Towards Trust-Based Acquisition of Unverifiable Information

Eugen Staab, Volker Fusenig, and Thomas Engel

Faculté des Sciences, de la Technologie et de la Communication,
Université du Luxembourg,
Campus Kirchberg, 6, rue R. Coudenhove-Kalergi, L-1359 Luxembourg
{eugen.staab,volker.fusenig,thomas.engel}@uni.lu

Abstract. We present a trust-based mechanism for the acquisition of information from possibly unreliable sources. Our mechanism addresses the case where the acquired information cannot be verified. The idea is to intersperse questions ("challenges") for which the correct answers are known. By evaluating the answers to these challenges, probabilistic conclusions about the correctness of the unverifiable information can be drawn. Less challenges need to be used if an information provider has shown to be trustworthy. This work focuses on three major issues of such a mechanism. First, how to estimate the correctness of the unverifiable information. Second, how to determine an optimal number of challenges. And finally, how to establish trust and use it to reduce the number of challenges. Our approach can resist collusion and shows great promise for various application areas such as *distributed computing* or *peer-to-peer networks*.

Keywords: Information acquisition, trust.

1 Introduction

A lot of research addresses trust that is based on direct experiences [1,2]. These direct experiences result from evaluating the outcomes of interactions with other agents. Such an evaluation however is not possible when the outcome of an interaction is information that cannot be verified, or the verification would be too costly. An example illustrates this situation:

Example 1. Agent Alice wants to know the result of $15 + 8$. However, Alice cannot compute the result because she is out of resources at the time. So Alice asks another agent Bob to do it for her. Although Bob knows how to calculate the result, he returns the wrong result 26 because he is malicious and wants to harm Alice. Consequently, Alice, who does not want to verify the result because she wanted to save resources, uses the wrong result in her further work. This will cause additional costs for her, and if she is not aware of them, she even does not classify the experience with Bob as a negative experience.

An attempt to solve this problem is to ask several agents for the desired information, to compare their answers, and to discard these if they are not the

M. Klusch, M. Pechoucek, and A. Polleres (Eds.): CIA 2008, LNAI 5180, pp. 41–54, 2008.

same (see e.g. [3]). However, this approach is sensitive to *collusion* [2], especially in settings with only few information providers, and its efficiency ought to be improved. Therefore, we propose an alternative approach. The main idea is to merge requests for information with so called "challenges" for which the correct answers are already known. The requesting agent evaluates the responses to the challenges and draws conclusions about the responses to the "real" requests. This leads to an estimation of another agent's trustworthiness which in turn can be used to reduce the number of challenges that need to be used. However, a minimal number of challenges is always retained to account for the *first-time offender problem* [4].

The remainder of the paper is organized as follows. In Section 2, we outline several application scenarios for the mechanism and show how challenges can be generated in each scenario. The mechanism for information acquisition is presented in Section 3. We discuss several issues concerning the practical use of the mechanism in Section 4. Section 5 is used to refer to related work. We conclude our paper and give an outlook on future work in Section 6.

2 Application Scenarios

This section outlines some application areas for which the mechanism shows great promise.

The mechanism can be used in cases where calculations are outsourced and the results shall not be verified. As it was motivated in Ex. 1, our mechanism can be applied to the scenario of *distributed computing*, more specifically *grid-computing* [5] or *cloud-computing* [6]. In these cases, challenges can either be provided by trusted nodes or be computed whenever the system of the requesting agent is idle.

Another application scenario is the exchange of *routing information* in *Wireless Ad Hoc Networks* [7]. As new routing information cannot be verified, the trust-based mechanism would help to enforce the provision of reliable information. The challenges can be chosen to be questions about routes that are known to exist (e.g. because packets have been sent over these routes in the recent past).

In *peer-to-peer networks*, our trust-based mechanism can be used against pollution and poisoning attacks (see [8]). Challenges would consist of requests for files that already have been verified by a human to match their description and to be free of "bad chunks". Note that in these settings, a small number of challenges for a given number of real requests would be essential for the practicability of the mechanism. Also, the verification of a partly downloaded response to the challenge should start as soon as a certain amount of packets is received.

The mechanism can also be useful for the purpose of exchanging *reputation information* such that it implements the concept of "semantic distance" which was introduced by Abdul-Rahman and Hailes [9]. The answers to challenges would be used to determine the semantic distance, which in turn could be used to weight the answers to the real requests. Related to that, the trust-based mechanism can be used in *Multi-Agent Systems*, in which *beliefs* about the world (which cannot easily be verified) are exchanged. The mechanism would prevent

the acquisition and the spread of wrong or outdated beliefs as well as it would filter out "incompatible" beliefs. In this case, challenges would be based on the *knowledge* of an agent.

