Simplification and Analysis of Transitive Trust Networks

Audun Jøsang[*], Elizabeth Gray[†], and Michael Kinateder[‡]
[*]School of Software Engineering and Data Communications
Queensland University of Technology, Australia
a.josang@qut.edu.au
[†]Computer Science Department
Trinity College Dublin, Ireland
grayl@cs.tcd.ie
[‡]Institute of Parallel and Distributed Systems
Faculty of Computer Science, University of Stuttgart, Germany
Michael.Kinateder@informatik.uni-stuttgart.de

Abstract

When transacting and interacting through open computer networks, traditional methods used in the physical world for establishing trust can no longer be used. Creating virtual network substitutes with which people, organisations and software agents can derive trust in other parties requires computerised analysis of the underlying trust networks. This article describes an approach to trust network analysis using subjective logic (TNA-SL), that consists of the three following elements. Firstly it uses a concise notation with which trust transitivity and parallel combination of trust paths can be expressed. Secondly it defines a method for simplifying complex trust networks so that they can be expressed in this concise form. Finally it allows trust measures to be expressed as beliefs, so that derived trust can be automatically and securely computed with subjective logic. We compare our approach with trust derivation algorithms that are based on normalisation such as PageRank and EigenTrust. We also provide a numerical example to illustrates how TNA-SL can be applied.

Index Terms

Trust, transitivity, networks, reputation, subjective logic, belief, DSPG, TNA-SL, P2P, PageRank, EigenTrust.

I. Introduction

Modern communication media are increasingly removing us from familiar styles of interacting and conducting business in ways that traditionally rely on some degree of pre-established trust between business partners. Moreover, most traditional cues for assessing trust in the physical world are not available through those media. We may now be conducting business with people and organisations of which we know nothing, and we are faced with the difficult task of making decisions involving risk in such situations. As a result, the topic of trust in open computer networks is receiving considerable attention in the network security community and e-commerce industry [1], [2], [3], [4], [5], [6], [7]. State of the art technology for stimulating trust in e-commerce includes cryptographic security mechanisms for providing confidentiality of communication and authentication of identities. However, merely having a cryptographically certified identity or knowing that the communication channel is encrypted is not enough for making informed decisions if no other knowledge about a remote transaction partner is available. Trust therefore also applies to the truthfulness of specific claims made by parties who request services in a given business

[*] The work reported in this article has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science, and Training).
[*] Support from the ARC Research Network Secure Australia acknowledged.
[†] This work has been supported in part by the SECURE project (IST-2001-32486), a part of the EU FET Global Computing initiative.

context as described in the WS-Trust specifications [5], and trust between business partners regarding security assertions as described in the Liberty Alliance Framework [6], [7]. Trust also applies to the honesty, reputation and reliability of service providers or transaction partners, in general or for a specific purpose. In this context, the process of assessing trust becomes part of quality of service (QoS) evaluation, decision making and risk analysis.

Being able to formally express and reason with these types of trust is needed not only to create substitutes for the methods we use in the physical world, like for instance trust based on experiences or trust in roles, but also for creating entirely new methods for determining trust that are better suited for computer networks. This will facilitate the creation of communication infrastructures where trust can thrive in order to ensure meaningful and mutually beneficial interactions between players.

The main contribution of this article is a notation for specifying trust networks consisting of multiple paths between the relying party and the trusted party, and a practical method for analysing and deriving measures of trust in such environments. Our method, which is called TNA-SL (Trust Network Analysis with Subjective Logic), is based on analysing trust networks as directed series-parallel graphs that can be represented as canonical expressions, combined with measuring and computing trust using subjective logic. In order to provide a comparison with other methods, we briefly describe proposals for trust network analysis that do not require canonical expressions, but instead use the principle of trust measure normalisation. We finally provide a numerical example of how trust can be analysed and computed using our method.

II. Trust Diversity

Trust facilitates interaction and acceptance of risk in situations of incomplete information. However, trust is a complex concept that is difficult to stringently define. A wide variety of definitions of trust have been put forward [8], many of which are dependent on the context in which interaction occurs, or on the observer's subjective point of view. For the purpose of this study the following working definition inspired by McKnight & Chervany [8] will be used:

> *Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.*

This definition explicitly and implicitly includes the basic ingredients of trust which are *dependence*, *risk* and *uncertainty*. The aspect of dependence is explicitly expressed whereas risk and uncertainty are expressed through the possibility of negative consequences. This definition also illustrates that abstract and inert material things can also be trusted, even though they do not have free will to behave honestly or dishonestly in the way that humans do.

Thus, we may say that trust is related to belief in the honesty, reliability, competence, willingness, etc. of the trusted entity, it being a person, organisation, or system. Trust can also be related to a particular property of material or abstract objects such as a computer system or institutions. Despite this variation in meanings, many researchers simply assume a specific definition when using the term trust, where for example the common expression *"trusted public key"* actually refers to the *authenticity* of a cryptographic key used in a public-key system.

The fact that different entities can have different kinds of trust in the same target entity indicates that trust is subjective. It is also important to notice that trust is related to the scope of the relationship, e.g. an employee of a company might be trusted to deal with financial transactions up to a specific amount, but not to make public statements about the company. The term *trust scope*[1] will be used throughout this document to denote the specific type(s) of trust assumed in a given trust relationship.

In order to be more specific when talking about trust, classes of trust scopes have been defined, such as Grandison & Sloman's classification (2000) [9] illustrated in Fig.1. Each trust class is briefly described

[1]The terms "trust context" [9] and "trust purpose" [10] have been used in the literature with the same meaning.

below[2]. A given trust scope can thus be interpreted as an operational instantiation of trust from any of the classes. In other words, it defines the specific purpose and semantics of a given trust relationship. A particular trust scope can for example be *"to be a good car mechanic"*, which can be grouped under the provision trust class.

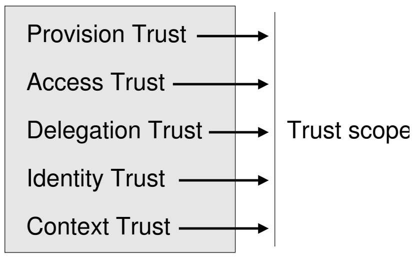


Fig. 1. Trust Classes according to Grandison & Sloman.

- **Provision trust** describes the relying party's trust in a service or resource provider, and is relevant when the relying party is a user seeking protection from malicious or unreliable service providers. The Liberty Alliance Project[3] uses the term "business trust" [7] to describes mutual trust between companies emerging from contract agreements that regulate interactions between them, and this can be interpreted as provision trust. For example, when a contract specifies quality requirements for the delivery of services, then this business trust would be provision trust in our terminology.
- **Access trust** describes trust in principals for the purpose of accessing resources owned by or under the responsibility of the relying party. This relates to the access control paradigm which is a central element in computer security. A good overview of access trust systems can be found in Grandison & Sloman (2000) [9].
- **Delegation trust** describes trust in an agent (the delegate) that acts and makes decision on behalf of the relying party. Grandison & Sloman point out that acting on one's behalf can be considered to a special form of service provision.
- **Identity trust**[4] describes the belief that an agent identity is as claimed. Trust systems that derive identity trust are typically authentication schemes such as X.509 and PGP [11]. Identity trust systems have been discussed mostly in the information security community, and a brief overview and analysis can be found in Reiter & Stubblebine (1997) [12].
- **Context trust**[5] describes the extent to which the relying party believes that the necessary systems and institutions are in place in order to support the transaction, and to provide a safety net in case something should go wrong. Factors for this type of trust can for example be critical infrastructures, insurance, legal system, law enforcement and stability of society in general.

Conceptually, identity trust and provision trust can be seen as two layers on top of each other, where identity trust can exist alone and provision trust must be based on identity trust. In the absence of identity trust, it is only possible to have a baseline provision trust in an agent or a thing. The distinction between agents and non-living or abstract things can be blurred, for example in the case of automated systems. On one hand these systems can be classified as things because they behave deterministically and do not have free will of their own. On the other hand, they can be seen as extensions of their human masters who do exhibit free will.

In order to form meaningful trust networks, trust relationships must be expressed with three basic diversity attributes [13], where the first represents the *trustor* or trust source denoted by "S", the second

[2]Grandison & Sloman use the terms *service provision trust*, *resource access trust*, *delegation trust*, *certification trust*, and *infrastructure trust*.

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

[4]Called "authentication trust" in Liberty Alliance (2003) [7]

[5]Called "system trust" in McKnight & Chervany (1996) [8]

represents the *trustee* or the trust target denoted by “T”, and the third represents the *trust scope* denoted by “σ”. This is illustrated in Fig.2.

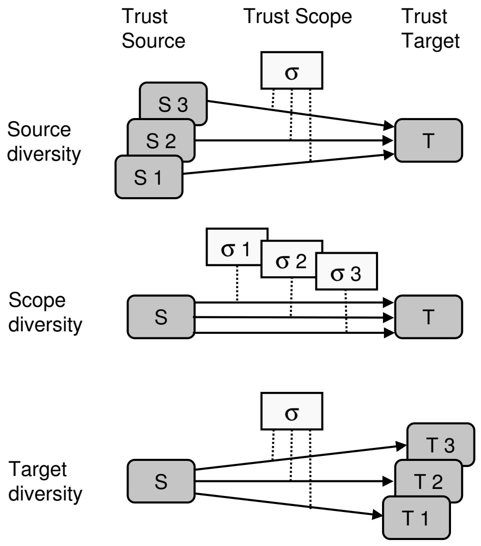


Fig. 2. Basic trust diversity

In addition to the three basic trust attributes, a *measure* can be associated with each trust relationship. The trust measure could for example be binary (trusted, not trusted), discrete (e.g. strong trust, weak trust, strong distrust, weak distrust, etc.) or continuous in some form, e.g. probability, percentage or belief functions.

In addition, a fifth important element to a trust relationship is its *time* component. Quite obviously the trustor’s level of trust in the trustee at one point in time might be quite different from the level of trust after several transactions between these two entities have taken place. This means, that we can model time as a set of discrete events taking place between the involved parties. However, even if no transactions take place, a trust relationship will gradually change with time passing. Therefore, in addition to the discrete changes that are made when events have occurred, we must also take into account continuous changes to trust relationships, as pointed out by Kinateder et al. in [14].

