Covered Trust Values in Distributed Systems

Birgit Borcherding

European Institute for System Security, University of Karlsruhe

76128 Karlsruhe, Germany, Tel: ++49-721-6084327. Fax: ++49-721-696893.

email: kleinb@ira.uka.de

Malte Borcherding

Institute of Computer Design and Fault Tolerance, University of Karlsruhe

76128 Karlsruhe, Germany, Tel: ++49-721-6083961. Fax: ++49-721-370455.

email: malte.borcherding@informatik.uni-karlsruhe.de

Abstract

During the last years it has become recognized that trust is an essential component in the design of protocols in distributed systems. If, for example, a Key Distribution Center (KDC) is employed to distribute public keys, the receiver of such a key has to trust the KDC that the key is authentic. Sometimes, a trusted entity may recommend another entity as being trustworthy, which in turn can recommend further entities. Obviously, the longer such a trust path grows, the smaller the trust towards the final entity will be. On the other hand, the higher the number of trustworthy recommendations about an entity, the more trustworthy this entity will become. These observations have led to the introduction of *trust values*.

When deriving the value of a trust path from the trust values of the individual entities in a distributed manner, one faces the problem that the estimation of each other's trustworthiness is a private matter. Up to now, this problem has been circumvented by either ignoring it or by not allowing for the derivation of a trust path's value. In this paper, we propose a technique for the derivation of the value of a trust path without exposing the individual trust values to other entities.

Keywords

Distributed systems, trust values, authentication, delegation

1 INTRODUCTION

During the last few years, many new services for distributed systems were developed. They usually rely on some kind of trust between the entities involved. In particular this is true when an authentication service is required for secure systems.

In *Proceedings of the Working Conference on Multimedia and Communication Security*, Graz, Austria, pp. 24–31, Chapman & Hall 1995.

Most existing systems require an entity to trust another entity either completely or not at all. That means that an entity cannot express its trust restricted to certain capabilities or to a certain degree. Furthermore, there is often a fixed hierarchy of trust relationships (Birrell *et al.* 1986, CCITT 1988, Gligor *et al.* 1992, Lampson *et al.* 1991). If, for example, an entity A needs to gain trust into a remote authentication server D, it can ask an already trusted server B to recommend D. If B does not trust D, it may ask another server C along the hierarchy for a recommendation of D. This can be iterated until the desired trust relationship is established. If the trust relationships are indeed organized as a hierarchy, trust derivations are simple. But if in our example A distrusts an entity C on the shortest path to D, it will never gain trust in D since there are no alternative ways which avoid C in a hierarchy.

To overcome these problems, Yahalom, Klein, and Beth (1993) have introduced a more differentiated view of trust in distributed systems. Several trust classes are defined and individual trust relationships between entities are considered. The fixed structure of trust relationships between the servers is given up and a new trust derivation algorithm is introduced. A forwarding algorithm under slightly different assumptions is given by Yahalom, Klein, and Beth (1994).

Because trust is not only a question of trust or distrust, Beth, Borcherding, and Klein (1994) presented trust values and their derivation in distributed systems. Trust is measured between 0 and 1, and functions for the derivation and combination of trust values are given. A more limited type of trust values is already implemented in PGP (Zimmermann 1994). In this system, an entity can bind *shades* of trust to public keys of trusted entities. These shades (unknown, untrusted, marginally trusted, completely trusted) express the trust in the recommendation capabilities of others. If an entity receives a previously unknown public key with recommendations of known entities, it checks whether the weighted sum of the trusted entities' shades exceeds a certain threshold. For example, two marginally trusted entities may be deemed as credible as a completely trusted entity and it takes two completely trusted entities to judge a public key as valid. These weights can be selected by each entity. However, when recommending a key, an entity cannot express a level of trust in the key's authenticity, since the estimation of trustworthiness of the previous signers is considered private.

In this paper, we introduce a technique for the computation of trust values which takes into account the values of the respective entities on a path without revealing them. The individual trust relationships and the trust derivation algorithm of Yahalom, Klein, and Beth (1993) are used. This algorithm will be implemented in the SELANE Protocol (Beth 1991, Horster and Knobloch 1992, Bauspiess and Knobloch 1989) and be made readily available in its proposed release in the Internet.