3 Trust-Based Mechanism for Information Acquisition

In the following, we describe one run of the mechanism. An agent wants to get answers to m questions. We will call these questions "real requests". In our particular case, the agent will not be able to verify the correctness of the answers which he will get to the real requests (because, as mentioned earlier, he is incapable or does not want to spend resources on it). Therefore, before sending the request, the agent adds n challenges for which the answers are known to him. These challenges must be chosen in a way such that another agent is not able to easily distinguish them from real requests; how this choice can be made depends on the concrete setting (see Sect. 2 for examples). The agent merges the m requests and the n challenges into a vector of size $m + n$, in an equally distributed manner. This *request-vector* is then transferred to the information provider which is expected to reply with a *response-vector* of the same size. After reception, the response-vector is evaluated, i.e. the answers to the challenges are verified. The resulting error rate is used for two things. First, to estimate the error rate of the answers to the real requests (see Sect. 3.1) and second, as input for a trust-update algorithm (see Sect. 3.2). It is shown how an optimal number of challenges can be computed when an information provider is not known (see Sect. 3.3) and how the established trust can be used to reduce this number of (costly) challenges (see Sect. 3.4). Finally, a decision needs to be made whether the obtained response-vector is accurate enough or a new request to other information providers should be done. This decision-making process however is not part of this work (see Sect. 4 for a discussion).

3.1 Evaluation of a Response

An agent A sends a request-vector with $m + n$ questions to an information provider who is expected to reply with a response-vector of the same size. Agent A evaluates the answers to the n challenges in such a response-vector, and finds r correct and s incorrect answers, with $r + s = n$. We will call the rate of incorrect answers $\frac{s}{s+r}$ the *error rate*. As the challenges and real requests in the request-vector were equally distributed, r and s can be used to estimate a probability distribution for the error rate in the remaining part of the response-vector (the part with the *real requests*). In this work we will assume that the information provider can be described by an error-probability p_w: with probability p_w he answers independently each question incorrectly (see Sect. 4 for a discussion of this assumption). Let $\{x_i\}_{i=1}^n$ denote the evaluated answers to the challenges, so each x_i is in $\{correct, incorrect\}$. We can use *Bayes' theorem* to get the probability of each error-probability p_w:

$$P(p_w|\{x_i\}_{i=1}^n) = \frac{P(\{x_i\}_{i=1}^n|p_w)P(p_w)}{P(\{x_i\}_{i=1}^n)} \tag{1}$$

We can simplify this formula in the following way. First, the denominator can be calculated by marginalization over all rates that could have produced the given observations. Second, no prior information on p_w is given, so we have to assume all p_w to be equally probable, and so it can be left out (it's probability density function is 1 in $[0,1]$). Third, from statistical independence between all single answers it follows that $P(\{x_i\}_{i=1}^n|p_w) = p_w^s(1-p_w)^r$. As a result, we get the probability density function of the *beta distribution* with parameters $\alpha = s+1$ and $\beta = r+1$:

$$f(p_w; s+1, r+1) = \frac{p_w^s(1-p_w)^r}{\int_0^1 x^s(1-x)^r dx} \tag{2}$$

Note that this probability distribution estimates the error probability p_w of the information provider and hence estimates also the error rate of the answers to the real requests.

The mean of the beta distribution is given by $\frac{\alpha}{\alpha+\beta}$ [10]. So, having costs c_w for one wrong answer, the expected costs for using the answers to the real requests can be computed as follows:

$$E[m * c_w * f(p_w; s+1, r+1)] = m * c_w * E(f(p_w; s+1, r+1)) \tag{3}$$

$$= \frac{m * c_w * (s+1)}{s+r+2}. \tag{4}$$

3.2 Bayesian Trust-Model

In this section, we present a formal trust-model which represents an agent's trust in another agent by a *trust-value* t and the corresponding *uncertainty* u. These values will be used in Sect. 3.4 to reduce the number of challenges that are needed for a request. We describe and justify how t and u are computed and define initial values.