III. Trust Transitivity

Trust transitivity means, for example, that if Alice trusts Bob who trusts Eric, then Alice will also trust Eric. This assumes that Bob actually tells Alice that he trusts Eric, which is called a *recommendation*. This is illustrated in Fig.3, where the indexes indicate the order in which the trust relationships and recommendations are formed.

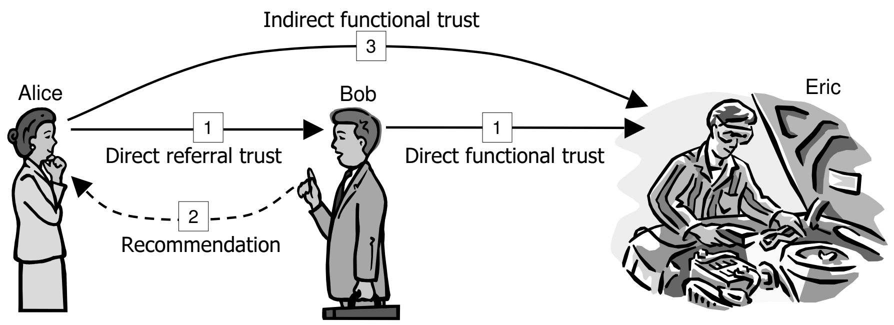


Fig. 3. Transitive trust principle

It can be shown that trust is not always transitive in real life [15]. For example the fact that Alice trusts Bob to look after her child, and Bob trusts Eric to fix his car, does not imply that Alice trusts Eric for looking after her child, or for fixing her car. However, under certain semantic constraints [16], trust can be transitive, and a trust system can be used to derive trust. In the last example, trust transitivity collapses because the scopes of Alice's and Bob's trust are different.

Based on the situation of Fig.3, let us assume that Alice needs to have her car serviced, so she asks Bob for his advice about where to find a good car mechanic in town. Bob is thus trusted by Alice to know about a good car mechanic and to tell his honest opinion about that. Bob in turn trusts Eric to be a good car mechanic.

It is important to separate between trust in the ability to recommend a good car mechanic which represents *referral trust*, and trust in actually being a good car mechanic which represents *functional trust*. The scope of the trust is nevertheless the same, namely to be a good car mechanic. Assuming that, on several occasions, Bob has proven to Alice that he is knowledgeable in matters relating to car maintenance, Alice's referral trust in Bob for the purpose of recommending a good car mechanic can be considered to be *direct*. Assuming that Eric on several occasions has proven to Bob that he is a good mechanic, Bob's functional trust in Eric can also be considered to be direct. Thanks to Bob's advice, Alice also trusts Eric to actually be a good mechanic. However, this functional trust must be considered to be *indirect*, because Alice has not directly observed or experienced Eric's skills in car mechanics.

Let us slightly extend the example, wherein Bob does not actually know any car mechanics himself, but he knows Claire, whom he believes knows a good car mechanic. As it happens, Claire is happy to recommend the car mechanic named Eric. As a result of transitivity, Alice is able to derive trust in Eric, as illustrated in Fig.4.

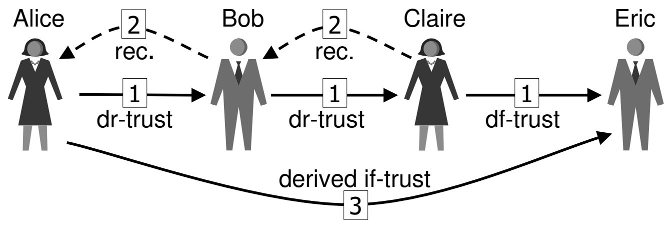


Fig. 4. Trust derived through transitivity

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

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

Definition 1 (Functional Trust Derivation Criterion): Derivation of functional trust through referral trust, requires that the last trust arc represents functional trust, and all previous trust arcs represents referral trust.

In practical situations, a trust scope can be characterised by being general or specific. For example, knowing how to change wheels on a car is more specific than to be a good car mechanic, where the former scope is a subset of the latter. Whenever the functional trust scope is equal to, or a subset of the referral trust scopes, it is possible to form transitive paths. This can be expressed with the following consistency criterion.

Definition 2 (Trust Scope Consistency Criterion): A valid transitive trust path requires that the trust scope of the functional/last arc in the path be a subset of all previous arcs in the path.

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

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

The examples above assume some sort of absolute trust between the agents in the transitive chain. In reality trust is never absolute, and many researchers have proposed to express trust as discrete verbal statements, as probabilities or other continuous measures. One observation which can be made from a human perspective is that trust is weakened or diluted through transitivity. Revisiting the above example, we note that Alice's trust in the car mechanic Eric through the recommenders Bob and Claire can be at most as strong as Claire's trust in Eric. This is illustrated in Fig.4.

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

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

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

IV. Parallel Trust Combination

It is common to collect advice from several sources in order to be better informed when making decisions. This can be modelled as *parallel trust combination* illustrated in Fig.5.

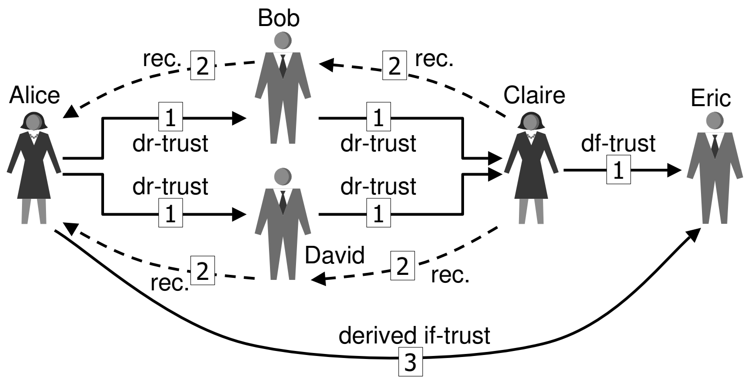


Fig. 5. Trust derived by parallel combination of trust paths

Let us assume again that Alice needs to get her car serviced, and that she asks Bob to recommend a good car mechanic. When Bob replies that Claire, a good friend of his, recommended Eric to him, Alice would like to get a second opinion, so she asks David whether he has heard about Eric. David also knows and trusts Claire, and has heard from her that Eric is a good car mechanic. Alice who does not know Claire personally, is unable to obtain a first hand recommendation about the car mechanic Eric, i.e. she does not directly know anybody with functional trust in Eric. Intuitively, if both Bob and David recommend Claire as a good advisor regarding car mechanics, Alice's trust in Claire's advice will be stronger than if she had only asked Bob. Parallel combination of positive trust thus has the effect of strengthening the derived trust.

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

V. Structured Notation

Transitive trust networks can involve many principals, and in the examples below, capital letters A, B, C, D, E and F will be used to denote principals instead of names such as Alice and Bob.

We will use basic constructs of directed graphs to represent transitive trust networks. We will add some notation elements which allow us to express trust networks in a structured way.

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

The symbol ":" will be used to denote the transitive connection of two consecutive trust arcs to form a transitive trust path. The trust relationships of Fig.4 can be expressed as:

$$([A, E]) = ([A, B] : [B, C] : [C, E]) \tag{1}$$

where the trust scope is implicit. Let the trust scope e.g. be defined as σ: *"trusts X to be a good car mechanic"*. Let the functional variant be denoted by $\mathrm{f}\sigma$ and the referral variant by $\mathrm{r}\sigma$. A distinction can be made between initial *direct trust* and derived *indirect trust*. Whenever relevant, the trust scope can be prefixed with "d" to indicate direct trust ($\mathrm{d}\sigma$), and with "i" to indicate indirect trust ($\mathrm{i}\sigma$). This can be combined with referral and functional trust, so that for example indirect functional trust can be denoted as $\mathrm{if}\sigma$. A reference to the trust scope can then be explicitly included in the trust arc notation as e.g. denoted by $[A, B, \mathrm{dr}\sigma]$. The trust network of Fig.4 can then be explicitly expressed as:

$$([A, E, \mathrm{if}\sigma]) = ([A, B, \mathrm{dr}\sigma] : [B, C, \mathrm{dr}\sigma] : [C, E, \mathrm{df}\sigma]) \tag{2}$$

Let us now turn to the combination of parallel trust paths, as illustrated in Fig.5. We will use the symbol "$\diamond$" to denote the graph connector for this purpose. The "$\diamond$" symbol visually resembles a simple graph of two parallel paths between a pair of agents, so that it is natural to use it in this context. Alice's combination of the two parallel trust paths from her to Eric in Fig.5 can then be expressed as:

$$([A, E, \mathrm{if}\sigma]) = ((([A, B, \mathrm{dr}\sigma] : [B, C, \mathrm{dr}\sigma]) \diamond ([A, D, \mathrm{dr}\sigma] : [D, C, \mathrm{dr}\sigma])) : [C, E, \mathrm{df}\sigma]) \tag{3}$$

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

$$([A, E]) = ((([A, B] : [B, C]) \diamond ([A, D] : [D, C])) : [C, E]) \tag{4}$$

It can be noted that Fig.5 contains two paths. The graph consisting of the two separately expressed paths would be:

$$([A, E]) = ([A, B] : [B, C] : [C, E]) \diamond ([A, D] : [D, C] : [C, E]) \tag{5}$$

A problem with Eq.(5) is that the arc $[C, E]$ appears twice. Although Eq.(4) and Eq.(5) consists of the same two paths, their combined structures are different. Some computational models would be indifferent to Eq.(4) and Eq.(5), whereas others would produce different results depending on which expression is being used. When implementing the serial ":" as binary logic "AND", and the parallel "$\diamond$" as binary logic "OR", the results would be equal. However, when implementing ":" and "$\diamond$" as probabilistic multiplication and comultiplication respectively, the results would be different. It would also be different in the case of applying subjective logic operators for transitivity and parallel combination which will be described in Sec.X below. In general, it is therefore desirable to express graphs in a form where an arc only appears once. This will be called a *canonical expression*.

Definition 3 (Canonical Expression): An expression of a trust graph in structured notation where every arc only appears once is called canonical.

With this structured notation, arbitrarily large trust networks can be explicitly expressed in terms of source, target, and scope, as well as other attributes such as measure and time.

