Measures

Valuation of Trust in Open Networks

Thomas Beth Malte Borcherding* Birgit Klein

European Institute for System Security
University of Karlsruhe, Germany

Abstract. Authentication in open networks usually requires participation of trusted entities. Many protocols allow trust to be obtained by recommendation of other entities whose recommendations are known to be reliable. To consider an entity as being trustworthy, especially if there have been several mediators or contradicting recommendations, it is necessary to have a means of estimating its trustworthiness. In this paper we present a method for the valuation of trustworthiness which can be used to accept or reject an entity as being suitable for sensitive tasks. It constitutes an extension of the work of Yahalom, Klein and Beth ([YKB93]).

Keywords: Trust values, Trust measures, Distributed systems

1 Introduction

Communication in open networks often requires information about the trustworthiness of the participating entities, especially when authentication protocols need to be performed. If, for example, user A receives a message signed allegedly by user B without having B's verification data at hand[1], she can ask a trusted *authentication server* (AS) of her choice to confirm the signature. In large distributed networks it will frequently happen that this AS does not have the required data in its database and will have to ask another trusted AS for assistance. This AS, in turn, can repeat this procedure until eventually a sufficiently informed AS is reached and the data can be handed to A. For A to believe in the received data being authentic she has to trust the terminal AS and hence the sequence of mediating ASs (the *recommendation path*). The longer the path becomes, the less trustworthy the final entity intuitively will be.

Depending on the task which A wants an entity of such a path to perform, she has to decide whether it is sufficiently trustworthy. Usually there is a maximum value one is willing to risk within a certain trust relationship. To determine such a maximum value, one has to estimate *degrees* of trust.

A valuation of trustworthiness also becomes relevant when different entities offer different allegedly authentic data of the same entity. In such cases the trustworthiness of these entities needs to be compared.

---

* Now working at the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, Germany

[1] B's verification data corresponds to its public key when using a digital signature scheme.

In the past there have been several approaches to describe trust formally (e.g. [BAN89, GNY90, Ran92, YKB93]). The result were logics which can be used to draw conclusions from given initial trust relationships like who is trustworthy and which public data belongs to whom. These logics lack the notion of degrees of trust; an entity is considered either trustworthy or not.

In [TH92] Tarah and Huitema propose a valuation of *certification paths* (which are related to the recommendation paths mentioned above) and give some hints on how to perform such a valuation. As examples for measures they suggest the minimum of the involved trust values or the length of the certification path.

Some questions remain open, e.g. the meaning of an actual value of trustworthiness and how different recommendations about an entity with different degrees of trustworthiness can be combined to yield a unique value.

In this paper we introduce a measure of trustworthiness based on the work of Yahalom, Klein and Beth ([YKB93]) and consider the aforementioned questions. Further work on this topic can be found in [Bor93].

The paper is organized as follows. In section 2 we introduce a formal representation of valued trust relationships, in sections 3 and 4 we show how new relationships and their values can be derived from already existing ones. This method of derivation is demonstrated by an example in section 5. In section 6 we show how the actual decision whether to trust an entity or not can be made. We conclude with a summary in section 7.

2 Formal Representation of Trust Relationships

In this section we introduce a formal representation of valued trust relationships. It is an extension of the representation used in [YKB93]. We assume the following underlying model of a distributed system:

The system consists of entities which communicate via links. Each entity has a unique identifier and may have a secret which can be used for authentication purposes. The entities can generate, read and modify any message on any link. Entities may have some computational power e.g. for the encryption and decryption of messages. Some entities are distinguished as *authentication servers (AS)* as they support the authentication of other entities.

To model degrees of trust, we need the notion of *numbers of positive/negative experiences.* We assume that an entity can assign a certain number (value) to each task it entrusts to another entity. This number can be thought of as the number of ECU being lost when the task is not fulfilled. Each lost or not lost entrusted ECU increments the number of positive or negative experiences by one.

2.1 Trust Classification

In [YKB93] it is pointed out that there is no need to trust an entity completely if one expects it only to perform a limited task. After examining some authentication protocols, the following classes were identified:

- **key generation**: Providing good quality keys to be used with some agreed upon cryptographic function.
- **identification**: Correctly associating entities (respectively their unique identifiers) with their identifying data, e.g. public or shared keys.
- **keeping secrets**: Keeping classified information secret.
- **non interference**: Not interfering in other entities' sessions (e.g. by eavesdropping or impersonating).
- **clock synchronization**: Maintaining close clock synchronization.
- **performing algorithmic steps**: Following protocol specifications correctly.