Let (t_{AB}, u_{AB}) denote the trust that an agent A has in an information providing agent B. In our model, a trust-value $t_{AB} \in (0,1)$ is A's estimation of the rate of correct answers in a response-vector that will eventually be received from B. The corresponding uncertainty $u_{AB} \in (0,1]$ describes how certain A is about this trustworthiness estimation t_{AB}. Having no experience with B, an agent A trusts always with $(t_{AB}, u_{AB}) = (0.5, 1)$, i.e. the initially expected error probability is 0.5 but this is believed with the highest possible uncertainty. This seems intuitive but also results from the formulas (which are defined below) when no information is given.

Several forms of representation for trust have been proposed in literature (for an overview see [11]). While uncertainty is usually represented as in our case, there are several different representations of trust-values. To us, it seems to be a mere issue of convention. However we want to justify our choice for $(0,1)$ because, based on experiences, full trust ($t = 1$) or distrust ($t = 0$) seem not to be reasonable – even though full trustworthiness and untrustworthiness are possible.

Trust-Values. Let r^B_{old} and s^B_{old} denote the amount of respectively correct and incorrect answers from past response-vectors received from an agent B. More precisely, to give recent experiences a higher importance than older ones, each single experience in r^B_{old} and s^B_{old} has been weighted with a so called "aging factor" λ (q.v. [11]). The trust-value t_{AB} is defined to be the expected value of a beta distributed random variable X with parameters $\alpha = s^B_{old} + 1$ and $\beta = r^B_{old} + 1$:

$$t_{AB} \stackrel{def}{=} E(X) = \frac{r^B_{old} + 1}{r^B_{old} + s^B_{old} + 2} \tag{5}$$

The probabilistic justification for this calculation follows directly from Section 3.1: the trust-value is the most probable error probability p_w based on past observations. Formula (5) constitutes the core part of the *trust-value* computation in the proposed mechanism. We want to emphasize the possibility to extend this computation with other approaches for trust-value computation that have been proposed numerously in literature (e.g. see [1,2]).

Uncertainty. The uncertainty u_{AB} in our model is based on the *variance* σ^2 of the beta distribution, i.e. the lower the expected variance, the lower the uncertainty. We will show that for parameters $\alpha \geq 1$ and $\beta \geq 1$, the variance of the beta distribution has the two properties that Wang and Singh [12] claim for certainty measures. However, in comparison to their approach and the one used in TRAVOS [13] that both compute integrals over a beta distribution, the variance of the beta distribution is much easier to compute. For a beta distributed random variable X with parameters α and β, the variance σ^2 is given by ([10]):

$$\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{6}$$

As parameters for σ^2 we have again $\alpha = r^B_{old} + 1$ and $\beta = s^B_{old} + 1$, so $\alpha, \beta \geq 1$ because $r^B_{old}, s^B_{old} \geq 0$. The highest value the variance can take in this case is $1/12$ at point $\alpha = \beta = 1$ (which will follow directly from Theorems 1 and 2). We normalize σ^2 accordingly and get u_{AB}:

$$u_{AB} \stackrel{def}{=} 12\sigma^2 \tag{7}$$

It remains to show that the variance has the properties demanded by Wang and Singh [12] – or rather the *inverse* properties, because we are addressing uncertainty (they addressed certainty). First, we show that for fixed *conflict* χ between α and β, σ^2 decreases. Conflict refers to the similarity of α and β: Many positive *and* negative experiences give reason for a higher uncertainty about an agents trustworthiness. Without loss of generality we assume $\alpha \leq \beta$ such that the conflict can be expressed as $\frac{\alpha}{\beta}$. Second, we show that for increasing conflict and increasing α, σ^2 decreases to $\frac{\alpha+\beta}{2}$.

Theorem 1. *For $\alpha \leq \beta$ and fixed conflict $\chi := \frac{\alpha}{\beta}$, σ^2 decreases for increasing $\alpha + \beta$.*

Proof. We have to show that for any $\chi > 0$ the first derivative of σ^2 is negative for all possible α, β. In (6) we substitute β by (α/χ) and differentiate with respect to α:

$$\frac{d\sigma^2}{d\alpha} = \frac{d}{d\alpha}\left(\frac{\alpha^2/\chi}{(\alpha/\chi+\alpha)^2(\alpha/\chi+\alpha+1)}\right) \tag{8}$$

$$= \cdots = -\frac{\chi^2}{(1+\chi)(\chi+\alpha+\chi\alpha)^2} < 0, \forall \beta \geq \alpha \geq 1. \tag{9}$$

□

Theorem 2. *Assuming fixed $\gamma := \alpha + \beta$. For increasing conflict, σ^2 increases.*

