klauck and Dipl.-Ing. Frank Siedel for their critical examination of my work, which has considerably contributed to it.

8 References

[RFC5280] COOPER, D.; SANTESSON, S.; FARRELL, S.; BOEYEN, S.; HOUSLEY, R.; POLK, W.: *RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. 2008

[ITUX680] INTERNATIONAL TELECOMMUNICATIONS UNION: *Information technology-Abstract Syntax Notation One (ASN.1): Specification of basic notation* (ITU-T Recommendation X.680). ITU, 2008.

[ABDU97] ABDUL-RAHMAN, Alfarez; HAILES, Stephen: A distributed trust model. In: *NSPW '97 Proceedings of the 1997 workshop on New security paradigms*. New York: ACM, 1997. pp 48-60

[ABDU00] ABDUL-RAHMAN, Alfarez; HAILES, Stephen: Supporting Trust in Virtual Communities. In: *HICSS '00 Proceedings of the 33rd Hawaii International Conference on System Sciences*, Vol. 6. Washington DC, USA: IEEE, 2000.

[ADAM05] ADAMS, William J.; DAVIS, Nathaniel J.: Toward a Decentralized Trust-based Access Control System for Dynamic Collaboration. In: IAW '05 Proceedings from the Sixth Annual IEEE SMC. IEEE, 2005. pp 317-324

[ALKA12] AL-KAHTANI, Mohammed S.: Survey on Security Attacks in Vehicular Ad hoc Networks. In: *6th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, 2012. pp 1-9

[ARTI09] ARTIKIS, Alexander; KAPONIS, Dimosthenis; PITT, Jeremy: Dynamic Specifications for Norm-Governed Systems. In: *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global, 2009. pp 460-479

[AXEL84] AXELROD, Robert M.: *The evolution of cooperation*. New York: Basic Books, 1984.

[BENT76] BENTHAM, Jeremy: *A Fragment on Government*. Accessed 2015-07-19, URL: http://www.constitution.org/jb/frag_gov.htm

[BISM12] BIßMEYER, Norbert; NJEUKAM, Joël; PETIT, Jonathan; BAYAROU, Kpatcha M.: Central misbehavior evaluation for VANETs based on mobility data plausibility. In: *VANET '12 Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications*. New York: ACM, 2012. pp 73-82

[BLAZ96] BLAZE, Matt; FEIGENBAUM, Joan; LACY, Jack: Decentralized Trust Management. In: *SP '96 Proceedings of the 1996 IEEE Symposium on Security and Privacy*. Washington DC, USA: IEEE, 1996. pp 164-173

[BUTT01] BUTTYÁN, Levante; HUBAUX, Jean-Pierre: Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. In: *Technical Report No. DSC/2001/001*. Lausanne, Switzerland: Swiss Federal Institute of Technology, 2001.

[CHEN10] CHEN, Chen; ZHANG, Jie; COHEN, Robin; HO, Pin-Han. A Trust Modeling Framework for Message Propagation and Evaluation in VANETs. In: *2nd International Conference on Information Technology Convergence and Services (ITCS)*. IEEE, 2010. pp 1-8

[CHOJ11] CHO, Jin-Hee: Towards Trust-based Cognitive Networks:
A Survey of Trust Management for Mobile Ad Hoc Networks. In: *Communications Surveys & Tutorials*, Vol. 13, No. 4. IEEE, 2011. pp 562-583

[CHRI13] CHRISTIANSON, B.: Living in an Impossible World: Real-izing the Consequences of Intransitive Trust. In: *Philosophy and Technology*, Vol. 26, no. 4. 2013. pp 411-429

[COMO11] COMODO GROUP INC.: *Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011*. 2011. Accessed 2015-09-21, URL: https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

[DELL00] DELLAROCAS, Chrysanthos: Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. In: *EC '00 Proceedings of the 2nd ACM conference on Electronic commerce*. New York: ACM, 2000. pp 150-157

[DENN93] DENNING, Dorothy E.: A new paradigm for trusted systems. In: *NSPW '92-93 Proceedings on the 1992-1993 workshop on New security paradigms*. New York: ACM, 1993. pp 36-41