Trust with respect to one of these classes is independent of trust with respect to other classes. It may be perfectly reasonable to trust an authentication server in a different domain with respect to identifying entities in that domain without trusting its key generation capabilities.

2.2 Direct Trust and Recommendation Trust

For each of the classes of trust there are two *types* of trust: direct trust and recommendation trust. To trust an entity directly means to believe in its capabilities with respect to the given trust class. Recommendation trust expresses the belief in the capability of an entity to decide whether another entity is reliable in the given trust class and in its honesty when recommending third entities.

Recommendation trust can be granted in a restricted manner. Constraints can be imposed on the properties of the recommending entities further along the path as well as on the entities which are eventually recommended as being directly trustworthy. These properties can include the very names of entities, their domains or the number of entities on the path so far. The constraints are used to express *distrust* towards entities.

Due to the different notions of direct trust and recommendation trust, we present their formal representations separately.

Direct Trust

$$P \; trusts_x^{seq} \; Q \; value \; V$$

A direct trust relationship exists if *all* experiences with Q with regard to trust class x which P knows about are positive experiences. *seq* is the sequence of entities who mediated the experiences[2] (the recommendation path) excluding P and Q. V is a value of the trust relationship which is an estimation of the probability that Q behaves well when being trusted. It is based on the number of positive experiences with Q which P knows about.

Let p be the number of positive experiences. The value v_z of these experiences is computed as follows:

$$v_z(p) = 1 - \alpha^p \; . \tag{1}$$

[2] We regard a recommendation as propagation of positive experiences.

This value is the probability that Q has a *reliability* of more than α, founded on the information P possesses about Q. The reliability is the probability that Q turns out to be reliable when being entrusted with a single task, i.e. a task of value 1. α should be chosen reasonably high to ensure sufficiently safe estimations.

Proposition 1: *Let* ok *be a variable for the number of positive experiences and* r *be the reliability of an entity. Assume further that* r *is distributed uniformly over the set of all entities. Then the probability that* r *is greater than* α *on the condition that* ok *equals* p *is* $1 - \alpha^{p+1}$.

Proof:

$$\begin{aligned} P(r > \alpha | ok = p) &= \frac{P(r > \alpha, ok = p)}{P(ok = p)} = \frac{\int_\alpha^1 x^p \mathrm{d}x}{\int_0^1 x^p \mathrm{d}x} \\ &= \frac{(p+1)^{-1}(1 - \alpha^{p+1})}{(p+1)^{-1}} \\ &= 1 - \alpha^{p+1} \end{aligned} \tag{2}$$

□

We use the formula $1 - \alpha^p$ in our model instead of $1 - \alpha^{p+1}$ to enforce a trust degree of zero for an unknown entity. This slight deviation from the stated semantics (which is meaningless for large p) buys us a much more convenient model for the trust values.

The value of direct trust has the following (realistic) property: A single additional positive experience has more influence on a low value than on a high value. Obviously, a positive experience with a stranger offers more information than the same experience with a reliable friend.

If there have been negative experiences, there is no trust relationship. We did not model *values of distrust* since a distrusted entity should not be trusted with anything at all, no matter how small the non-trustworthiness may be. Instead, a distrusted entity can be excluded from being recommended by other entities by using the target constraints described in the next paragraph.

Recommendation Trust

P *trusts.rec*$_x^{seq}$ Q *when.path* S_p *when.target* S_t *value* V

A recommendation trust relationship exists if P is willing to accept reports from Q about experiences with third parties with respect to trust class x. This trust is restricted to experiences with entities in S_t (the *target constraint set*) mediated by entities in S_p (the *path constraint set*). Again, *seq* is the sequence of entities who mediated the recommendation trust. V is the value of the trust relationship. It represents the portion of offered experiences that P is willing to accept from Q and is based on the experiences P has had with the entities *recommended* by Q.

Given numbers of positive and negative experiences p and n, respectively, with the recommended entities, the recommendation trust value v_r is computed according to the following formula:

$$v_r(p,n) = \begin{cases} 1 - \alpha^{p-n} & \text{if } p > n \\ 0 & \text{else} \end{cases} .$$

This value can be regarded as a degree of similarity between P and Q, taking into account that different entities may have different experiences with a third party. An entity which usually sends unclassified messages will less often be confronted with treachery than an entity which transmits secret information. Given such a dissimilarity, the latter entity will not put much weight on what the former is telling about other entities' discretion.