2 PRELIMINARIES

In this section we give an overview of the formal representation of trust and trust values. For a more detailed description, the reader is referred to (Beth, Borcherding, and Klein 1994, Yahalom, Klein, and Beth 1993).

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.

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 (European Currency Unit) being lost when the task is not fulfilled. Each lost or not lost entrusted ECU increments the number of negative or positive experiences by one.

Since there is no need to trust an entity completely if one expects it only to perform a limited task, trust can be granted in distinguished *trust classes*, e.g., trust in key generation or trust in keeping secrets.

2.1 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 or towards paths with certain properties. 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* (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:

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

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. If there have been negative experiences, there is no trust relationship.

The parameter α has to be chosen once for the whole system. It determines the threshold for the reliability under consideration. If one wants the trust value to express the

*We regard a recommendation as propagation of positive experiences.

probability that an entity is more reliable than 0.9, one chooses α accordingly. Since α is known within the system, each entity can recompute its trust values locally to express different reliability thresholds.

Recommendation Trust

$$P \; trusts.rec_x^{seq} \; Q \; when.path \; \mathcal{S}_P \; when.target \; \mathcal{S}_T \; value \; v$$

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:

$$v_r(p,n) = \begin{cases} 1 - \alpha^{p-n} & \text{if } p > n \\ 0 & \text{else} \end{cases} \quad . \tag{2}$$

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 can be seen in the next section, 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. A constraint set of this type reflects distrust in *paths* with certain properties and is not coined to distrust certain entities.

2.2 Deriving trust relationships

Here we give the rules of inference. The first rule is concerned with the derivation of direct trust from recommendation trust and direct trust (i.e., recommendation of direct trust). The second rule describes how new recommendation trust can be derived from two recommendation trust expressions (i.e., recommendation of recommendation trust). Notational details are described below.

RULE1: (NEW DIRECT TRUST)

$$
\begin{aligned}
& P\ trusts.rec_x^{seq_1}\ Q\ when.path\ S_P\ when.target\ S_T\ value\ v_1 \\
\wedge\ & Q\ trusts_x^{seq_2}\ R\ value\ v_2 \\
\wedge\ & R \in_s S_T \\
\wedge\ & \forall X : (X \in_l seq_2 \Rightarrow (X \in_s S_P \wedge X \notin_l P \circ seq_1)) \\
\Rightarrow\ & P\ trusts_x^{seq_1 \circ Q \circ seq_2}\ R\ value\ (v_1 \odot v_2)
\end{aligned}
$$

RULE2: (NEW RECOMMENDATION TRUST)

$$
\begin{aligned}
& P\ trusts.rec_x^{seq_1}\ Q\ when.path\ \mathcal{S}_{P_1}\ when.target\ \mathcal{S}_{T_1}\ value\ v_1 \\
\wedge\ & Q\ trusts.rec_x^{seq_2}\ R\ when.path\ \mathcal{S}_{P_2}\ when.target\ \mathcal{S}_{T_2}\ value\ v_2 \\
\wedge\ & \forall X : (X \in_l seq_2 \circ R \Rightarrow (X \in_s \mathcal{S}_{P_1} \wedge X \notin_l P \circ seq_1)) \\
\Rightarrow\ & P\ trusts.rec_x^{seq_1 \circ Q \circ seq_2}\ R \\
& when.path\ (\mathcal{S}_{P_1} \cap \mathcal{S}_{P_2})\ when.target\ (\mathcal{S}_{T_1} \cap \mathcal{S}_{T_2})\ value\ (v_1 \cdot v_2)
\end{aligned}
$$

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. The function $\odot$ in the derivation of direct trust is defined as

$$v_1 \odot v_2 = 1 - (1 - v_2)^{v_1} \ . \tag{3}$$

This formula is based on the computation of the direct trust according to formula (1) 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 "$v_1 \cdot p$" experiences. In the next section, we will need the "inverse" function $\oslash$ defined as

$$v_2 \oslash v_1 = 1 - (1 - v_2)^{1/v_1} \tag{4}$$

for $v_1 > 0$. With this definition, the following formula holds:

$$(v_1 \odot v_2) \oslash v_1 = v_2 \ . \tag{5}$$

Combination of trust relationships
If there are several derived trust relationships towards an entity with respect to the same trust class, the values of these relationships can be *combined* to yield a unique value. This combined value is usually higher than each of the values involved in the combination. Since we do not need this technique in our paper, the reader is referred to (Beth, Borcherding, and Klein 1994)