Proof. We have to look at the "slices" of σ^2 where $\gamma := \alpha + \beta$ is fixed. The function at the respective "slice" should increase when the conflict increases and decrease again when the conflict decreases. In σ^2 we substitute β by $(\gamma - \alpha)$:

$$g(\alpha) := \frac{\alpha(\gamma-\alpha)}{\gamma^2 * (\gamma+1))} \tag{10}$$

The maximum of $g(\alpha)$ is to be shown to be at the point where the conflict is maximal, i.e. $\alpha = \beta$ which is $\alpha = \gamma/2$. If additionally $g(\alpha)$ is concave down, i.e. the second derivation is negative for all possible α and γ, we're done. We find:

$$\frac{d^2g(\alpha)}{d\alpha^2} = \cdots = \frac{dg(\alpha)}{d\alpha}\left(\frac{\gamma-2\alpha}{\gamma^2+\gamma^3}\right) = -\frac{2}{\gamma^2+\gamma^3} \tag{11}$$

For $\gamma > 0$ the first derivation has its only root (and so its maximum) clearly at $\alpha = \gamma/2$. The second derivation is negative for all $\gamma > 0$. □

Depending on the scenario, it might be necessary to control the speed with which the uncertainty decreases; however this is done (e.g. taking the nth root of the variance), it has to be guaranteed that the properties of the variance described in Theorems 1 and 2 are still fulfilled.

3.3 Optimizing the Number of Challenges

In this section, we show how to find an optimal number of challenges n when given a number of real requests m. With "optimal" we refer to a number of challenges that minimizes the expected costs that arise when the error rate for the challenges and the one for the real requests differs. For example, an agent gets many correct answers to the verifiable challenges but not a single correct answer to the real requests. Then, the agent would underestimate the error rate for the real requests and work with incorrect information. Therefore, we find the number of challenges for which the expected difference between errors to the challenges and errors to the real requests is minimized. Note that a malicious information provider could try to answer all challenges correctly while giving wrong answers to all real requests. However, such situation could only be reached by *guessing* the number and the positions of the challenges – because real requests and challenges are randomly

merged. We proceed as follows. We first calculate the probabilities for all possible differences in the error rates. These probabilities are used to determine the expected costs. The resulting cost-function is minimized in respect to n.

First, let us virtually separate challenges from real requests and denote the vector that contains the answers to the challenges with $\vec{n}$ and accordingly the vector that contains the answers to the real requests with $\vec{m}$. Let $\vec{m}$ be of size m and $\vec{n}$ be of size n. Further let $w(\vec{x})$ be a function returning the error rate in some vector $\vec{x}$. What we want to know first is the probability of having an error rate j in $\vec{m}$ given an error rate i in $\vec{n}$:

$$P\left(w(\vec{m}) = j | w(\vec{n}) = i\right) \tag{12}$$

The error rates $w(\vec{m})$ and $w(\vec{n})$ seem to be statistically independent. That is however not the case because they both depend on the same error probability p_w, according to which the answers were answered incorrectly (see also Sect. 3.1). Therefore, we have to consider $P(\vec{m}, p_w|\vec{n})$ (for the moment ignore the function $w(\cdot)$). We use the basic product rule (see [14], p. 51) and get:

$$P(\vec{m}, p_w|\vec{n}) = P(\vec{m}|p_w, \vec{n})P(p_w|\vec{n}) \tag{13}$$

$$= P(\vec{m}|p_w)P(p_w|\vec{n}) \tag{14}$$

The last step leading to (14) is allowed because $\vec{m}$ is independent of $\vec{n}$ for given p_w. The probability $P(w(\vec{m}) = j|p_w)$ is the probability for k failures in m independent Bernoulli trials with error probability p_w, where $k = j * m$. So, we have a binomial distribution with parameters m and p_w; we will write $P_{p_w}(k|m)$. The probability $P(p_w|w(\vec{n}) = i)$ follows the beta distribution f with parameters $\alpha = i * n + 1$ and $\beta = (1 - i) * n + 1$ (as derived in Section 3.1). In order to get (12) we can integrate over all mutually exclusive p_w:

$$P(w(\vec{m}) = j | w(\vec{n}) = i) = \int_0^1 P_{p_w}(j * m|m) \tag{15}$$

$$* f(p_w; i * n + 1, (1 - i) * n + 1)dp_w \tag{16}$$

Now, we can calculate the probability that the error rate in $\vec{m}$ differs from the error rate in $\vec{n}$ by some x. This is done by marginalization over all $\tau \in T$, where T is the set that contains all possible error rates in $\vec{n}$, i.e. $T = \{\frac{a}{n} | a \in \mathbb{N}_0, a \leq n\}$:

$$P(w(\vec{m}) - w(\vec{n}) = x) = \sum_{\tau \in T} P(w(\vec{m}) = \tau + x, w(\vec{n}) = \tau) \tag{17}$$

$$= \sum_{\tau \in T} P(w(\vec{m}) = \tau + x | w(\vec{n}) = \tau)P(w(\vec{n}) = \tau) \tag{18}$$