The experiences with the recommending entity are formed by the experiences with the recommended entities. If a recommended entity behaves well, we have the experience of a valuable recommendation. Otherwise, the recommendation seems to be questionable, but we cannot deduce that the recommending entity has been lying from its point of view. Hence there is no immediate reason to reject further recommendations; it is sufficient to state a certain dissimilarity and to lower the trust value. This is modeled by the following properties of the given formula:

- $v_r(p,n) = 0$ for $p = 0$.
- $v_r(p,n)$ approaches 1 with growing p and fixed n.
- $v_r(p,n)$ approaches 0 with growing n and fixed p.

If the negative experiences outnumber the positive experiences, the value becomes zero and the entity is excluded from the recommendation constraint set.

Representation of the Constraint Sets

The constraint sets need not be stated explicitly. It suffices to specify predicates which decide the membership of an entity to the set in question. Such a predicate could be "is-child-of(x, A)" which would be true if x is a child of A in a given hierarchy and hence describes implicitly the set of all children of A. These predicates have to be decidable to be useful in this context.

The predicates may depend on the trust expressions they are evaluated in. If the predicate in the example above is changed into "is-child-of(x, *current-entity*)" it defines the set of children of the trusted entity. As will be described later, predicates can be taken over from initial trust expressions into derived ones with different trusted entities so that the same predicate applies to different instances of *current-entity*. When used as path constraint set, the given sample predicate would restrict the recommendation path to a descending path in the given hierarchy.

3 Deriving Trust Relationships

The representation of trust by trust expressions leads to rules which describe how new trust comes into being when recommendation is performed. With the help of these rules one can derive new trust relationships from a given set of initial relationships. Since the trust model itself offers no inherent strategy to derive a trust relationship between two entities, derivation algorithms need to be added. In this section we introduce the rules of inference and describe two derivation algorithms.

3.1 Description of the Rules of Inference

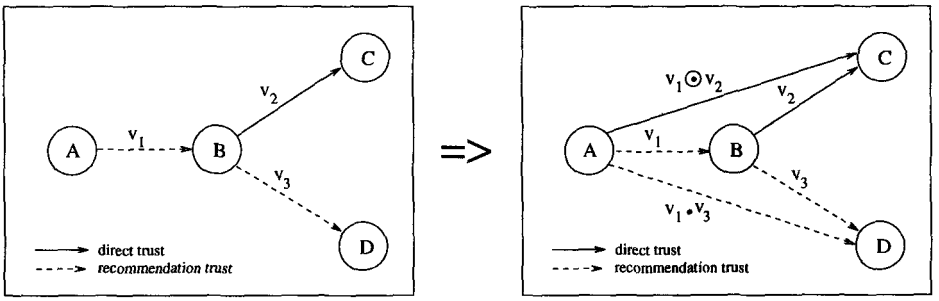

Fig. 1. Derivation of Trust Relationships

Before we introduce the rules of inference formally, we will describe them by the help of an example (see figure 1).

We start with the following initial trust relationships:

1. A *trusts.rec*$_x^{seq_1}$ B *when.path* S_{p_1} *when.target* S_{t_1} *value* V_1
2. B *trusts*$_x^{seq_2}$ C *value* V_2
3. B *trusts.rec*$_x^{seq_3}$ D *when.path* S_{p_3} *when.target* S_{t_3} *value* V_3

Deriving Direct Trust

From 1. and 2. a new direct trust relationship from A to C can be derived provided that

- C is in S_{t_1} (C is a possible target)
- All entities in seq_2 are also in S_{p_1} (no mediator of the direct trust is excluded by the recommendation trust)
- seq_1 contains no entity from seq_2 (noncyclicity)

The new recommendation path is composed of seq_1, B and seq_2 to $seq_1 \circ B \circ seq_2$. The value of the new relationship is computed according to the following formula:

$$V_1 \odot V_2 = 1 - (1 - V_2)^{V_1} \ . \tag{3}$$

This formula is a result of the computation of the direct trust values following 2.2 and the semantics of the recommendation trust values. If V_2 is based on p positive experiences, the following equation holds:

$$V_1 \odot V_2 = 1 - (1 - (1 - \alpha^p))^{V_1} = 1 - \alpha^{V_1 \cdot p} \ .$$

Thus the new value is based on the equivalent of "$p \cdot V_1$" experiences.

Deriving Recommendation Trust

From 1. and 3. a new recommendation trust relationship from A to D can be derived, provided that

- D and all entities in seq_3 are also in S_{p_1} (D and the mediators of the trust in D are not excluded)
- D and all entities in seq_3 are not in seq_1 (noncyclicity)

The new trust relationship has a recommendation path of $seq_1 \circ B \circ seq_3$, a path constraint set of $S_{p_1} \cap S_{p_2}$, a target constraint set of $S_{t_1} \cap S_{t_2}$ and a value of $V_1 \cdot V_3$.