A general directed trust graph is based on directed trust arcs between pairs of nodes. With no restrictions on the possible trust arcs, trust paths from a given source X to a given target Y can contain loops and dependencies, which could result in inconsistent calculative results. Dependencies in the trust graph must therefore be controlled when applying calculative methods to derive measures of trust between two parties. *Normalisation* and *simplification* are two different approaches to dependency control. Our model is based on graph simplification, but for comparison we will briefly examine trust measure normalisation in the next section. Section VII describes our method for Simplifying graphs by removing loops and dependencies from the graph between the source and the target parties, resulting in a directed series-parallel graph which eliminates the need for normalisation.

VI. Normalisation of Trust Measures

The basic principle behind normalisation is to retain the whole trust graph with its loops and dependencies, and normalise the computed trust measures in order to maintain consistency. The PageRank and EigenTrust algorithms, briefly described below, are two examples of how this can be done. Other proposed models including [17], [18], [19] also allow looped and/or dependent transitive trust paths.

A. The PageRank Algorithm

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

PageRank proposed by Page *et al.* (1998) [20] represents a way of ranking the best search results based on a page's reputation. Roughly speaking, PageRank ranks a page according to how many other pages are pointing at it. This can be described as a reputation system, because the collection of hyperlinks to a given page can be seen as public information that can be combined to derive a reputation score. A single hyperlink to a given web page can be seen as a positive rating of that web page. Google's search engine[6] is based on the PageRank algorithm, and the rapidly rising popularity of Google at the cost of Altavista was obviously caused by the superior search results that the PageRank algorithm delivered. The definition of PageRank from Page *et al.* (1998) [20] is given below:

Definition 4 (PageRank): Let P be a set of hyperlinked web pages and let u and v denote web pages in P. Let $N^-(u)$ denote the set of web pages pointing to u and let $N^+(v)$ denote the set of web pages that v points to. Let E be some vector over P corresponding to a source of rank. Then, the PageRank of a web page u is:

$$R(u) = cE(u) + c \sum_{v \in N^-(u)} \frac{R(v)}{|N^+(v)|}\ , \quad \text{where } c \text{ is chosen such that } \sum_{u \in P} R(u) = 1. \tag{6}$$

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

In [20] it is recommended that E be chosen such that $\sum_{u \in P} E(u) = 0.15$. The first term $cE(u)$ in Eq.(6) gives rank value based on initial rank. The second term $c \sum_{v \in N^-(u)} \frac{R(v)}{|N^+(v)|}$ gives rank value as a function of hyperlinks pointing at u.

According to Def.4 $R \in [0,1]$, but the PageRank values that Google provides to the public are scaled to the range [0,10] in increments of 0.25. We will denote the public PageRank of a page u as *PR*(u). This public PageRank measure can be viewed for any web page using Google's toolbar which is a plug-in to the MS Internet Explorer. Although Google do not specify exactly how the public PageRank is computed, it is widely conjectured that it measured on a logarithmic scale with base close to 10. An approximate expression for computing the public PageRank could for example be:

$$PR(u) = l + \log_{10} R(u) \tag{7}$$

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

It is not publicly known how the source rank vector E is defined, but it would be natural to distribute it over the root web pages of all domains weighted by the cost of buying each domain name. Assuming that the only way to improve a page's PageRank is to buy domain names, Clausen (2004) [21] shows that there is a lower bound to the cost of obtaining an arbitrarily good *PR*.

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

In order to provide a semantic interpretation of a PageRank value, a hyperlink can be seen as a positive rating of the page it points to. Negative ratings do not exist in PageRank so that it is impossible to blacklist web pages with the PageRank algorithm of Eq.(6) alone. Before Google with it's PageRank algorithm entered the search engine arena, some webmasters would promote web sites in a spam-like fashion by filling web pages with large amounts of commonly used search key words as invisible text or as metadata in order for the page to have a high probability of being picked up by a search engine no matter what the user searched for. Although this still can occur, PageRank seems to have reduced that problem because a high *PR* is also needed in addition to matching key words in order for a page to be presented to the user.

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

B. The EigenTrust Algorithm

Peer-to-Peer (P2P) networks represent an environment well suited for distributed reputation management. In P2P networks, every node plays the role of both client and server, and is therefore sometimes called a *servent*. This allows the users to overcome their passive role typical of web navigation, and to engage in an active role by providing their own resources. There are two phases in the use of P2P networks. The first is a *search* phase, also called service discovery, which locates servents offering the enquired service. The second phase is the actual service usage, for instance the download of a certain information item.

In some P2P networks, the service discovery relies on information stored in centralised service repositories. One such example is Napster[7] with its resource directory server. In pure P2P networks however, like Gnutella[8] and Freenet[9], also the service discovery phase is completely distributed without single centralised components. There are also hybrid architectures, e.g. the FastTrack architecture which is used in P2P networks like KaZaA[10], grokster[11] and iMesh[12]. In FastTrack based P2P networks, there are nodes

[7]`http://www.napster.com/`
[8]`http://www.gnutella.com`
[9]`http://www.zeropaid.com/freenet`
[10]`http://www.kazaa.com`
[11]`http://www.grokster.com/`
[12]`http://imesh.com`

and supernodes. The latter keep track of other nodes and supernodes that are logged onto the network at any one time, and thus act as directory servers during the search phase. The download phase normally consists of requesting the service from one of the nodes that was discovered during the search phase.

P2P networks introduce a range of security threats, as they can be used to spread malicious software, such as viruses and Trojan horses, and easily bypass firewalls. There is also evidence that P2P networks suffer from free riding [22]. Reputation systems are well suited to mitigate against these problems, e.g. by exchanging and sharing information about rogue, unreliable or selfish participants. P2P networks can be controversial in some applications because they have been used to distribute copyrighted material such as MP3 music files, and it has been claimed that content poisoning[13] has been used by the music industry to sabotage this practice. We do not defend using P2P networks for illegal file sharing, but it is obvious that reputation systems could be used by distributors of illegal copyrighted material to protect P2P networks from poisoning.

Many authors have proposed reputation systems for P2P networks [17], [23], [24], [25], [26], [27], [28]. The purpose of a reputation system in P2P networks is:

1) To determine which servents are most reliable at offering the best quality resources, and
2) To determine which servents provide the most reliable information with regard to (1).

In a distributed environment, each participant is responsible for collecting and combining ratings from other participants. Because of the distributed environment, it is often impossible or too costly to obtain ratings resulting from all interactions with a given agent. Instead the reputation score is based on a subset of ratings, usually from the relying party's "neighbourhood".

The EigenTrust algorithm proposed by Kamvar *et al.* (2003) [17] is aimed at deriving global reputation scores in P2P communities with the purpose of assisting members in choosing the most reputable peers.

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

$$s_{ij} = \text{sat}(i,j) - \text{unsat}(i,j) \tag{8}$$

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

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum\limits_{l \in L} \max(s_{i,l}, 0)} \tag{9}$$

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

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

$$t_{ik} = \sum_{j \in L} c_{ij} c_{jk} \tag{10}$$

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

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

[13]Poisoning music file sharing networks consists of distributing files with legitimate titles - and put inside them silence or random noise.

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

$$\vec{t_i} = C^n \vec{c_i} \tag{11}$$

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

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

VII. Network Simplification

Trust network analysis based on network simplification very different from methods based on normalisation. Simplification of a trust network consists of only including certain arcs in order to allow the trust network between the source trustor and the target trustee to be formally expressed as a canonical expression. Graphs that represent this type of networks are known as *directed series-parallel graphs* (DSPG) [29]. A DSPG can be constructed by sequences of serial and parallel compositions that are defined as follows [29]:

Definition 5 (Directed Series and Parallel Composition):

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

The principle of directed series and parallel composition is illustrated in Fig.6.

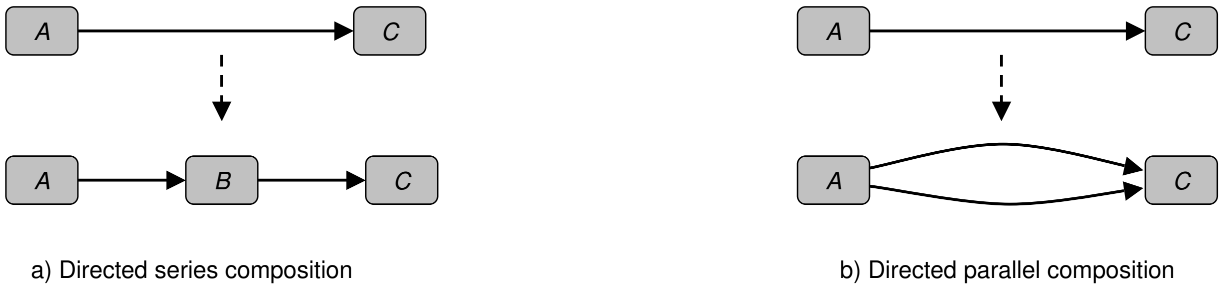

A
C
A
B
C
a) Directed series composition
A
C
A
C
b) Directed parallel composition

Fig. 6. The principles of directed series and parallel composition.

The calculative analysis of a DSPG trust network does not require normalisation, because a DSPG does not have loops and internal dependencies. We will first describe an algorithm for determining all possible paths from a given source to a given target, and secondly discuss how to select a subset of those paths for creating a DSPG. Finally, we will explain why it is necessary to pass a trust recommendation as first hand direct trust from the recommender to the relying party, and not as indirect derived trust.

A different approach to constructing efficient networks, which can be called discovery of small worlds, from a potentially large and complex network, has been described in [30].

A. Finding Paths

Without normalisation, a graph must be simplified in order to remove loops and dependencies. The first step is to determine the possible paths. The pseudo-code in Fig.7 represents an algorithm for finding all directed paths between a given pair of source and target peers, where each individual path is loop free.

Pseudo-Constructor for a trust arc between two parties:

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

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

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

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

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

Fig. 7. Path finding algorithm

Transitive trust graphs can be stored and represented on a computer in the form of a list of directed trust arcs with additional attributes. Based on the list of arcs, an automated parser can establish valid DSPGs between two parties depending on the need. The initial direct trust arcs of Fig.8 can for example be listed as in Table I.

A parser based on the algorithm of Fig.7 going through Table I will be able to determine the directed graph of Fig.8 between A and E. The principal A can be called a relying party because she relies on the recommendations from B, D and C to derive her trust in E.