Note that a priori all $\vec{n}$ are equiprobable and so we can compute $P(w(\vec{n}))$ in a combinatorial fashion; i.e. for $w(\vec{n}) = a/n$, we have to divide the number of possibilities to have a incorrect answers in $\vec{n}$, by the number of all possible vectors $\vec{n}$:

$$P(w(\vec{n}) = a/n) = \frac{\binom{n}{a}}{2^n} \tag{19}$$

Formula (18) will be calculated for all those differences x that are "possible", i.e. those contained in the set $\mathbb{X} := \{\frac{a}{m} - \frac{b}{n} | a, b \in \mathbb{N}_0, a \leq m, b \leq n\}$. Note that $\mathbb{X}$ contains both positive and negative x. For positive x, we have a higher error rate in $\vec{m}$, for negative x we have a higher error rate in $\vec{n}$.

We are now ready to define a cost-function that calculates the expected costs for a specific number of challenges n. Let the following parameters be given:

m – number of real requests,
c_c – costs for generating one challenge + costs for requesting the answer (the *information provider* may get some payment) + costs for evaluating the answer (which is a simple comparison to the already known answer),
c_d – costs for a difference between $w(\vec{n})$ and $w(\vec{m})$ of 1. So, a high c_d aims at a high accuracy in the estimation of $w(\vec{m})$.

Then, for a chosen number of challenges $n \geq 1$ the cost function is given by:

$$c(m, n, c_c, c_d) = n * c_c + \sum_{x \in \mathbb{X}} |x| * c_d * P(w(\vec{m}) - w(\vec{n}) = x) \tag{20}$$

This cost function (20) adds the costs for using n challenges, to the expected costs when using n challenges for m requests. To find an optimal n, the function (20) needs to be minimized in respect to n. As an optimization of this cost function at runtime would be a too costly computation, we propose to use precomputed values for various m, c_c and c_d. In (20) we can take c_d out of the sum and rewrite the cost function as

$$c(m, n, c_c, c_d) = n * c_c + c_d * d(m, n) \ , \tag{21}$$

where $d(m, n)$ stands for the remaining part in (20). This allows for computing $d(m, n)$ before actually running the mechanism and minimizing $c(\cdot)$ once c_c and c_d are determined.

Figure 1 shows $d(m, n)$ for specific m and n. Note that interestingly for fixed m and increasing n, the function $d(m, n)$ does not necessarily decrease. For example for $m = 7$, the expected difference is smaller for $n = 7$ than for $n = 8$ and smaller for $n = 14$ than for $n = 15$. The reason is that for $n = 7$, each error rate in $\vec{n}$ has an exact matching error rate in $\vec{m}$; but for $n = 8$, most error rates in $\vec{n}$ differ from all error rates in $\vec{m}$ what increases the expected difference of error rates. Apparently, this fact has an impact on $d(m, n)$ in all cases where n is a multiple of m or where n divides m (e.g. see $m = 10$ and $n = 5$).

3.4 Weighting the Number of Challenges with Trust

Up to now, we estimated the optimal number of challenges for a given number of real requests. For this calculation, we assumed that the information providing agent has not yet shown to be trustworthy. Looking at the case in which an agent would fully trust in an information provider, he would not need to use challenges any more. This justifies the consideration that the more trustworthy the opponent