The multiplication of the values ensures that the trust value descends as the recommendation path grows. The proposition "If someone whose recommendations are rather useless to me recommends someone whose recommendations are rather useless to him, then the recommendations of the recommended one are even less useful to me" may not be true in all cases since the usefulness of the recommendation may well be higher, but it helps to stay on the safe side.

We have required noncyclicity for the following reason: Cycles would lead to an infinite number of possible recommendation paths between two entities with trust values approaching zero as the pathlength increases. In section 4 we introduce a method on how to combine values of trust relationships between two entities which would be undermined by an arbitrary number of arbitrarily small trust values. Since cycles add no new information on who is trustworthy, they can safely be left out.

The intersection of the constraint sets is intended to preserve the type of representation used in the original constraint sets. Thus, when using the implicit representation, the new set is the conjunction of the predicates in the two sets. This is necessary to retain the meaning of the variables which depend on the context (the trust expression) they are evaluated in. The intersection of explicit sets yields another explicit set, as one might expect.

If the sets to be intersected are of different type, the explicit set can easily be transformed to an implicit representation, e.g. the explicit set "$\{A, B\}$" can be transformed to "$x = A$ or $x = B$".

3.2 The Rules of Inference

Here we present the rules of inference formally:

RULE1: (NEW DIRECT TRUST)

$$
\begin{array}{l}
\quad P \; trusts.rec_x^{seq_1} \; Q \; when.path \; S_p \; when.target \; S_t \; value \; V_1 \\
\wedge \; Q \; trusts_x^{seq_2} \; R \; value \; V_2 \\
\wedge \; R \in_s S_t \\
\wedge \; \forall X : (X \in_l seq_2 \Rightarrow (X \in_s S_p \wedge X \notin_l P \circ seq_1)) \\
\Rightarrow P \; trusts_x^{seq_1 \circ Q \circ seq_2} \; R \; value \; (V_1 \odot V_2)
\end{array}
$$

RULE2: (NEW RECOMMENDATION TRUST)

$$
\begin{array}{l}
\quad P \; trusts.rec_x^{seq_1} \; Q \; when.path \; S_{p_1} \; when.target \; S_{t_1} \; value \; V_1 \\
\wedge \; Q \; trusts.rec_x^{seq_2} \; R \; when.path \; S_{p_2} \; when.target \; S_{t_2} \; value \; V_2 \\
\wedge \; \forall X : (X \in_l seq_2 \circ R \Rightarrow (X \in_s S_{p_1} \wedge X \notin_l P \circ seq_1)) \\
\Rightarrow P \; trusts.rec_x^{seq_1 \circ Q \circ seq_2} \; R \\
\quad when.path \; (S_{p_1} \cap S_{p_2}) \; when.target \; (S_{t_1} \cap S_{t_2}) \; value \; (V_1 \cdot V_2)
\end{array}
$$

The symbol $\circ$ denotes concatenation of sequences or appending of elements to a sequence, the predicates $\in_l$ and $\in_s$ denote the membership of elements to a sequence or to a set, respectively.

3.3 Associativity of the Derivation Rules

The choice of the derivation strategy has no influence on the derivable trust expressions. That means that from a given sequence of trust expressions one can either derive a unique trust expression from the first trusting entity to the last trusted entity or none at all, independent of the order of rule application.

Thus it does not matter whether one first derives the recommendations along a path and then the desired direct trust, or whether one derives direct trust expressions from the end of a path towards its beginning.

To show this, we start with

Definition 2: Let $\mathcal{R}$ and $\mathcal{D}$ denote the sets of recommendation trust expressions and direct trust expressions, respectively. We define two functions $R_1 : \mathcal{R} \times \mathcal{D} \rightarrow \mathcal{D}$ and $R_2 : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$ as

$$
R_1(rx_1, dx_2) := \begin{cases} \text{Result of Rule1 when applied to } rx_1 \text{ and } dx_2 & \text{if applicable} \\ \bot & \text{else} \end{cases}
$$

$$
R_2(rx_1, rx_2) := \begin{cases} \text{Result of Rule2 when applied to } rx_1 \text{ and } rx_2 & \text{if applicable} \\ \bot & \text{else} \end{cases}
$$

Proposition 3: *For $rx_1, rx_2, rx_3 \in \mathcal{R}$ and $dx_3 \in \mathcal{D}$ the following equations hold:*

(1) $R_1(R_2(rx_1, rx_2), dx_3) = R_1(rx_1, R_1(rx_2, dx_3))$
(2) $R_2(R_2(rx_1, rx_2), rx_3) = R_2(rx_1, R_2(rx_1, rx_3))$