3 COVERED TRUST VALUES

In this section we introduce a technique for the derivation of trust values which does not reveal the actual values to other entities. It shows a certain similarity to the concept of path constraints. Consider a setting with the following trust relationships (the trust class KS stands for "keeping secrets"):

A $trusts.rec_x$ B $when.path$ $\mathcal{S}_{P_1}$ $when.target$ $\mathcal{S}_{T_1}$ $value$ v_1
B $trusts.rec_x$ C $when.path$ $\mathcal{S}_{P_2}$ $when.target$ $\mathcal{S}_{T_2}$ $value$ v_2
B $trusts_{KS}$ C $value$ v_2'
C $trusts_x$ D $value$ v_3

If entity A wants to find another entity which is trustworthy with respect to trust class x with a value of at least v_{min}, it sends a message to B of the following form:

$$A \rightarrow B : \{A, B, ts_1, \tilde{\mathcal{S}}_{P_1}, \tilde{\mathcal{S}}_{T_1}, v_{min} \oslash v_1, x\}_{K_A^-}$$

$\{m\}_{K_A^-}$ denotes a message signed with the secret key of entity A. ts_1 is a timestamp and $\tilde{\mathcal{S}}_{P_1}$ and $\tilde{\mathcal{S}}_{T_1}$ denote those parts of the constraint sets which express distrust towards path properties, not towards explicit entities (see section 2.1). Distrust towards entities is dealt with later. The value $v_{min} \oslash v_1$ is the minimum trust value B has to have in the final entity.[†] Thus the value v_1 is not revealed, if A keeps v_{min} as secret which is in its own interest.

Since the message is fresh and B does not have a direct trust relationship, it has to consider to ask C for a recommendation. For C to be suitable for this task, it has to be in the set $\tilde{\mathcal{S}}_{P_1}$. Furthermore, B has to believe C to be trustworthy with respect to not collaborating with A to deduce the trust value v_2, i.e., with respect to keeping the contents of the following message secret:

$$B \rightarrow C : \{B, C, ts_2, \tilde{\mathcal{S}}_{P_1} \cap \tilde{\mathcal{S}}_{P_2}, \tilde{\mathcal{S}}_{T_1} \cap \tilde{\mathcal{S}}_{T_2}, (v_{min} \oslash v_1) \oslash v_2, x\}_{K_B^-}$$

Since C trusts D with respect to class x, it compares v_3 against the value given in the message. If the trust value is higher or equal and D lies within $\tilde{\mathcal{S}}_{T_1} \cap \tilde{\mathcal{S}}_{T_2}$, it informs B about the fact that D is sufficiently trustworthy for A. Since B did not include its distrusted entities in the target constraint set, it now has to check whether it distrusts D. If this is the case, it has to continue the search by asking other trusted entities for recommendations if possible. If B does not distrust D, it informs A about the successful search. A checks again whether it distrusts C or D and, if that is not the case, can use the services of D.

For example, the server D could be a Key Distribution Center. Users of such an authentication service are able to authenticate other principals of the system based on their individual trust relationships without revealing trust values.

The technique described above can also be applied for *delegation*. Delegation in distributed systems is a process where an entity, the *delegator*, delegates another entity, the *delegate*, to be its proxy. The delegator gives its rights or parts of its rights to the delegate which then acts in the name of the delegator. Hence, the delegator has to find an entity which is capable and trustworthy to act as delegate. If the delegator does not know such an entity in advance and is not willing to trust any entity, it can ask for recommendations for sufficiently trustworthy entities. The minimum degree of trustworthiness required and the trust classes involved can be selected individually for each task to be delegated.

In the previous paragraphs we described how to find a path to any trustworthy entity.

[†]Note that the actual trust value of the path is composed of $v_1 \odot v'$ with v' denoting the value of the trust of B in the final entity and recall that $(v_1 \odot v') \oslash v_1 = v'$.