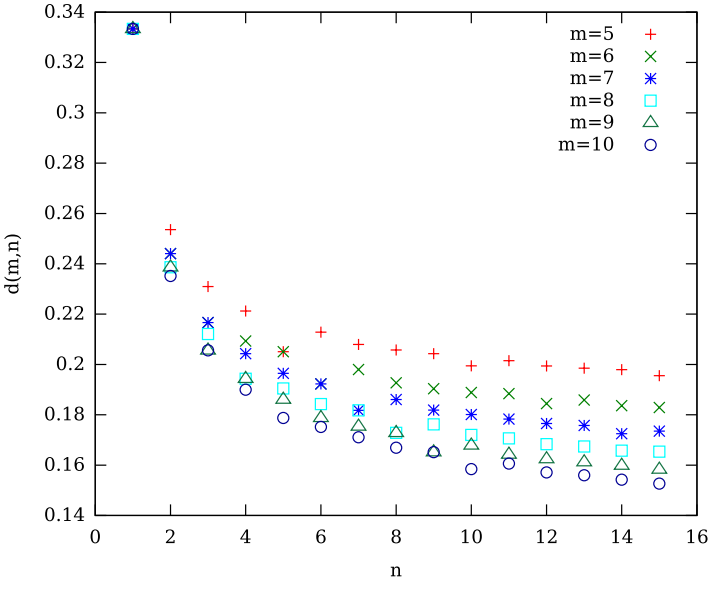

Fig. 1. Expected differences $d(m, n)$ between the error rates in $\vec{m}$ and $\vec{n}$

has shown to be, the smaller the number of challenges can be. In the following, we will integrate this notion of trust-based information acquisition into our model.

Basically, we want to weight the number of challenges that is optimal for the general case with the trust A has in the actual information provider B. As introduced in Section 3.2, A's trust in B is represented by a tuple $(t_{AB}, u_{AB}) \in (0, 1) \times (0, 1]$ with t_{AB} being the trust-value and u_{AB} the attached uncertainty (for simplicity we write t and u instead of t_{AB} and u_{AB} respectively). In this paper, we exclusively used probability theory to compute trust, as this can be formally justified. At this point however it seems appropriate to make use of our intuition and try to find the simplest formula that matches this intuition. The higher the trust-value t is, the smaller the number of challenges should be; the higher the attached uncertainty u is, the smaller the impact should be that the trust-value has on the number of challenges. This is met by $(1 - (1 - u)t)$, because for low trust-values or high uncertainty we take the full number of challenges ($\lim_{t \downarrow 0} 1 - t + tu = 1$ and we get 1 for $u = 1$), for low uncertainty the trust-value weights ($\lim_{u \downarrow 0} 1 - t + tu = 1 - t$) and for high trust-values the uncertainty weights ($\lim_{t \uparrow 1} 1 - t + tu = u$). So we get a number of challenges n that is weighted with trust by:

$$n = (1 - t + tu) \operatorname*{argmin}_{n'} c(m, n', c_c, c_d) \tag{22}$$

As desired, for initial trust $(t, u) = (0.5, 1)$ (see Sect. 3.2), the number of challenges is not reduced.

To account for changes in an agent's character or strategy and to prevent the unawareness towards the *first-time offender problem* [4], the number of challenges

should never become zero. For that purpose, we introduce a minimum number of challenges $min_n > 0$. Finally, an agent determines the number of challenges he will use by $\max(n, min_n)$.

4 Discussion

Several aspects for the use of the presented mechanism in practice have not yet been addressed and shall be discussed in this section.

Collusion. For the choice of challenges, two important rules have to be respected in order to avoid the possibility of *collusion*:

1. For requests that were not answered satisfactory and are therefore requested again from other agents, the same challenges are to be used.
2. For requests for differing information, different challenges are to be used.

The reader can easily verify that otherwise colluding agents would be able to identify real requests and challenges only by comparing the different request-vectors and checking what has changed.

Distributing questions over time. Depending on the setting, it might be impractical to send requests together with challenges bundled in a vector. If an agent wants to send only one question at a time, he can distribute challenges and real requests over time. In this case, he has to be careful to give an attacker no opportunity to deduce information from the points in time when questions are sent about whether a question is a challenge or a real request (similar attacks are called *timing attack* in cryptology).

Lack of resources. In specific cases where a lack of resources is the only reason for not being able to verify a whole response, real requests can be declared to be challenges after a response has been received. This has the advantage that challenges cannot be disclosed (there are no challenges beforehand) and do not cause additional costs. Then, an optimal number of challenges can be determined *during* verification by using statistical considerations. Analogous problems can be found in the area of *Statistical Quality Control* ([15]), where a fraction of the output of a system is selected randomly and tested.

Delayed verification. Assume the case where the verification of information is impossible because some knowledge is not given that would be needed for the verification. To give an example, let agent A have requested the circumference c of a circle given its diameter d (with $c = \pi * d$). Let us also assume that A did not know π at request-time and could not verify the response. When A obtains π at a later point in time, he can verify the response by hindsight. The same holds, if A did not verify the answer because he was too busy at the time but gets round to verify the data at a later point in time. In our mechanism, such delayed verifications can be easily integrated by just recalculating the trust in the information provider and rechecking the need for further requests.