[DOTZ05] DÖTZER, Florian; FISCHER, Lars; MAGIERA, Przemyslaw: VARS: A Vehicle Ad-Hoc Network Reputation System. In: *WOWMOM '05 Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*. Washington DC, USA: IEEE, 2005. pp 454-456

[DOUC02] DOUCEUR, John R.: The Sybil Attack. In: *IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London: Springer, 2002. pp 251-260

[ETSI09] European Telecommunications Standards Institute: *ETSI TR 102 638 V1.1.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*. 2007. Accessed 2015-10-14, URL: http://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf

[GAMB00] GAMBETTA, Diego: Can We Trust Trust?. In: GAMBETTA, Diego (ed.): *Trust: Making and Breaking Cooperative Relations*. Electronic edition. Department of Sociology, University of Oxford, 2000. pp 213-237. URL: http://www.sociology.ox.ac.uk/papers/gambetta213-237.pdf. Accessed using the Internet Wayback Machine, version of 2007-03-06

[GERL07] GERLACH, Matthias: Trust for Vehicular Applications. In: *ISADS '07 Eighth International Symposium on Autonomous Decentralized Systems*. Berlin: Fraunhofer FOKUS, 2007. pp 295-304

[GILB02] GILBERT, Seth; LYNCH, Nancy: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. In: *ACM SIGACT News*, Vol. 33, Issue 2. New York: ACM 2002. pp 51-59

[GOLL04] GOLLE, Philippe; GREENE, Dan; STADDON, Jessica: Detecting and correcting malicious data in VANETs. In: *VANET '04 Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. New York: ACM, 2004. pp 29-37

[GROV11] GROVER, Jyoti; GAUR, Manoj Singh; LAXMI, Vijay; PRAJAPATI, Nitesh Kumar: A sybil attack detection approach using neighboring vehicles in VANET. In: *SIN '11 Proceedings of the 4th international conference on Security of information and networks*. New York: ACM, 2011. pp 151-158

[GUTM02] GUTMANN, Peter: PKI: It's Not Dead, Just Resting. In: *Computer*, Vol. 35, Issue 8. Los Alamitos, USA: IEEE, 2002.pp 41-49

[GUTM11] GUTMANN, Peter: *Diginotar broken arrow as a tour-de-force of PKI fail* [Electronic mailing list message]. 2011. Accessed 2015-09-21, URL: http://comments.gmane.org/gmane.comp.security.cryptography.randombit/1215

[JOSA96] JØSANG, Audun: The right type of trust for distributed systems. In: *NSPW '96 Proceedings of the 1996 workshop on New security paradigms*. New York: ACM, 1996. pp 119-131

[JOSA01] JØSANG, Audun: A logic for uncertain probabilities. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 9 Issue 3. River Edge, NJ, USA: World Scientific Publishing Co., 2001. pp 279-311

[JOSA04] JØSANG, Audun; LO PRESTI, Stéphane: Analysing the Relationship between Risk and Trust. In: *Trust Management: Proceedings of the 2nd International Conference iTrust 2004*. Berlin: Springer, 2004. pp 135-145

[JOSA05] JØSANG, Audun; KESER, Claudia; DIMITRAKOS, Theo: Can we manage trust?. In: *iTrust'05 Proceedings of the Third international conference on Trust Management*. Heidelberg: Springer, 2005. pp 93-107

[JOS205] JØSANG, Audun; POPE, Simon: Semantic constraints for trust transitivity. In: *APCCM '05 Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*. Darlinghurst, Australia: Australian Computer Society Inc., 2005. pp 59-68

[JOSA07] JØSANG, Audun: Trust and Reputation Systems. In: ALDINI, Allesandro (ed.); GORRIERI, Roberto (ed.): *Foundations of Security Analysis and Design IV*. Heidelberg: Springer, 2007. pp 209-245

[KAMV03] KAMVAR, Sepandar D.; SCHLOSSER, Mario T.; GARCIA-MOLINA, Hector: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: *WWW '03 Proceedings of the 12th international conference on World Wide Web*. New York: ACM, 2003. pp 640-651

[KANT85] KANT, Immanuel; ELLINGTON, James W. (transl.): *Grounding or the Metaphysics of Morals 3rd ed.* (Orig. 1785). Indianapolis, USA: Hackett, 1993.