TABLE I

INITIAL DIRECT TRUST RELATIONSHIPS OF FIG.13

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

B. *Discovering the Optimal Directed Series-Parallel Graph*

Ideally, all the possible paths discovered by the algorithm of Fig.7 should be taken into account when deriving the trust value. A general directed graph will often contain loops and dependencies. This can be avoided by excluding certain paths, but this can also cause information loss. Specific selection criteria are needed in order to find the optimal subset of paths to include. With n possible paths, there are $2^n - 1$ different combinations for constructing graphs, of which not all necessarily are DSPGs. Of the graphs that are DSPGs, only one will be selected for deriving the trust measure.

Fig.8 illustrates an example of a non-DSPG with dependent paths, where it is assumed that A is the source and E is the target. While there can be a large number of possible distinct paths, it is possible to use heuristic rules to discard paths, e.g. when their confidence drop below a certain threshold.

In the pseudocode of Fig.7, the line "(Confidence(path + arc) $>$ Threshold)" represents such a heuristic rule for simplifying the graph analysis. By removing paths with low confidence, the number of paths to consider is reduced while the information loss can be kept to an insignificant level.

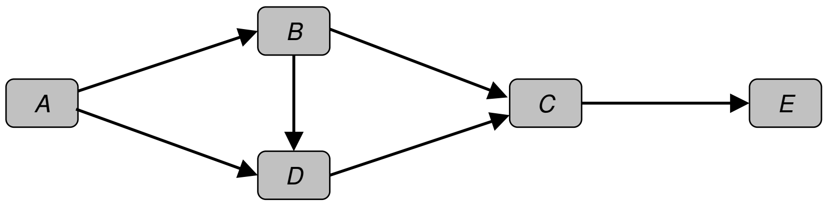


Fig. 8. Dependent paths

In the graph of Fig.8 there are 3 possible paths between A and E:

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

This leads to the following 7 potential combinations/graphs.

$$\begin{array}{lll}
\gamma_1 = \phi_1 & \gamma_4 = \phi_1 \diamond \phi_2 & \gamma_7 = \phi_1 \diamond \phi_2 \diamond \phi_3 \\
\gamma_2 = \phi_2 & \gamma_5 = \phi_1 \diamond \phi_3 & \\
\gamma_3 = \phi_3 & \gamma_6 = \phi_2 \diamond \phi_3 &
\end{array} \qquad (12)$$

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

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

- **Exhaustive Discovery of Optimal Trust Graphs**
  The exhaustive method of finding the optimal DSPG consists of determining all possible DSPGs, then deriving the trust value for each one of them, and finally selecting the DSPG and the corresponding canonical expression that produces the trust value with the highest confidence level. The computational complexity of this method is $\mathrm{Comp} = lm(2^n - 1)$, where n is the number of possible paths, m is the average number of paths in the DSPGs, and l is the average number of arcs in the paths.
- **Heuristic Discovery of Near-Optimal Trust Graphs**
  The heuristic method of finding a near-optimal DSPG consists of constructing the graph by including new paths one by one in decreasing order of confidence. Each new path that would turn the graph into a non-DSPG and break canonicity is excluded. This method only requires the computation of the trust value for a single DSPG and canonical expression, with computational complexity $\mathrm{Comp} = lm$, where m is average number of paths in the DSPGs, and l is the average number of arcs in the paths.

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

Figures 9, 10 and 11 illustrate how new paths can be included in a way that preserves graph canonicity. In the figures, the nesting level of nodes and arcs is indicated as an integer. A bifurcation is when a node has two or more incoming or outgoing arcs, and is indicated by brackets in the shaded node boxes. The opening bracket "(" increments the nesting level by 1, and the closing bracket ")" decrements the nesting level by 1. A sub-path is a section of a path without bifurcations. The equal sign "=" means that the node is part of a sub-path, in which case the nesting level of the arc on the side of the $=$ sign is equal to the nesting level of the node. Each time a new path is added to the old graph, some sub-path sections may already exist in the old graph which does not require any additions, whereas other sub-path sections that do not already exist, must be added by bifurcations to the old graph. The following criteria are needed for preserving a DSPG when adding new sub-paths. The source and target nodes refer to the source and target nodes of the new sub-path that is to be added to the old graph by bifurcation.

Definition 6 (Criteria for Preserving a DSPG when Adding New Sub-Paths):

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

These principles are illustrated with examples below. Requirement 1) is illustrated in Fig.9. The new arc $[B, D]$ is not allowed because D is not reachable from B in the old graph, whereas the new arc $[A, C]$ is allowed because C is reachable from A.

The new allowed arc can be included under the same nesting level as the sub-paths $([A, B] : [B : C])$ and $([A, D] : [D : C])$ in this example. The old and new graph of Fig.9 are expressed below. Note that the brackets around sub-paths, e.g. $([A, B] : [B, C])$, are not reflected in Fig.9 because they do not represent

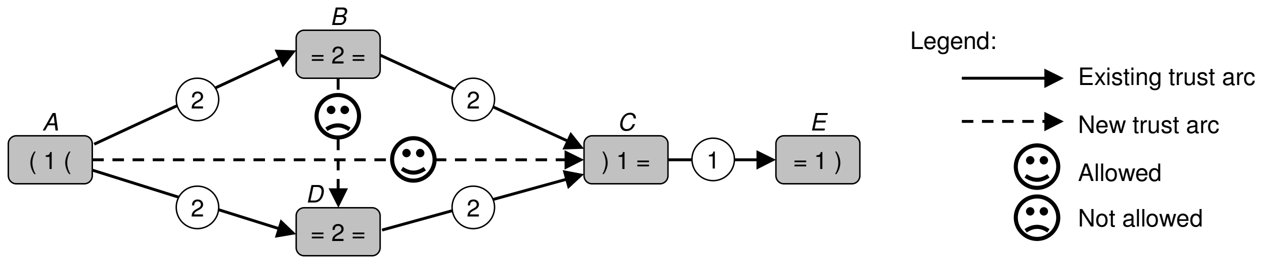


Fig. 9. Visualising the requirement that the target node must be reachable from the source node.

nesting, but simply grouping of arcs belonging to the same sub-path.

$$\begin{aligned}&\text{Old graph:} \quad ((([A,B]:[B,C]) \diamond ([A,D]:[D,C])):[C,E]) \\ &\text{New graph:} \quad ((([A,B]:[B,C]) \diamond ([A,D]:[D,C]) \diamond [A,C]):[C,E])\end{aligned} \tag{13}$$

Requirement 2) is illustrated in Fig.10. The new arc $[B,D]$ is not allowed because B and D have different nesting levels, whereas the new arc $[A,D]$ is allowed because A and D have equal nesting levels. Node A does in fact have nesting levels 1 and 2 simultaneously because two separate bifurcations with different nesting levels start from A.

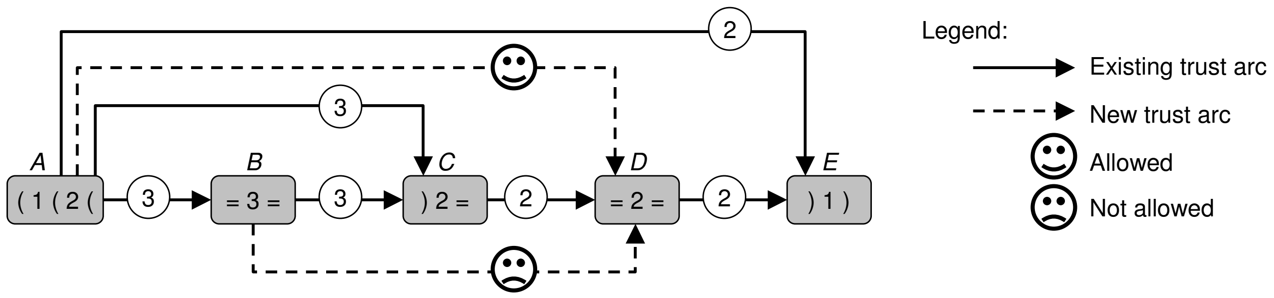


Fig. 10. Visualising the requirement that the source and target nodes must have equal nesting levels.

Adding the new allowed arc results in an additional nesting level being created, which also causes the nesting levels of the sub-paths $[A,B]:[B,C]$ and $[A,C]$ to increment. The old and new graph of Fig.10 can then be expressed as:

$$\begin{aligned}&\text{Old graph:} \quad (((([A,B]:[B,C]) \diamond [A,C]):[C,D]:[D,E]) \diamond [A,E]) \\ &\text{New graph:} \quad ((((([A,B]:[B,C]) \diamond [A,C]):[C,D]) \diamond [D,E]) \diamond [A,E])\end{aligned} \tag{14}$$

Requirement 3) is illustrated in Fig.11. The new arc $[B,D]$ is not allowed because the node C has a nesting level that is less that the nesting level of B and D, whereas the new arc $[A,E]$ is allowed because the nesting level of C is equal to that of A and E.

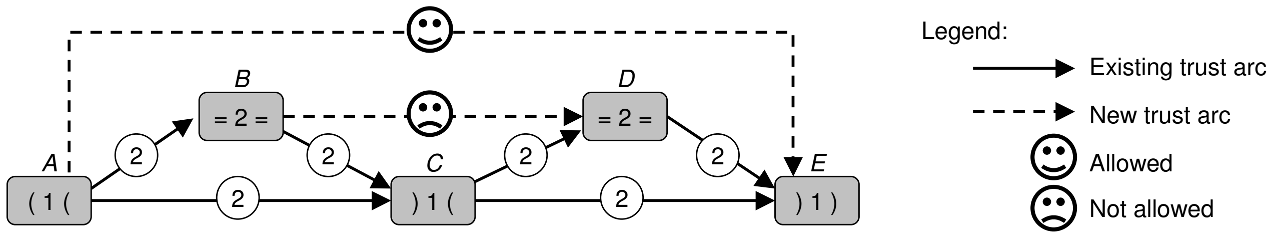


Fig. 11. Visualising the requirement that intermediate nodes can not have a nesting level less than the source and target nodes

Adding the new allowed arc results in an additional nesting level being created, which also causes the nesting levels of the existing sub-paths to increment. The old and new graph of Fig.11 can then be expressed as:

$$\text{Old graph:} \quad ((([A,B]:[B,C]) \diamond [A,C]) : (([C,D]:[D,E]) \diamond [C,E]))$$
$$\text{New graph:} \quad (((([A,B]:[B,C]) \diamond [A,C]) : (([C,D]:[D,E]) \diamond [C,E])) \diamond [A,E]) \tag{15}$$

C. First Hand Trust

Even if relying parties are aware of the need to base computations on canonical expressions, it is possible that the received recommendations prevent them from following that principle. Fig.12 shows an example of how a certain type of recommendation can lead to non-canonical hidden trust expressions.