Context-sensitivity. The *context-sensitivity* of trust is important in the field of information acquisition. Agents may be competent in some domains ("what is the prime factorization of 12345?") and incompetent in others ("will it rain today in Prague?"). In order to account for the assumption that an information provider can be described by an error-probability p_w (see Sect. 3.1), all questions in one request vector must belong to the same domain. Still, different request-vectors can belong to different domains. The approach by Rehák and Pechoucek [4] seems suitable to account for the latter issue. Alternatively, techniques such as *Latent Semantic Indexing (LSI)* [16], *PLSI* [17] or *Concept Indexing* [18] can be used. These techniques would allow for defining the context-space on the basis of acquired natural-language text.

Selection of information providers. How to decide whether a response-vector is accurate enough was not part of this work. However, the general idea is that an agent should discard acquired information and request it from other agents if he can expect that the improvement will be worth the investment. In case he decides to discard the information and request it again, he has to choose an information provider from which to request. Solutions to this problem, known as the *exploitation vs. exploration* problem, can be found in literature [19,20].

Efficiency. In the approach of redundant computation (see Sect. 1), the requesting agent needs to choose a positive integer $\rho > 1$ as redundancy level (usually $\rho = 2$ or $\rho = 3$ [3]). Opposed to this, in our approach, the requesting agent can adjust continuously the amount of resources used for verification. To give an example, if the ratio of challenges n to real requests m is chosen to be $2/4$, then this would amount to a redundancy level of $\rho = 0.5$ if that was possible.

However, the generation of challenges has also to be taken into account. As this generation is domain dependent, we will analyze an approach that is generally possible[1]: the results from redundant requests are used to fill a pool that contains the answers for future challenges; as soon as the pool contains n answers, these are used for a request-vector of size $m+n$ with m real requests and n challenges. A mechanism that only uses redundancy gives n results for $\rho * n$ requests, i.e. an efficiency of $\frac{1}{\rho}$. By additionally using challenges from the pool one gets overall $n + m$ results for $(\rho * n) + (m + n)$ requests, i.e. an efficiency of $\frac{n+m}{\rho*n+m+n}$. To illustrate the potential difference in efficiencies, we give the following example. In the approach without challenges, for $\rho = 3$ one gets an efficiency of $\frac{1}{3} \approx 0.333$. In the "hybrid" approach with redundancy and challenges, using $\rho = 3$, one can get an efficiency of $\frac{6}{15} = 0.4$ (for $m = n = 3$) or $\frac{8}{14} \approx 0.571$ (for $m = 6, n = 2$). The more an agent trusts a provider, the less challenges he can use and so the higher the efficiency will be.

[1] This approach would mainly improve the use of redundant requests in terms of efficiency, but still would be partially sensitive to collusion.

5 Related Work

Several trust and reputation models base their computations on a beta distribution [13,12,21,22]. However, they all assume that the assessing agent is able to verify what we call the "real requests". Our mechanism addresses the cases in which this verification is undesirable or not possible. These cases are numerous which was illustrated in Section 2.

Fullam et al. [23] propose a method for information acquisition from possibly malicious or incompetent sources. Also, they show how to manage the trade-off between costs for information acquisition, quality of the acquired information and the coverage of an agent's goals. In their model, the reliability of an information provider is assessed by checking whether some acquired information fits the agent's beliefs. In contrast to our mechanism, their approach does not allow to handle acquired information that is not related to an agent's beliefs (e.g. mathematical calculations, music data, etc.).

Liau[24] models the relationship among *belief*, *trust* and *information acquisition* by use of modal logics. They formally study the role that trust can play when uncertain information is assimilated in an agent's beliefs.

In network security, a family of authentication protocols uses the principle of *challenge-response* (e.g. [25,26]). Here, an entity proves its identity by answering to a challenge posed by the opponent – this challenge can only be answered, if the entity is in possession of a certain secret, and this secret is only given to the entity with the identity in claim. However, our mechanism is not related to this class of protocols: We do not use secrets and moreover, in our case it is not about authentication.

6 Conclusion and Future Work

In this paper, a trust-based mechanism was presented which uses challenges to estimate the correctness of acquired information that cannot be verified. We showed how to choose an optimal number of challenges for a given number of real requests and found that it is advantageous if the number of challenges is a multiple of the number of real requests, or divides it. A way to reduce this optimal number of challenges was proposed that makes use of *trust*. For these purposes, a formal trust-model was introduced that computes trust-values and uncertainty based on the beta distribution. It was proven that our uncertainty measure preserves the properties demanded by Wang and Singh [12] but is easier to compute.