[KHUN14] KHUN, Steven: Prisoner's Dilemma. In: ZALTA, Edward N. (Bearb.): *The Stanford Encyclopedia of Philosophy (Fall 2014 Edition)*. 2014. URL (statisch): http://plato.stanford.edu/archives/fall2014/entries/prisoner-dilemma/. Abrufdatum: 2015-07-19

[MANC98] MANCHALA, Daniel W.: Trust Metrics, Models and Protocols for Electronic Commerce Transactions. In: *Proceedings of 18th International Conference on Distributed Computing Systems 1998*. El Segundo, CA, USA: Xerox Corp., 1998. pp 312-321

[MARS94] MARSH, Stephen P.: *Formalising Trust as a Computational Concept*. University of Stirling, 1994.

[MART00] MARTI, Sergio; GIULI, T.J.; LAI, Kevin; BAKER, Mary: Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In: *MobiCom '00 Proceedings of the 6th annual international conference on Mobile computing and networking*. New York: ACM, 2000. pp 255-265

[MASH11] MA, Shuo; WOLFSON, Ouri; LIE, Jin: A Survey on Trust Management for Intelligent Transportation System. In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. New York: ACM, 2011. pp 18-23

[MCKN96] MCKNIGHT, D. Harrison; CHERVANY, Norman L.: The Meanings of Trust. In: *Technical Report MISRC Working Paper Series 96-04*. University of Minnesota, Management Information Systems Reseach Center. Accessed 2015-06-27, URL: http://misrc.umn.edu/workingpapers/fullPapers/1996/9604_040100.pdf

[MERR03] *Merriam-Webster's Collegiate Dictionary*. 11th ed. Springfield, MA: Merriam-Webster, 2003. Also available at http://www.merriam-webster.com/

[MICH02] MICHIARDI, Pietro; MOLVA, Refik: CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks. In: *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security*. Deventer, NL: Kluwer BV, 2002. pp 107-121

[MINH10] MINHAS, Umar F.; ZHANG, Jie; TRAN, Thomas; COHEN, Robin: Towards expanded trust management for agents in vehicular ad-hoc networks. In: *International Journal of Computational Intelligence: Theory and Practice*, Vol. 5, no. 1. Serials Publications, 2010.

[NAKA08] NAKAMOTO, Satoshi: *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. Accessed 2015-09-11, URL: https://bitcoin.org/bitcoin.pdf

[NHTS14] National Highway Traffic Safety Administration: HARDING, John; POWELL, Gregory; YOON, Rebecca; FIKENTSCHER, Joshua; DOYLE, Charlene; SADE, Dana; LUKUC, Mike; SIMONS, Jim; WANG, Jing: *Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application* (Report No. DOT HS 812 014). Washington DC, USA: 2014.

[PERR13] PERRY, Mike: *[tor-talk] Why the Web of Trust Sucks* [Electronic mailing list message]. 2013. Accessed 2015-09-06, URL: https://lists.torproject.org/pipermail/tor-talk/2013-September/030235.html

[PRIN11] PRINS, J. R.: *Fox-IT Interim Report: DigiNotar Certificate Authority breach "Operation Black Tulip"*. 2011. Accessed 2015-09-21, URL: https://www.rijksoverheid.nl/binaries/rijksoverheid/documenten/rappoten/2011/09/05/diginotar-public-report-version-1/rapport-fox-it-operation-black-tulip-v1-0.pdf

[PUKA15] PUKA, Bill: The Golden Rule. In: *The Internet Encyclopedia of Philosophy*. Accessed 2015-07-19, URL: http://www.iep.utm.edu/goldrule/

[RANG98] RANGAT, P. Venkat: An axiomatic basis of trust in distributed systems. In: *SP'88 Proceedings of the 1988 IEEE conference on Security and privacy*. Washington DC, USA: IEEE, 1988. pp 204-211