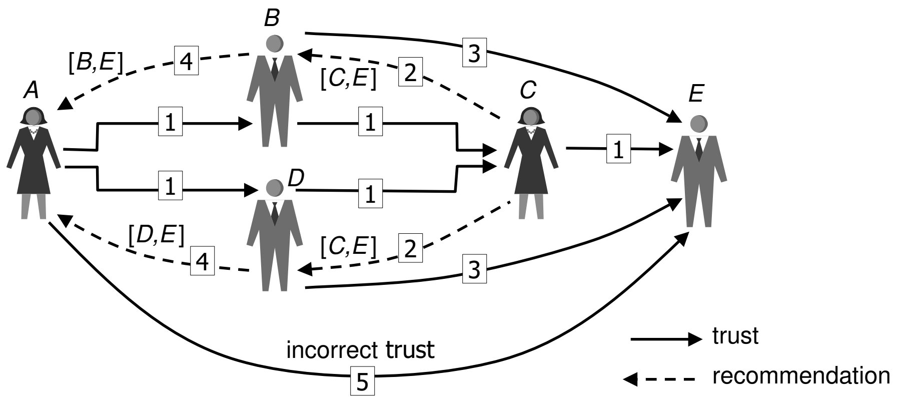


Fig. 12. Incorrect recommendation

Here the trust and recommendation arrows are indexed according to the order in which they are formed. In the scenario of Fig.12, C passes her recommendation about E to B and D (index 2), so that B and D are able to derive indirect trust in E (index 3). Now B and D pass their derived indirect trust in E to A (index 4), so that she can derive indirect trust in E (index 5). The problem with this scenario is that A is ignorant about C so that A in fact analyses a hidden graph with a non-canonical expression that is different from expression of the perceived graph:

$$\begin{array}{ccc} \text{Perceived graph:} & & \text{Hidden graph:} \\ (([A,B]:[B,E]) \diamond ([A,D]:[D,E])) & \neq & (([A,B]:[B,C]:[C,E]) \diamond ([A,D]:[D,C]:[C,E])) \end{array} \tag{16}$$

The reason for this is that when B and D recommend $[B,E]$ and $[D,E]$, they implicitly recommend $([B,C]:[C,E])$ and $([D,C]:[C,E])$, but A is not aware of this [31]. It can easily be seen that neither the perceived nor the hidden graph is equal to the real graph, which shows that this way of passing recommendations can lead to incorrect analysis.

We argue that B and D should pass the recommendations explicitly as $([B,C]:[C,E])$ and $([D,C]:[C,E])$ respectively so that the relying party A knows their origin. The recommenders B and D are normally are able to do this, but then A also needs to be convinced that B and D have not altered the recommendation from C. If B and D are unreliable, they might for example try to change the recommended trust measures from C. Not only that, any party that is able to intercept the recommendations from B, D, or C to A might want to alter the trust values or any other parameters, and A therefore needs to receive evidence of the authenticity and integrity of the recommendations. Cryptographic security mechanisms can typically be used to solve this problem, and this will be discussed in more detail in Sec.VIII. An example of a correct way of passing recommendations is indicated in Fig.13

In the scenario of Fig.13, A receives all the recommendations directly, resulting in a perceived graph that is equal to the real graph, that can be expressed as:

$$([A,E]) = ((([A,B]:[B,C]) \diamond ([A,D]:[D,C])) : [C,E]) \tag{17}$$

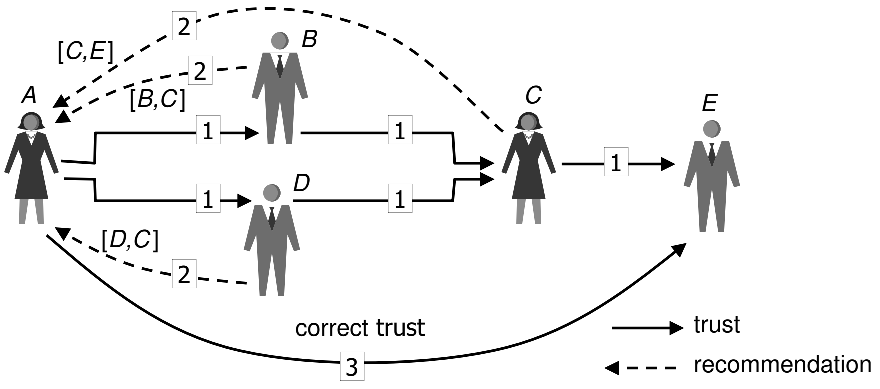


Fig. 13. Correct recommendations

The lesson to be learned from the scenarios in Fig.12 and Fig.13 is that there is a crucial difference between recommending trust in a principal, resulting from your own experience with that principal, and recommending trust in a principal which has been derived as a result of recommendations from others. As already mentioned, the term *direct trust* represents the former, and *indirect trust* the latter. Fig.12 illustrated how problems can occur when indirect trust is recommended, so the rule is to only recommend direct trust [31]. For example, A's derived indirect trust in E in Fig.13 should not be recommended to others.

VIII. Integrity and Authenticity of Recommendations

Cryptography can be used to provide authenticity and integrity of trust recommendations. This in turn requires that every participant holds a trusted (i.e. authentic) key. The process of generating, distributing and using cryptographic keys is called key management, and this still is a major and largely unsolved problem on the Internet today.

Public-key infrastructures (PKI) can simplify key management and distribution, but have very strict trust requirements. A PKI refers to an infrastructure for distributing public keys where the authenticity of public keys is certified by Certification Authorities (CA). A certificate basically consists of the CA's digital signature on the concatenation of the public key and the owner identifier, thereby linking the key and the owner identifier together in an unambiguous way. In order to verify a certificate, the CA's public key is needed, thereby creating an identical authentication problem. The CA's public key can be certified by another CA etc., but in the end one must receive the public key of some CA out-of-band in a secure way. Although out-of-band channels can be expensive to set up and operate, they are absolutely essential in order to obtain a complete chain of trust from the relying party to the target public key.

However, there are potential trust problems in this design. What happens if a CA issues a certificate but does not properly check the identity of the owner, or worse, what happens if a CA deliberately issues a certificate to someone with a false owner identity? Furthermore, what happens if a private key with a corresponding public-key certificate is leaked to the public domain by accident, or worse, by intent? Such events could lead to systems and users making totally wrong assumptions about digital identities. Clearly CAs must be trusted to be honest and to do their job properly, and users must be trusted to protect their private keys.

When including security in the description of our scheme, it must be assumed that every principal has a public/private key pair that can be used for authentication and encryption. We can either assume that the public keys are absolutely trusted (i.e. that the relying party is absolutely certain about their authenticity) or that they too can have various levels of trustworthiness. The easiest, of course, is to assume absolute trust, because then the authenticity and integrity of the trust recommendations can be assumed, and trust networks can be analysed as described in the previous sections.

If on the other hand trust in cryptographic keys can have varying measures, then the trust in every cryptographic key must be determined before the primary trust network of interest can be analysed. Trust in public keys can be derived from trust in the various components of a PKI. A method for analysing trust in the authenticity of public keys in a PKI is described in detail in [31]. We build on that model by explicitly separating between the functional and referral variants of trust scopes, as illustrated in Fig.14.

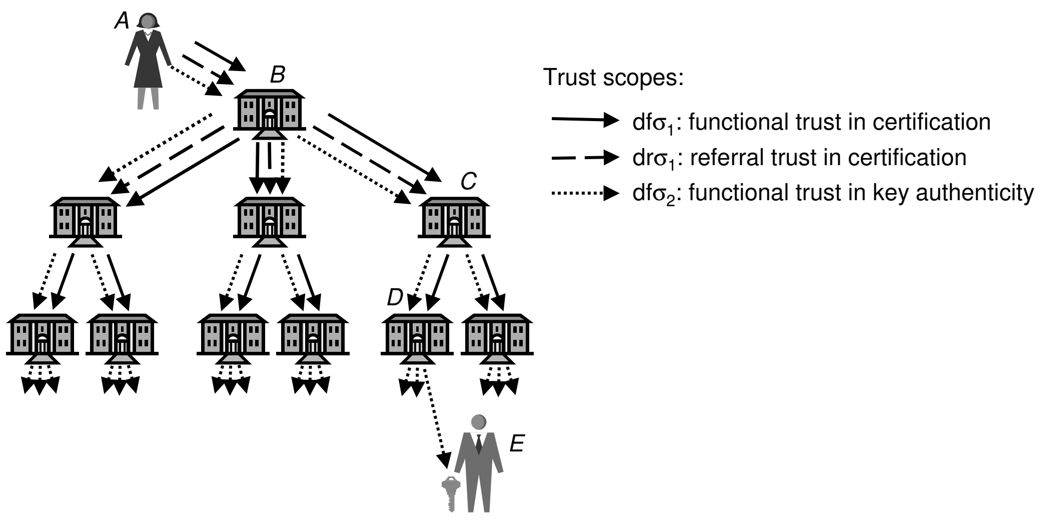

A
B
C
D
E
Trust scopes:
dfσ1: functional trust in certification
drσ1: referral trust in certification
dfσ2: functional trust in key authenticity

Fig. 14. Structure of trust scopes in PKI

Two different trust scopes are used:

- σ_1: *"Trusts the CA to correctly certify public keys"*
- σ_2: *"Trusts the public key to be authentic"*.

In Fig.14 it is assumed that the relying party A has obtained the public key of the root CA B through a secure out-of-band channel. Depending on the security of that out-of-band channel, A has a level of direct functional trust in the authenticity of B's public key, which can be denoted by $[A, B, \mathrm{df}\sigma_2]$. In addition, A has direct functional trust in B to correctly certify the public keys of subordinate CAs, which can be denoted by $[A, B, \mathrm{df}\sigma_1]$. Finally A has direct referral trust in B with the same scope, meaning that A trusts B to ascertain that subordinate CAs are able to correctly certify public keys, which can be denoted by $[A, B, \mathrm{dr}\sigma_1]$.