Currently, we are working on a bootstrapping procedure for the mechanism. Besides, we develop a procedure for deciding whether some acquired information is accurate enough or should be requested again from other agents. Finally, the mechanism is planned to be implemented and tested.

Acknowledgments. We would like to thank Ulrich Sorger for many valuable discussions and Uwe Roth and Daniel Fischer for their constructive feedbacks.

References

1. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24(1), 33–60 (2005)
2. Ramchurn, S.D., Huynh, T.D., Jennings, N.R.: Trust in multi-agent systems. Knowl. Eng. Rev. 19(1), 1–25 (2004)
3. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: An experiment in public-resource computing. Commun. ACM 45(11), 56–61 (2002)
4. Rehák, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 298–312. Springer, Heidelberg (2007)
5. Berman, F., Fox, G., Hey, A.J.G.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Inc., New York (2003)
6. Weiss, A.: Computing in the clouds. netWorker 11(4), 16–25 (2007)
7. Toh, C.K.: Ad Hoc Wireless Networks: Protocols and Systems. Prentice Hall PTR, Upper Saddle River (2001)
8. Christin, N., Weigend, A.S., Chuang, J.: Content availability, pollution and poisoning in file sharing peer-to-peer networks. In: EC 2005: Proc. of the 6th ACM Conf. on Electronic commerce, pp. 68–77. ACM Press, New York (2005)
9. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: HICSS 2000: Proc. of the 33rd Hawaii Int. Conf. on System Sciences. IEEE Computer Society Press, Los Alamitos (2000)
10. Rohatgi, V.K.: Statistical Inference. Dover Publications, Incorporated, Mineola (2003)
11. Kinateder, M., Baschny, E., Rothermel, K.: Towards a generic trust model - comparison of various trust update algorithms. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) Trust 2005. LNCS, vol. 3477, pp. 177–192. Springer, Heidelberg (2005)
12. Wang, Y., Singh, M.P.: Formal trust model for multiagent systems. In: IJCAI 2007: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, pp. 1551–1556 (2007)
13. Teacy, W.T.L., Patel, J., Jennings, N.R., Luck, M.: Travos: Trust and reputation in the context of inaccurate information sources. Auton. Agents Multi-Agent Syst. 12(2), 183–198 (2006)
14. Jaynes, E.T.: Probability Theory: The Logic of Science. Cambridge University Press, Cambridge (2003)
15. Montgomery, D.C.: Introduction to Statistical Quality Control, 5th edn. John Wiley, Chichester (2004)
16. Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S., Harshman, R.: Using latent semantic analysis to improve access to textual information. In: CHI 1988: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, pp. 281–285. ACM, New York (1988)
17. Hofmann, T.: Probabilistic latent semantic indexing. In: SIGIR 1999: Proc. of the 22nd annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 50–57. ACM, New York (1999)
18. Karypis, G., Han, E.: Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical Report TR-00-0016, University of Minnesota (2000)
19. Dearden, R., Friedman, N., Andre, D.: Model based bayesian exploration. In: UAI 1999: Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence, pp. 150–159 (1999)

20. Chalkiadakis, G., Boutilier, C.: Coordination in multiagent reinforcement learning: a bayesian approach. In: AAMAS 2003: Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 709–716. ACM Press, New York (2003)
21. Buchegger, S., Boudec, J.Y.L.: A robust reputation system for mobile ad hoc networks. Technical Report IC/2003/50, EPFL-IC-LCA, CH-1015 Lausanne (July 2003)
22. Jøsang, A., Ismail, R.: The beta reputation system. In: Proc. of the 15th Bled Conf. on Electronic Commerce, pp. 324–337 (2002)
23. Fullam, K.K., Park, J., Barber, K.S.: Trust-driven information acquisition for secure and quality decision-making. In: KIMAS 2005: Proc. of Int. Conf. on Integration of Knowledge Intensive Multi-Agent Systems, pp. 303–310 (2005)
24. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems: a modal logic formulation. Artif. Intell. 149(1), 31–60 (2003)
25. Otway, D., Rees, O.: Efficient and timely mutual authentication. SIGOPS Oper. Syst. Rev. 21(1), 8–10 (1987)
26. Steiner, J.G., Neuman, C., Schiller, J.I.: Kerberos: An authentication service for open network systems. In: Proc. of the Winter 1988 Usenix Conference, pp. 191–2024 (1988)