A similar method can be applied if an entity received a message which is signed by some entities, so that the trust path is already given. To find out about the trustworthiness of the message, the receiver starts the protocol as above and includes the path of the signing entities in the path constraints. The entity will get back an answer which says whether it can accept the message as sufficiently trustworthy.

3.1 Secrecy properties

We are mainly concerned with an entity not learning how much it is trusted by others. To extract a single trust value during the given protocol, the two entities before and after the trusting entity would have to collaborate. To avoid such a collaboration is the responsibility of the entity in between. It is the only entity affected by the collaboration and it chooses the next entity on the path by itself.

A collaboration of entities which have more than one entity between them could yield the product of the trust values of the enclosed entities. This is a leakage of information which has to be accepted if the trustworthiness of path is to be evaluated.

As stated above, we avoid the exposure of distrust by splitting the constraint sets in distrust towards individual entities and distrust towards properties of the trust paths. We claim that the latter need not be kept secret, since it merely describes a derivation policy as it is used in traditional hierarchical systems. Individual distrust, on the other hand, is a private matter and is not propagated during trust derivation.

4 CONCLUSION

In this paper we have shown that considering trust values a private matter does not imply the uselessness of these values for the evaluation of trust paths. We have proposed a technique to find entities which are trustworthy to a given degree with respect to a given trust class without exposing the individual trust values. Our technique is not completely free of information leakage, but we argue that this price is not too high for being able to use the values.

5 REFERENCES

Bauspiess, F. and Knobloch, H.-J. (1989) How to Keep Authenticity Alive in a Computer Network, in *Advances in Cryptology, Proccedings of Eurocrypt 1989*, Springer, Berlin 1990, pp. 38–46.

Beth, Th., Borcherding, M., and Klein, B. (1994) Valuation of Trust in Open Networks, in *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS)*, Brighton, UK, LNCS 875, pp. 3–18, Springer-Verlag.

Beth, Th. (1991) Keeping Secrets a Personal Matter or: The Exponential Security System, in *Proceedings of the IMA-Workshop on Cryptography and Coding*, Cirencester, UK.

Birrell, A., Lampson, B., Needham, R., Schroeder, M. (1986) A Global Authentication Service Without Global Trust, in *Proceedings of the 1986 IEEE Symposium on Security and Privacy*, pp. 223–230.

CCITT (1988) X.509, The Directory - Authentication Framework, IX^{TH} Plenary Assembly, Melbourne, 14-25 November 1988.

Gligor, V. D., Luan, S.-W., and Pato, J. N. (1992) On Inter-realm Authentication in Large Distributed Systems, in *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, pp. 2–17.

Horster, P., Knobloch, H.-J. (1992) Cryptographic Protocols and Network Security, in *Security and Control: From Small Systems to Large*, Proceedings of the IFIP/Sec '92, Singapur.

Lampson, B., Abadi, M., Burrows, M., and Wobber, E. (1991) Authentication in Distributed Systems: Theory and Practice, in *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pp. 165–182.

Yahalom, R., Klein, B., and Beth, Th. (1993) Trust Relationships in Secure Systems – A Distributed Authentication Perspective, in *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy.*

Yahalom, R., Klein, B., and Beth, Th. (1994) Trust-Based Navigation in Distributed Systems. *Computing Systems* vol. 7, no. 1, pp. 45–73.

Zimmermann, Ph. (1994) PGP User's Guide, Volume I: Essential Topics, Revision 11 October 1994 for PGP version 2.6.2. Distributed by the Massachusetts Institute of Technology.

6 BIOGRAPHIES

Birgit Borcherding received the diploma in mathematics in 1990 from the University of Clausthal, Clausthal-Zellerfeld, Germany, and the Dr.rer.nat. degree in computer science in 1994 from the University of Karlsruhe, Karlsruhe, Germany. Currently she is a research assistant at the University of Karlsruhe. Her areas of research include security in distributed systems and cryptography.

Malte Borcherding received the diploma in computer science in 1993 from the University of Karlsruhe, Karlsruhe, Germany. Currently he is working as a research assistant at the University of Karlsruhe. His areas of research include authentication and fault tolerance in distributed systems.