[RAYA06] RAYA, Maxim; PAPADIMITRATOS, Panos; HUBAUX, Jean-Pierre: Securing Vehicular Communications. In: *IEEE Wireless Communications*, Vol. 13, no. 5. IEEE, 2006. pp 8-15

[RAYA07] RAYA, Maxim; HUBAUX, Jean-Pierre: Securing vehicular ad hoc networks. In: *Journal of Computer Security - Special Issue on Security of Ad-hoc and Sensor Networks*, Vol. 15, Issue *1*. Amsterdam: IOS Press, 2007. pp 39-68

[REPA06] REPANTIS, Thomas; KALOGERAKI, Vana: Decentralized Trust Management for Ad-Hoc Peer-to-Peer Networks. In: *MPAC '06 Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing*. New York: ACM, 2006.

[SABA02] SABATER, Jordi; SIERRA, Carles: Reputation and Social Network Analysis in Multi-Agent Systems. In: *AAMAS '02 Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. New York: ACM, 2002. pp 475-482

[SCHM08] SCHMIDT, Robert K; LEINMÜLLER, Tim; BÖDDEKER, Bert: V2X Kommunikation. In: *Proceedings of the 17th Aachener Kolloquium*. Aachen, Germany: 2008.

[SURY06] SURYANARAYANA, Girish; DIALLO, Mamadou H.; ERENKRANTZ, Justin R.; TAYLOR, Richard N.: Architectural support for trust models in decentralized applications. In: *ICSE '06 Proceedings of the 28th international conference on Software engineering*. New York: ACM, 2006. pp 52-61

[THEO06] THEODORAKOPOULOS, George; BARAS, John S.: On Trust Models and Trust Evaluation Metrics for Ad Hoc Networks. In: *IEEE Journal on Selected Areas in Communications*, Vol. 24, Issue 2. IEEE, 2006. pp 318-328

[TOMA14] TOMANDL, Andreas; HERRMANN, Dominik; FUCHS, Karl-Peter; FEDERRATH, Hannes; SCHEUER, Florian: VANETsim: An open source simulator for security and privacy concepts in VANETs. In: *Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2014. pp 543-550

[USDO15] United States Department of Transportation, Office of the Assistant Secretary for Research and Technology: *Vehicle-to-Infrastructure (V2I) communications for safety*. Accessed 2015-10-14, URL: http://www.its.dot.gov/factsheets/v2isafety_factsheet.htm

[VAND12] VANDENBERGE, Wim; MOERMAN, Ingrid; DEMEESTER, Piet: Adoption of Vehicular Ad Hoc Networking Protocols by Networked Robots. In: *Wireless Personal Communications: An International Journal*, Vol. 64, Issue 3. Hingham, MA, USA: Kluwer, 2012. pp 489-522

[WEIZ14] WEI, Zhexiong; YU, Fei R.; BOUKERCHE, Azzedine: Trust Based Security Enhancement for Vehicular Ad Hoc Networks. In: *DIVANet '14 Proceedings of the fourth ACM international symposium on Development and analysis of intelligent vehicular networks and applications*. New York: ACM, 2014. pp 103-109

[ZHAN11] ZHANG, Jie: A Survey on Trust Management for VANETs. In: *AINA '11 Proceedings of the 2011 IEEE International Conference on Advanced Information Networking and Applications*. Washington DC, USA: IEEE, 2011. pp 105-112

[ZHAO04] ZHAOYU, Liu; JOY, Anthony W.; THOMPSON, Robert A.: A Dynamic Trust Model for Mobile Ad Hoc Networks. In: *FTDCS 2004. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*. IEEE, 2004. pp 80-85

[ZIMM95] ZIMMERMANN, P.R.: *The Official PGP User's Guide*. MIT Press, 1995.

9 Appendices

This thesis is handed in for review alongside a CD. That CD contains the following attached files and directories:

- The GenericTrustSystem project (Eclipse: Java)
- The TrustCoherencySolver project (Eclipse: Java)
- The modified VANETsim project (Eclipse: Java)
- A PDF version of this document

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den ________________ ____________________________