B needs to trust C with exactly the same scopes as A's trust in B. As there are no subordinate CAs under D, the referral variant of C's σ_1 trust in D is not needed. D's trust in the user E is the simplest, because it only focuses on the authenticity of E's public-key, which can be denoted by $[D, E, \mathrm{df}\sigma_2]$.

The relying party A is interested in deriving a measure of authenticity of user E's public key through the trust web of this PKI. With the specified trust scopes and trust relationships, this can be expressed as:

$$\begin{aligned}([A,E,\mathrm{if}\sigma_2]) = \quad & (([A,B,\mathrm{df}\sigma_2] \wedge [A,B,\mathrm{df}\sigma_1] \wedge [A,B,\mathrm{dr}\sigma_1]) \\ & : ([B,C,\mathrm{df}\sigma_2] \wedge [B,C,\mathrm{df}\sigma_1] \wedge [B,C,\mathrm{dr}\sigma_1]) \\ & : ([C,D,\mathrm{df}\sigma_2] \wedge [C,D,\mathrm{df}\sigma_1]) \\ & : [D,E,\mathrm{df}\sigma_2])\end{aligned} \tag{18}$$

The existence of up to three separate trust arcs with different scopes between parties requires some way for combining them together. Various methods can be imagined for this purpose, and one possibility is to use conjunction (i.e. logical AND in the binary case) of the two trust scopes [31]. The connector "$\wedge$" is used in Eq.(18) to denote that a conjunction of the trust scopes is needed, e.g. meaning that A must trust B to have an authentic key, AND to provide reliable certification, AND to verify that subordinate CAs also provide reliable certification.

The consequence of having to derive trust in public keys is that the relying party might have to analyse a separate auxiliary trust network for every principal in the trust network of interest. Deriving indirect trust in a remote party would then have to take the authenticity of the public keys into account in addition to the trust in the principal. We will illustrate this with a very simple example, such as for delivering an online service, where A received a trust recommendation from E about F with a particular scope. The trust relationships that must be taken into account are illustrated in Fig.15 below. The parties A and E are the same as those in Fig.14, and A is now using the trust she derived from the PKI. Two different trust scopes are used:

- σ_2: *"Trusts the public key to be authentic"*.
- σ_3: *"Trusts the agent to provide quality service"*

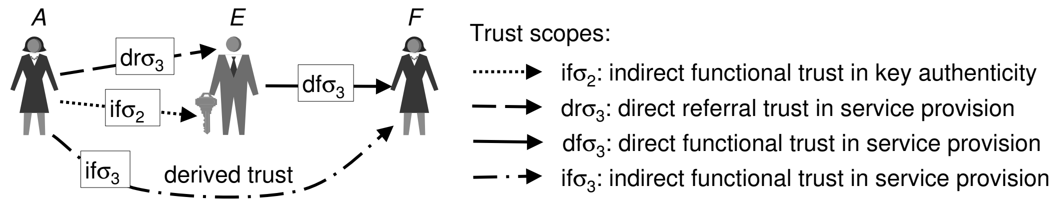

A
E
F
drσ3
ifσ2
dfσ3
ifσ3
derived trust
Trust scopes:
ifσ2: indirect functional trust in key authenticity
drσ3: direct referral trust in service provision
dfσ3: direct functional trust in service provision
ifσ3: indirect functional trust in service provision

Fig. 15. Trust transitivity with authenticated public keys

We will again use the symbol "$\wedge$" to denote conjunction of two trust arcs with different scopes between the same pair of entities. A's derived trust in service provider F can then be expressed as:

$$([A, F, \mathrm{if}\sigma_3]) = (([A, E, \mathrm{if}\sigma_2] \wedge [A, E, \mathrm{dr}\sigma_3]) : [E, F, \mathrm{df}\sigma_3]) \tag{19}$$

For a parser to be able to derive this trust network, it is required that the relying party A has received and stored all these trust recommendations, for example in the form of a table similar to Table I. Only the first trust scope in a conjunctive trust relationship is used by the parser to determine the actual trust network. The second trust scope is only used when deriving the trust measure.

To illustrate the role of key authenticity, take for example the case when a principal is recommended to be reliable, but that the binding between the principal and his key is broken, e.g., because it is known that the private key has been stolen by an intruder. The conjunction between trust in the principal and distrust in his key would result in low trust, indicating that a principal identified by this particular public key can not be strongly trusted despite an otherwise positive trust recommendation. This is what intuition would dictate because it is now possible that recommendations that appear to come from the principal in fact originate from the intruder who stole the private key and who can not be trusted.

IX. Measuring and Computing Trust

In previous sections we have indicated some intuitive principles that trust measures and computational rules should follow. This section describes additional requirements that trust measures and operators should satisfy. Sec.X below describes a practical example of how measures of trust can be mathematically expressed and derived.

Many trust measures have been proposed in the literature [10], [14] varying from discrete measures [11], [32], [33], [34], [35], to continuous measures [31], [36], [37], [38], [39], [40].

Typical discrete trust measures are for example "strong trust", "weak trust", strong distrust" and "weak distrust". PGP[11] is a well known software tool for cryptographic key management and email security that for example uses the discrete trust measures "ultimate", "always trusted", "usually trusted", "usually not trusted" and "undefined" for key owner trustworthiness. In order to obtain compatibility between discrete and continuous methods, it is possible to interpret such discrete verbal statements by mapping them to continuous measures [41].

The type of trust described in industry specification efforts such as WS-Trust [5] and the Liberty Alliance [6], [7] must be seen as binary, i.e. an agent or a claim is either trusted or not trusted. Parallel trust paths and conflicting trust recommendations are not considered in these frameworks.

When measuring trust, it is critical that the trust value is *meaningful* to, and *usable* for both the source and the target transacting partners. Otherwise, if trust is subjectively measured by each party using different methods and scales, the values become meaningless when used in a calculative manner.

By explicitly defining the trust scopes σ_1 and σ_2 in the scenarios above, the interacting parties are able to establish a common understanding of the trust scope, and thereby determine semantically compatible measures of trust.

The trust scope in the network to be analysed must be consistent along every valid path. Again, by explicitly defining σ_1, σ_2 and σ_3, the scopes become clear to all parties involved in the collection and analysis of trust data.

As mentioned in Sec. II, the *time* element should be captured together with trust measures. This element is necessary not only to demonstrate how trust is evolving, but also in order to enable transaction partners to assess trust based on, for example, the most recent trust values available.

Determining the *confidence* of the trust measure is also a requirement. For example, the weakening of trust through long transitive paths should result in a reduced confidence level. On the other hand, combination of parallel trust paths should result in an increased confidence level.

Finally, in order to derive trust measures from graphs, calculative methods are needed for *serial combination of trust measures* along transitive paths illustrated in Fig.4, for *fusion of trust measures* from parallel paths illustrated in Fig.5, as well as for *conjunctive combination of trust measures* illustrated in Fig.15. The validation and suitability assessment of any computational approach should be based on simulations and usability studies, in environments equal or similar to those where it is intended for deployment.

X. Trust Network Analysis with Subjective Logic

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

A. Subjective Logic Fundamentals

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

Subjective logic[36] represents a specific belief calculus that uses a belief metric called *opinion* to express beliefs. An opinion denoted by $\omega_x^A = (b, d, u, a)$ expresses the relying party A's belief in the truth of statement x. When a statement for example says *"Party X is honest and reliable regarding σ"*, then the opinion about the truth of that statement can be interpreted as trust in X within the scope of σ. Here b, d, and u represent belief, disbelief and uncertainty respectively where $b, d, u \in [0, 1]$ and $b + d + u = 1$. The parameter $a \in [0, 1]$ is called the base rate, and is used for computing an opinion's probability expectation value that can be determined as $\mathrm{E}(\omega_x^A) = b + au$. More precisely, a determines how uncertainty shall contribute to the probability expectation value $\mathrm{E}(\omega_x^A)$. In the absence of any specific evidence about a given party, the base rate determines the *a priori* trust that would be put in any member of the community.

The opinion space can be mapped into the interior of an equal-sided triangle, where, for an opinion $\omega_x = (b_x, d_x, u_x, a_x)$, the three parameters b_x, d_x and u_x determine the position of the point in the triangle representing the opinion. Fig.16 illustrates an example where the opinion about a proposition x from a binary state space has the value $\omega_x = (0.7,\ 0.1,\ 0.2,\ 0.5)$.

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

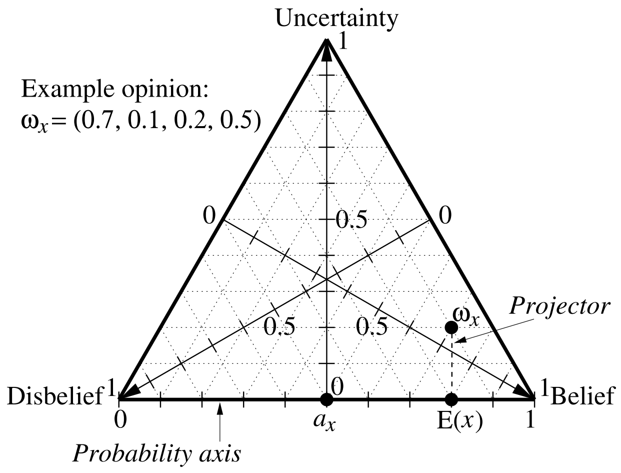


Fig. 16. Opinion triangle with example opinion

The base of the triangle is called the probability axis. The base rate is indicated by a point on the probability axis, and the projector starting from the opinion point is parallel to the line that joins the uncertainty vertex and the base rate point on the probability axis. The point at which the projector meets the probability axis determines the expectation value of the opinion, *i.e.* it coincides with the point corresponding to expectation value $\mathrm{E}(\omega_x^A)$.

Opinions can be ordered according to probability expectation value, but additional criteria are needed in case of equal probability expectation values. We will use the following rules to determine the order of opinions[36]:

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

1) The opinion with the greatest probability expectation is the greatest opinion.
2) The opinion with the least uncertainty is the greatest opinion.
3) The opinion with the least base rate is the greatest opinion.