Proof:
For the proof of (1), we consider trust relationships $rx_1, rx_2 \in \mathcal{R}$ and $dx_3 \in \mathcal{D}$:

$rx_1 = P_1\ trusts.rec_x^{seq_1}\ Q_1\ when.path\ S_{p_1}\ when.target\ S_{t_1}\ value\ V_1$
$rx_2 = P_2\ trusts.rec_x^{seq_2}\ Q_2\ when.path\ S_{p_2}\ when.target\ S_{t_2}\ value\ V_2$
$dx_3 = P_3\ trusts_x^{seq_3}\ Q_3\ value\ V_3.$

First we show the equality in case the rules are applicable:

$$R_1(R_2(rx_1, rx_2), dx_3) = P_1\ trusts_x^{(seq_1 \circ Q_1 \circ seq_2) \circ Q_2 \circ seq_3}\ Q_3\ value (V_1 \cdot V_2) \odot V_3$$
$$R_1(rx_1, R_1(rx_2, dx_3)) = P_1\ trusts_x^{seq_1 \circ Q_1 \circ (seq_2 \circ Q_2 \circ seq_3)}\ Q_3\ value V_1 \odot (V_2 \odot V_3)$$

These two expressions are equal because the concatenation of sequences is associative and

$$\begin{aligned} V_1 \odot (V_2 \odot V_3) &= V_1 \odot (1 - (1 - V_3)^{V_2}) \\ &= 1 - (1 - (1 - (1 - V_3)^{V_2}))^{V_1} = 1 - (1 - V_3)^{V_1 \cdot V_2} \\ &= (V_1 \cdot V_2) \odot V_3. \end{aligned}$$

We now show that the conditions for $R_1(R_2(rx_1, rx_2), dx_3) \neq \perp$ are equivalent to the conditions for $R_1(rx_1, R_1(rx_2, dx_3)) \neq \perp$:

Conditions for $R_2(rx_1, rx_2) \neq \perp$:
(1) $Q = P_2$
(2) $x \in seq_2 \circ Q_2 \Rightarrow x \in S_{p_1} \wedge x \notin P_1 \circ seq_1$
Additional conditions for $R_1(R_2(rx_1, rx_2), dx_3) \neq \perp$:
(3) $Q_2 = P_3$
(4) $Q_3 \in S_{t_1} \cap S_{t_2}$
(5) $x \in seq_3 \Rightarrow x \in S_{p_1} \cap S_{p_2} \wedge x \notin P_1 \circ seq_1 \circ Q_1 \circ seq_2$

Conditions for $R_1(rx_2, dx_3) \neq \perp$:
(6) $Q_2 = P_3$
(7) $Q_3 \in S_{t_2}$
(8) $x \in seq_3 \Rightarrow x \in S_{p_2} \wedge x \notin P_2 \circ seq_2$
Additional conditions for $R_1(rx_1, R_1(rx_2, dx_3)) \neq \perp$:
(9) $Q_1 = P_2$
(10) $Q_3 \in S_{t_1}$
(11) $x \in seq_2 \circ Q_2 \circ seq_3 \Rightarrow x \in S_{p_1} \wedge x \notin P_1 \circ seq_1$

$\{(1), (2), \ldots, (5)\} \Rightarrow \{(6), (7), \ldots, (11)\}$ holds because (3) $\Rightarrow$ (6), (4) $\Rightarrow$ (7), (1) $\wedge$ (5) $\Rightarrow$ (8), (1) $\Rightarrow$ (9), (4) $\Rightarrow$ (10) and (2) $\wedge$ (5) $\Rightarrow$ (11).
$\{(6), (7), \ldots, (11)\} \Rightarrow \{(1), (2), \ldots, (5)\}$ holds because (9) $\Rightarrow$ (1), (11) $\Rightarrow$ (2), (6) $\Rightarrow$ (3), (7) $\wedge$ (10) $\Rightarrow$ (4) and (8) $\wedge$ (9) $\wedge$ (11) $\Rightarrow$ (5).

The proof of proposition (2) can be carried out in a similar manner and is thus omitted. □

3.4 Trust Derivation Algorithms

To track down all entities which can be trusted by an entity P with respect to a trust class x, one has to go along all noncyclic paths in the network which consist of trust relationships of the desired class, start at P, follow the collected constraints and end with a direct trust relationship.

For this purpose a trust derivation algorithm is proposed in [YKB93] which derives all possible direct trust relationships with an entity P as trusting entity from a given set of initial trust relationships. The algorithm tries for each recommendation trust expression to derive as many new trust expressions as possible. Then the considered recommendation trust is removed from the set and the new recommendation trusts are inserted into it. This process is continued until the set is empty (which happens in finite time since the paths are not to contain cycles).