The probability density over binary event spaces can be expressed as beta PDFs (probability density functions) denoted by beta(α, β) [42]. Let r and s express the number of positive and negative past observations respectively, and let a express the *a priori* or base rate, then α and β can be determined as:

$$\alpha = r + 2a \ , \qquad \beta = s + 2(1 - a) \ . \tag{20}$$

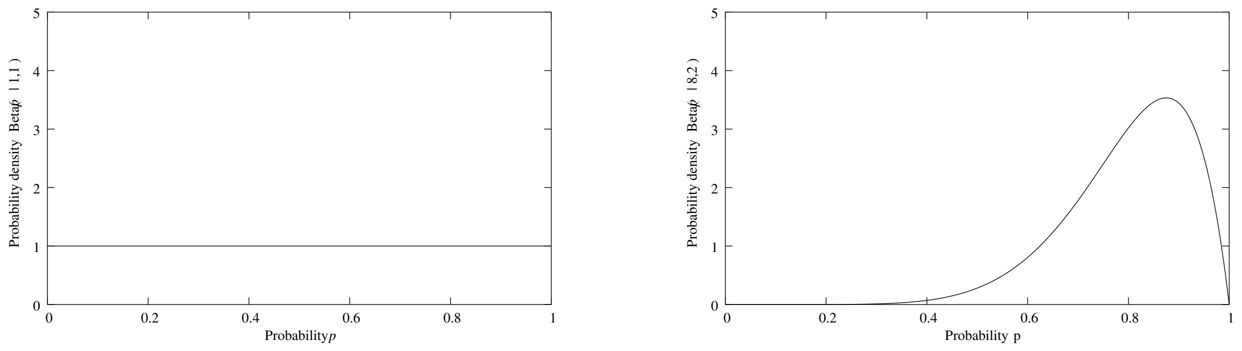
A bijective mapping between the opinion parameters and the beta PDF parameters can be analytically derived [36], [43] as:

$$\left\{ \begin{array}{l} b_x = r/(r+s+2) \\ d_x = s/(r+s+2) \\ u_x = 2/(r+s+2) \\ a_x = \text{base rate of } x \end{array} \right. \Longleftrightarrow \left\{ \begin{array}{ll} r & = 2b_x/u_x \\ s & = 2d_x/u_x \\ 1 & = b_x + d_x + u_x \\ a & = \text{base rate of } x \end{array} \right. \tag{21}$$

This means for example that a totally ignorant opinion with $u_x = 1$ and $a_x = 0.5$ is equivalent to the uniform PDF beta$(1,1)$ illustrated in Fig.17.a. It also means that a dogmatic opinion with $u_x = 0$ is equivalent to a spike PDF with infinitesimal width and infinite height expressed by beta$(b_x\eta,\ d_x\eta)$, where $\eta \to \infty$. Dogmatic opinions can thus be interpreted as being based on an infinite amount of evidence.

After r positive and s negative observations in case of a binary state space (i.e. $a = 0.5$), the *a posteriori* distribution is the beta PDF with $\alpha = r + 1$ and $\beta = s + 1$. For example the beta PDF after observing 7 positive and 1 negative outcomes is illustrated in Fig.17.b, which also is equivalent to the opinion illustrated in Fig.16

A PDF of this type expresses the uncertain probability that a process will produce positive outcome during future observations. The probability expectation value of Fig.17.b. is $\mathrm{E}(p) = 0.8$. This can be



(a) Uniform beta(1,1)

(b) beta(8,2)

Fig. 17. Example beta probability density functions

interpreted as saying that the relative frequency of a positive outcome in the future is somewhat uncertain, and that the most likely value is 0.8.

The variable p is a probability variable, so that for a given p the probability density beta$(p \,|\, \alpha, \beta)$ represents second order probability. The first-order variable p represents the probability of an event, whereas the density beta$(p \,|\, \alpha, \beta)$ represents the probability that the first-order variable has a specific value. Since the first-order variable p is continuous, the second-order probability beta$(p \,|\, \alpha, \beta)$ for any given value of $p \in [0, 1]$ is vanishingly small and therefore meaningless as such. It is only meaningful to compute $\int_{p_1}^{p_2}$ beta$(p \,|\, \alpha, \beta)$ for a given interval $[p_1, p_2]$, or simply to compute the expectation value of p. The expectation value of the PDF is always equal to the expectation value of the corresponding opinion. This provides a sound mathematical basis for combining opinions using Bayesian updating of beta PDFs.

B. Reasoning with Beliefs

Subjective logic defines a number of operators[36], [41], [44]. Some operators represent generalisations of binary logic and probability calculus operators, whereas others are unique to belief theory because they depend on belief ownership. Here we will only focus on the *discounting* and the *consensus* operators. The discounting operator can be used to derive trust from transitive paths, and the consensus operator can be used to derive trust from parallel paths. These operators are described below.

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

$$\omega_x^{A:B} = \omega_B^A \otimes \omega_x^B \qquad \begin{cases} b_x^{A:B} = b_B^A b_x^B \\ d_x^{A:B} = b_B^A d_x^B \\ u_x^{A:B} = d_B^A + u_B^A + b_B^A u_x^B \\ a_x^{A:B} = a_x^B \;. \end{cases} \tag{22}$$

  The effect of discounting in a transitive chain is that uncertainty increases, not disbelief [45].
- **Consensus** is equivalent to Bayesian updating in statistics. The consensus of two possibly conflicting opinions is an opinion that reflects both opinions in a fair and equal way. Let ω_x^A and ω_x^B be A's and B's opinions about the same proposition x. The opinion $\omega_x^{A\diamond B}$ is then called the consensus between ω_x^A and ω_x^B, denoting an imaginary agent $[A, B]$'s opinion about x, as if she represented both A and

B. By using the symbol '$\oplus$' to designate this operator, we define $\omega_x^{A\diamond B} = \omega_x^A \oplus \omega_x^B$.

$$\omega_x^{A\diamond B} = \omega_x^A \oplus \omega_x^B \qquad \begin{cases} b_x^{A\diamond B} = (b_x^A u_x^B + b_x^B u_x^A)/(u_x^A + u_x^B - u_x^A u_x^B) \\ d_x^{A\diamond B} = (d_x^A u_x^B + d_x^B u_x^A)/(u_x^A + u_x^B - u_x^A u_x^B) \\ u_x^{A\diamond B} = (u_x^A u_x^B)/(u_x^A + u_x^B - u_x^A u_x^B) \\ a_x^{A\diamond B} = a_x^A \end{cases} \tag{23}$$

where it is assumed that $a_x^A = a_x^B$. Limits can be computed [46] for $u_x^A = u_x^B = 0$. The effect of the consensus operator is to amplify belief and disbelief and reduce uncertainty.

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

C. Example Derivation of Trust Measures

This numerical example is based the trust graph of Fig.13. Table II specifies trust measures expressed as opinions. The DSTC Subjective Logic API[14] was used to compute the derived trust values.

TABLE II

DIRECT TRUST MEASURES OF FIG.13

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

By applying the discounting and consensus operators to the expression of Eq.(17), the derived indirect trust measure can be computed.

- Case a:

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

$$\begin{aligned} \omega_E^A &= ((\omega_B^A \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.74,\ 0.00,\ 0.26,\ 0.50) \end{aligned} \tag{24}$$

- Case b:

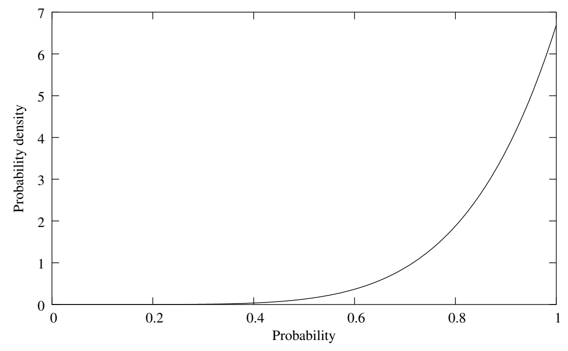
  Let us now assume that based on new experience on 19.04.2006, A's trust in B suddenly is reduced to that of the last entry for $[A, B]$ in Table II. As a result of this, A needs to update her derived trust in E and computes:

$$\begin{aligned} \omega_E^{'A} &= ((\omega_B^{'A} \otimes \omega_C^B) \oplus (\omega_D^A \otimes \omega_C^D)) \otimes \omega_E^C \\ &= (0.287,\ 0.000,\ 0.713,\ 0.500) \end{aligned} \tag{25}$$

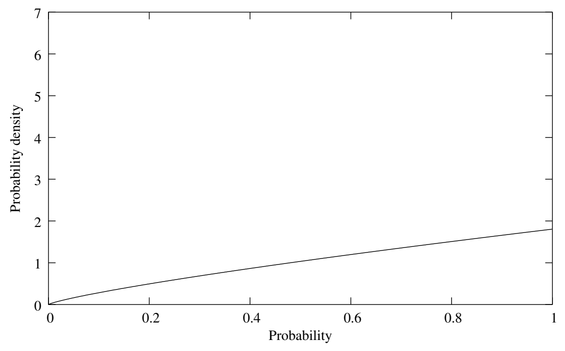
The derived trust measures can be translated into beta PDFs according to Eq.(21) and visualised as density functions as illustrated by Fig.18 below.

It can be seen that the trust illustrated in Fig.18.a is relatively strong but that the trust in Fig.18.b approaches the uniform distribution of Fig.17.a and therefore is very uncertain. The interpretation of this is that the distrust introduced in the arc $[A, B]$ in case (b) has rendered the path $([A, B] : [B, C] : [C, E])$ useless. In other words, when A distrusts B, then whatever B recommends is completely discounted by A. It is as if B had not recommended anything at all. As a result A's derived trust in E must be based on the path $([A, D] : [D, C] : [C, E])$ which was already weak from the start.

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



(a) $\omega_E^A \equiv \text{beta}(6.7,\ 1.0)$



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

Fig. 18. Derived trust visualised as beta probability density functions

XI. Discussion and Conclusion

We have presented a notation for expressing trust networks, and a method for trust network analysis based on graph simplification and trust derivation with subjective logic. This approach is called Trust Network Analysis with Subjective Logic (TNA-SL).

Our approach is very different from trust network analysis based on normalisation, as e.g. in PageRank and EigenTrust which were briefly explained above. The main advantage of normalisation is that it can be applied to large highly connected random graphs without loosing any trust information. The main disadvantages of normalisation is that it makes trust measures relative, which prevents them from being interpreted in any absolute sense like e.g. statistical reliability. It is also very difficult to express and analyse negative trust in models based on normalisation. Neither PageRank nor EigenTrust can handle negative trust.