The trusted entities in the derived direct trust expressions are collected in an extra set which finally contains all entities P may trust in the given trust class.

A disadvantage of this algorithm is its exponential complexity (in the number of nodes) with no remedy to be expected since the problem has proven to be NP-complete ([Bor93]). So it was necessary to make use of some sensible heuristics.

In [YKB94] a distributed algorithm is proposed which can handle all types of networks but is especially designed for tree-like structures as they frequently appear in the real world. In the case of a pure tree, the complexity is logarithmic. To accelerate the search when the tree structure is disturbed, it makes use of routing tables which store information about shortcuts in the structure.

These algorithms can easily be adopted to support the extended model of valued trust relationships.

4 Combination of Trust Values

Since there are not necessarily unique recommendation paths, there will sometimes be several (derived) trust relationships of the same trust class between two entities. They will usually have different values, so one has to find a way to draw a consistent conclusion. The introduced semantics of the trust values lead to the result that different values do not imply a contradiction, but can be used as collective information to compute a combined value. In this section we show how this combination can be performed. Again, we consider recommendation trust and direct trust separately.

4.1 Recommendation Trust

Suppose A has several recommendation trust relationships to B. That means it has different estimations about its similarity with B which could have been derived via different recommendation paths. To combine these values to a unique

value one has a choice of possible averages. This family is represented by

$$\bar{V} = \left(\frac{\sum_{i=1}^{n} V_i^{\beta}}{n} \right)^{1/\beta}$$

with $\beta \in \mathbf{R} \setminus \{0\}$ as free parameter. Special cases are maximum and minimum with $\beta \to \infty$ and $\beta \to -\infty$, respectively, harmonic, geometric and arithmetic mean with $\beta = -1$, $\beta \to 0$ and $\beta = 1$, respectively.

In order to avoid relationships with extreme values to determine the combination completely, we decided to use neither minimum nor maximum. When using the maximum of the values, a single unjustified trust with a high value could override any amount of low valued recommendations. A similar argument holds for the minimum. So the best choice seemed to be the arithmetic mean which regards all values equal.

Given n values of recommendation trust relationships between the same entities and with respect to the same trust class $V_i\,(i = 1 \ldots n)$, $V_i \neq 0$, their combination V_{com} is computed according to the following formula:

$$V_{com} = \frac{1}{n} \sum_{i=1}^{n} V_i \ . \tag{4}$$

4.2 Direct trust

Here we have several direct trust relationships between two entities with respect to the same trust class. It is necessary to classify the trust expressions with respect to the last recommending entity on the recommendation path to get a result which conforms to the semantics of the trust values.

Let $P_i\,(i = 1 \ldots m)$ be the different last entities on the recommendation paths and, in case of empty paths, the trusting entity and $V_{i,j}\,(i = 1 \ldots m,\ j = 1 \ldots n_i)$, $V_{i,j} \neq 0$ the values of the trust relationships (with n_i denoting the number of relationships having P_i as last recommending entity). The $V_{i,*}$ represent classes of trust values with each class containing the values of trust relationships with the same last recommending entity P_i.

The values of the direct trust relationships are combined according to the following formula:

$$V_{com} = 1 - \prod_{i=1}^{m} \sqrt[n_i]{\prod_{j=1}^{n_i} (1 - V_{i,j})} \tag{5}$$

This formula follows from the meaning of a recommendation: There exists an entity which has had some experiences with the entity to be recommended. These experiences have been propagated along the recommendation paths, undergoing a reduction corresponding to the values of the recommendation trusts on their way. Since there are not necessarily unique paths from one entity to another, the same experiences may be propagated to an entity several times via different

paths and with different reductions. In the previous paragraph we pointed out that these reductions can be averaged to yield a unique value.

In particular, we can split the $V_{i,j}$ into their inherent positive experiences p_i directly from entities P_i and diminishing factors $\tilde{V}_{i,j}$ which fulfill the equation

$$V_{i,j} = 1 - \alpha^{\tilde{V}_{i,j} \cdot p_i} .$$

The combined trust value is computed to

$$V_{com} = 1 - \prod_{i=1}^{m} \sqrt[n_i]{\prod_{j=1}^{n_i}(1 - V_{i,j})} = 1 - \prod_{i=1}^{m} \sqrt[n_i]{\prod_{j=1}^{n_i} \alpha^{\tilde{V}_{i,j} \cdot p_i}}$$

$$= 1 - \prod_{i=1}^{m} \alpha^{\frac{1}{n_i}(\sum_{j=1}^{n_i} \tilde{V}_{i,j}) \cdot p_i} = 1 - \alpha^{\sum_{i=1}^{m} \frac{1}{n_i}(\sum_{j=1}^{n_i} \tilde{V}_{i,j}) \cdot p_i}$$

and hence corresponds to experiences of $\sum_{i=1}^{m} \frac{1}{n_i}(\sum_{j=1}^{n_i} \tilde{V}_{i,j}) \cdot p_i$ which is exactly what we described above.