Trust network simplification results in the network being expressed as a directed series-parallel graph. A trust arc has the three basic attributes of source, target and scope, where the trust scope can take either the functional or the referral variant. This makes it possible to express and analyse fine-grained semantics of trust, which is not possible with e.g. PageRank or EigenTrust. Additionally, we have incorporated the attributes of measure and time into the model in order to make it suitable for deriving indirect trust measures through computational methods.

One advantage of TNA-SL is that negative trust can be explicitly expressed and propagated. In order for distrust to be propagated in a transitive fashion, all intermediate referral arcs must express positive trust, with only the last functional arc expressing negative trust. Another advantage is that trust measures in our model are equivalent to beta PDFs, so that trust measures can be directly interpreted in statistical terms, e.g. as measures of reliability. This also makes it possible to consistently derive trust measures from statistical data. Our model is for example directly compatible with Bayesian reputation systems [47], [48], so that reputation scores can be directly imported as trust measures. This rich way of expressing trust separates between the nominal trust value (positive/negative) and the confidence level (high/low), and also carries information about the baseline trust in the community.

The main disadvantage of TNA-SL is that a complex and cyclic network must be simplified before it can be analysed, which can lead to loss of information. While the simplification of large highly connected networks could be slow, heuristic techniques can significantly reduce the computational effort. This is done by ignoring paths for which the confidence level drops below a certain threshold, and by including the paths with the strongest confidence level first when constructing a simplified network. This also leads to minimal loss of information.

The approach to analysing transitive trust networks described here provides a practical method for expressing and deriving trust between peers/entities within a community or network. It can be used in a wide range of applications, such as monitoring the behaviour of peers and assisting decision making in P2P communities, providing a quantitative measure of quality of web services, assessing the reliability of peers in Internet communities, and evaluating the assurance of PKI certificates. We also described how integrity and authenticity of trust recommendations can be protected by using an overlaying authentication infrastructure such as a PKI. Combined with subjective logic, TNA-SL allows trust measures to be efficiently analysed and computed, and ultimately interpreted by humans and software agents.

REFERENCES

[1] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of the 1996 IEEE Conference on Security and Privacy*, Oakland, CA, 1996.

[2] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, *RFC 2704 - The KeyNote Trust Management System Version 2*. Network Working Group, IETF, September 1999, url: http://www.ietf.org/rfc/rfc2704.txt.

[3] C. Ellison *et al.*, *RFC 2693 - SPKI Certification Theory*. IETF, September 1999, url: http://www.ietf.org/rfc/rfc2693.txt.

[4] R. L. Rivest and B. Lampson, "SDSI – A simple distributed security infrastructure," Presented at CRYPTO'96 Rumpsession, 1996. [Online]. Available: citeseer.ist.psu.edu/article/rivest96sdsi.html

[5] WS-Trust, *Web Services Trust Language (WS-Trust)*. ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, February 2005.

[6] Liberty-Alliance, *Liberty ID-FF Architecture Overview.* Version: 1.2-errata-v1.0. http://www.projectliberty.org/specs/liberty-idff-arch-overview-v1.2.pdf, 2003.

[7] ——, *Liberty Trust Models Guidelines*, Draft Version 1.0-15 ed., J. Linn, Ed. http://www.projectliberty.org/specs/liberty-trust-models-guidelines-v1.0.pdf, 2003.

[8] D. McKnight and N. Chervany, "The Meanings of Trust," http://www.misrc.umn.edu/wpaper/wp96-04.htm, University of Minnesota, Management Information Systems Reseach Center, Tech. Rep. MISRC Working Paper Series 96-04, 1996.

[9] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," *IEEE Communications Surveys and Tutorials*, vol. 3, 2000.

[10] A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision (to appear)," *Decision Support Systems*, vol. 00, no. 00, pp. 00–00, 2005.

[11] P. Zimmermann, *The Official PGP User's Guide*. MIT Press, 1995.

[12] M. Reiter and S. Stubblebine, "Toward acceptable metrics of authentication," in *Proceedings of the 1997 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1997.

[13] A. Jøsang, "The right type of trust for distributed systems," in *Proc. of the 1996 New Security Paradigms Workshop*, C. Meadows, Ed. ACM, 1996.

[14] M. Kinateder, E. Baschny, and K. Rothermel, "Towards a Generic Trust Model," in *Proc. of the Third International Conference on Trust Management*, ser. LNCS, P. Herrmann, V. Issarny, and S. Shiu, Eds., no. 3477. Versailles, France: Springer-Verlag, May 2005, pp. 177–192.

[15] B. Christianson and W. S. Harbison, "Why Isn't Trust Transitive?" in *Proceedings of the Security Protocols International Workshop*. University of Cambridge, 1996.

[16] A. Jøsang and S. Pope, "Semantic Constraints for Trust Tansitivity," in *Proceedings of the Asia-Pacific Conference of Conceptual Modelling (APCCM) (Volume 43 of Conferences in Research and Practice in Information Technology)*, S. Hartmann and M. Stumptner, Eds., Newcastle, Australia, February 2005.

[17] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, May 2003.

[18] R. Levien, "Attack Resistant Trust Metrics," Ph.D. dissertation, University of California at Berkeley, 2004.

[19] C.-N. Ziegler and G. Lausen, "Spreading Activation Models for Trust Propagation," in *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04)*, Taipei, March 2004.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.

[21] A. Clausen, "The Cost of Attack of PageRank," in *Proceedings of The International Conference on Agents, Web Technologies and Internet Commerce (IAWTIC'2004)*, Gold Coast, July 2004.

[22] E. Adar and B. Huberman, "Free Riding on Gnutella," *First Monday (Peer-reviewed Journal on the Internet)*, vol. 5, no. 10, p. 8, Otober 2000.

[23] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*, H. Paques, L. Liu, and D. Grossman, Eds. ACM Press, 2001, pp. 10–317.

[24] F. Cornelli *et al.*, "Choosing Reputable Servents in a P2P Network," in *Proceedings of the eleventh international conference on World Wide Web (WWW'02)*. ACM, May 2002.

[25] E. Damiani *et al.*, "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks," in *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS'02)*. ACM, 2002, pp. 207–216.

[26] D. Fahrenholtz and W. Lamesdorf, "Transactional Security for a Distributed Reputation Management System," in *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web)*, vol. LNCS 2455. Springer, September 2002, pp. 214–223.

[27] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, 2003.

[28] C. Liau *et al.*, "Efficient Distributed Reputation Scheme for Peer-to-Peer Systems." in *Proceedings of the 2nd International Human.Society@Internet Conference (HSI)*, vol. LNCS 2713. Springer, 2003, pp. 54–63.

[29] P. Flocchini and F. Luccio, "Routing in Series Parallel Networks," *Theory of Computing Systems*, vol. 36, no. 2, pp. 137–157, 2003.

[30] V. Latora and M. Marchiori, "Economic small-world behavior in weighted networks," *The European Physical Journal B*, vol. 32, pp. 249–263, 2003.

[31] A. Jøsang, "An Algebra for Assessing Trust in Certification Chains," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99)*, J. Kochmar, Ed. The Internet Society, 1999.

[32] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model," in *Proceedings of the 1997 New Security Paradigms Workshop*. ACM, 1997, pp. 48–60.

[33] V. Cahill, B. Shand, E. Gray, *et al.*, "Using Trust for Secure Collaboration in Uncertain Environments," *Pervasive Computing*, vol. 2, no. 3, pp. 52–61, July-September 2003.

[34] M. Carbone, M. Nielsen, and V. Sassone, "A Formal Model for Trust in Dynamic Networks," in *Proc. of International Conference on Software Engineering and Formal Methods (SEFM'03)*, Brisbane, September 2003.

[35] D. Manchala, "Trust Metrics, Models and Protocols for Electronic Commerce Transactions," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, 1998.

[36] A. Jøsang, "A Logic for Uncertain Probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 3, pp. 279–311, June 2001.

[37] R. Kohlas and U. Maurer, "Confidence valuation in a public-key infrastructure based on uncertain evidence," in *Proceedings of the International Workshop on Theory and Practice of Public-Key Cryptography*. Springer, 2000.

[38] M. Kinateder and K. Rothermel, "Architecture and Algorithms for a Distributed Reputation System," in *Proc. of the First International Conference on Trust Management*, ser. LNCS, P. Nixon and S. Terzis, Eds., no. 2692. Crete, Greece: Springer-Verlag, May 2003, pp. 1–16.

[39] E. Gray, P. O'Connell, C. Jensen, S. Weber, J.-M. Seigneur, and C. Yong, "Towards a Framework for Assessing Trust-Based Admission Control in Collaborative Ad Hoc Applications," Dept. of Computer Science, Trinity College Dublin, Technical Report 66, dec 2002.

[40] S. Marsh, "Formalising Trust as a Computational Concept," Ph.D. dissertation, University of Stirling, 1994.

[41] S. Pope and A. Jøsang, "Analsysis of competing hypotheses using subjective logic," in *Proceedings of the 10th International Command and Control Research and Technology Symposium*. United States Department of Defense Command and Control Research Program (DoDCCRP), 2005.

[42] M. DeGroot and M. Schervish, *Probability and Statistics (3rd Edition)*. Addison-Wesley, 2001.

[43] A. Jøsang and S. Pope, "Normalising the Consensus Operator for Belief Fusion," in *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, Sydney 2005.

[44] A. Jøsang, S. Pope, and M. Daniel, "Conditional deduction under uncertainty," in *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, 2005.

[45] A. Jøsang, "Subjective Evidential Reasoning," in *Proceedings of the International Conference on Information Processing and Management of Uncertainty (IPMU2002)*, Annecy, France, July 2002.

[46] A. Jøsang, M. Daniel, and P. Vannoorenberghe, "Strategies for Combining Conflicting Dogmatic Beliefs," in *Proceedings of the 6th International Conference on Information Fusion*, X. Wang, Ed., 2003.

[47] A. Jøsang and R. Ismail, "The Beta Reputation System," in *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002.

[48] A. Withby, A. Jøsang, and J. Indulska, "Filtering Out Unfair Ratings in Bayesian Reputation Systems," *The Icfain Journal of Management Research*, vol. 4, no. 2, pp. 48–64, 2005.