4.3 Nonmonotonicy of the Combination

The combination of values has the property of being *nonmonotonic*. If the set of trust relationships between two entities with respect to the same trust class grows, the behaviour of the combined values cannot be predicted. This property is a result of the averaging of the recommendation trust values. The average can increase or decrease as new relationships arise.

If it is unknown whether all possible trust relationships have been derived, an estimation of possible effects of new relationships on the combined value would be necessary. Hence the combination of values would imply the consideration of probability densities. Otherwise, when using the "incomplete" combined value, one can but hope that no significant changes will appear.

Considering these alternatives, the combined value should only be computed from a complete set of trust relationships. We suppose that in real world scenarios this is not a problem since the paths are few and rather determined by the existing hierarchies. Another, somewhat optimistic, solution would be not to use the arithmetic mean as average, but the maximum of the values.

5 An Example

In this example we demonstrate the derivation of trust relationships and the combination of their values. Since the example focuses on the values, we make no use of the path constraints. For simplicity, we replace the part "*when.path*[true] *when.target*[true]" in each recommendation trust expression by dots (...).

We start with the following trust relationships (see figure 2):

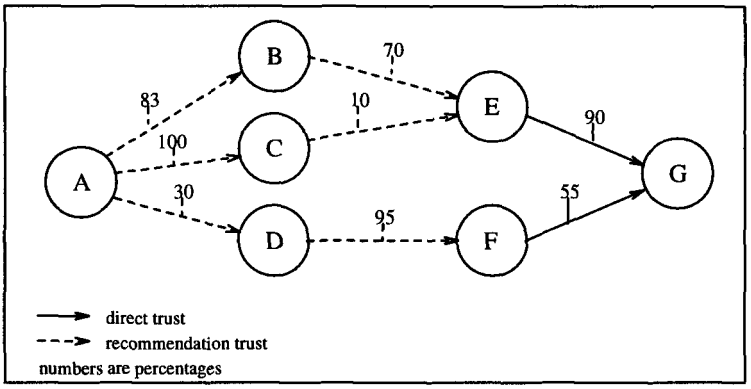

Fig. 2. Combination of Trust Values

(r1) A *trusts.rec*$_x^{\emptyset}$ B ... *value* 0.83
(r2) A *trusts.rec*$_x^{\emptyset}$ C ... *value* 1
(r3) A *trusts.rec*$_x^{\emptyset}$ D ... *value* 0.30
(r4) B *trusts.rec*$_x^{\emptyset}$ E ... *value* 0.70
(r5) C *trusts.rec*$_x^{\emptyset}$ E ... *value* 0.10
(r6) D *trusts.rec*$_x^{\emptyset}$ F ... *value* 0.95
(d1) E *trusts*$_x^{\emptyset}$ G *value* 0.90
(d2) F *trusts*$_x^{\emptyset}$ G *value* 0.55

Derivation of a Direct Trust Relationship

There are two basic strategies for the derivation of a direct trust relationship from A to G (e.g. via B and E): iterative and recursive. Both yield the same trust expressions (with the same values) as was pointed out in section 3.3. The values in this example are rounded to three digits.

Iterative Derivation: We first derive a recommendation trust from A to E, followed by the derivation of the direct trust from A to G:

(r1), (r4), Rule2 $\Rightarrow$ (r7) A *trusts.rec*$_x^B$ E ... *value* $\underbrace{0.83 \cdot 0.7}_{=0.581}$

(r7), (d1), Rule1 $\Rightarrow$ (d3) A *trusts*$_x^{B \circ E}$ G *value* $\underbrace{0.581 \odot 0.9}_{=0.738}$

Recursive Derivation: Here we start with the derivation of the direct trust from B to G which is then used to derive the direct trust from A to G:

(r4), (d1), Rule1 $\Rightarrow$ (d4) B *trusts*$_x^E$ G *value* $\underbrace{0.7 \odot 0.9}_{=0.800}$

(r1), (d4), Rule1 $\Rightarrow$ (d5) $A\ trusts_x^{B \circ E}\ G\ value\ \underbrace{0.83 \odot 0.800}_{=0.738}$

Combination of Direct Trust Relationships:

The following direct trust relationships from A to G can be derived using either strategy:

(d3) $A\ trusts_x^{B \circ E}\ G\ value\ 0.738$
(d4) $A\ trusts_x^{C \circ E}\ G\ value\ 0.206$
(d5) $A\ trusts_x^{D \circ F}\ G\ value\ 0.204$

There are two different entities at the end of the recommendation paths: E and F. So we identify two classes of trust expressions, the first consisting of (d3) and (d4), the second being solely (d5).

The computation of the combined value yields

$$V_{com} = 1 - \sqrt{(1 - 0.738) \cdot (1 - 0.206)} \cdot (1 - 0.204) = 0.637.$$

6 How to Use the Values

In this section we show how the introduced values can be used to decide whether or not an entity is sufficiently trustworthy with respect to a certain task.

As mentioned earlier, we assume that the value of each task can be measured in units, e.g. in ECU which are lost when the task is performed incorrectly. Our estimations about the reliability of entities were made relative to tasks consisting of a single unit. If we wish to entrust a task consisting of T units, the trusted entity has to fulfill T "atomic" tasks in order to complete the whole task. Bearing that in mind, we can estimate the risk when entrusting a task to an entity.

Let f denote the density of an entity's reliability r provided it passed p tests successfully ($ok = p$). Since $P(r \leq \alpha | ok = p) = \alpha^{p+1}$ (see equation (2)), we have that $f = (p+1)r^p$. From this equation and equation (1) follows that when T units are entrusted to an entity with trust value v, the average loss l is

$$\begin{aligned} l(\alpha, v, T) &= \int_0^1 (log_\alpha(1-v) + 1) \cdot r^{log_\alpha(1-v)} \cdot T \cdot (1 - r^T)\ dr \\ &= \frac{T^2}{log_\alpha(1-v) + T + 1}. \end{aligned}$$

If, for example, $\alpha = 0.99$, $v = 0.637$ and we wish to entrust a task worth 100 units then we must be willing to risk $l(0.99, 0.637, 100) = 49.5$ units.

7 Summary

In this paper we introduced the notion of valuated trust as an extension of the trust model of Yahalom, Klein and Beth [YKB93]. We pointed out that the semantics of direct trust values differ from that of recommendation trust values: the former expresses the probability that an entity is working with a certain minimum reliability, the latter states how useful the recommendations of the trusted entity are expected to be.

The introduction of trust values gives rise to the problem that recommendations about one entity can differ in value, depending on the recommending entities. We have shown that these different values are not contradictory but can be combined to a single value which considers the information of all the respective recommending entities.

When trust values are to be used in real systems, it is necessary to evaluate the entrusted tasks. We have shown how the trust values can be used to estimate the risk when such a task is to be assigned to an entity.

Using the trust values, a flexible policy can be applied which allows to select appropriately trustworthy entities depending on the task's value. Combined with the notion of different trust classes the requirements for trusted entities can be kept to a minimum.

Acknowledgment

We would like to thank Raphael Yahalom and the anonymous referees for their valuable comments.

References

[BAN89] M. Burrows, M. Abadi, R. Needham, "A Logic of Authentication", *Proc. of the 12th ACM Symposium on Operating Systems Principles*, Litchfield Park, Arizona, 1989. Published as ACM Operating Systems Review, 23 no 5 (1989).

[Bor93] M. Borcherding, "Ermittlung verschieden vertrauenswürdiger Pfade in offenen Netzen", Diploma thesis at the European Institute for System Security, University of Karlsruhe, 1993 (in German).

[GNY90] L. Gong, R. Needham, R. Yahalom, "Reasoning about Belief in Cryptographic Protocols", *Proc. 1990 IEEE Symp. on Research in Security and Privacy*, 234-248.

[Ran92] P. V. Rangan, "An Axiomatic Theory of Trust in Secure Communication Protocols", *Computers and Security* 11 (1992) 163-172, Elsevier Science Publishers Ltd., Oxford 1992.

[TH92] A. Tarah, Ch. Huitema, "Associating Metrics to Certification Paths", *Proceedings of the Second European Symposium on Research in Computer Security (ESORICS) 1992*, 175-189, Springer LNCS 648, Berlin 1992.

[YKB93] R. Yahalom, B. Klein, Th. Beth, "Trust Relationships in Secure Systems – A Distributed Authentication Perspective", *Proc. 1993 IEEE Symp. on Research in Security and Privacy*, 150-164.

[YKB94] R. Yahalom, B. Klein, Th. Beth, "Trust-based Navigation in Distributed Systems", to appear in: Special issue "Security and Integrity of Open Systems" of the journal "Computing Systems", 1994.