A DISTRIBUTED REPUTATION MANAGEMENT SCHEME FOR MOBILE AGENT-BASED APPLICATIONS

Omaima Bamasak
School of Computer Science
University of Manchester, UK
obamasak@cs.man.ac.uk

Ning Zhang
School of Computer Science
University of Manchester, UK
nzhang@cs.man.ac.uk

ABSTRACT

This paper proposes a new distributed reputation management scheme to support agent-based applications. The scheme uses a transaction feedback system and five metrics to real-time evaluate the reputation of Trusted Third Party (TTP) hosts that are selected and employed to support an agent in performing its transactional tasks. The scheme exhibits two interesting features not seen in previous works: firstly, it offers better efficiency and robustness, and secondly, it credits/penalizes a TTP-host according to its transactional response, the transaction value and the reputation of the source of feedback. In addition to facilitating TTP-host selection based upon its reputation, this mechanism of credit and penalisation is expected to deter dishonesty or misbehaviour by the entities involved.

Keywords: Distributed reputation management, e-commerce security, mobile agent security.

1. Introduction

Mobile agents offer a new computing paradigm that allows us to gain a greater access to resources on the Internet in a more efficient and automatic manner. By delegating tasks to mobile agents, users can accomplish extended or complicated tasks that they, otherwise, would rather not or cannot perform themselves. Moreover, by sending mobile agents close to information sources, we can reduce network traffic and application latency. However, the success in using the mobile agent paradigm depends on the level of security it offers. Without adequate security provision, security-sensitive tasks cannot be delegated to an agent that may execute the tasks in a remote and possibly malicious host.

To illustrate the security risks faced by a mobile agent in the Internet environment, we here analyse an e-Commerce application scenario. A businessman is to search for an airline ticket for a business trip on the Web. As the search may take a little while and he is too busy to do the search in person himself, he has decided to delegate this task to a mobile agent. The businessman specifies some requirements for this purchase. For example, he may specify that the purchase should only proceed if the price of a ticket is no more than £200, and if such an offer is found, then the agent should book the flight at once on his behalf. It is clear that, in such a scenario, a remote airline host has the incentive to misbehave or to attack the agent. For example, the host may force the agent to sign a deal that is above the specified price threshold, or the host may forge the agent signature(s) altogether by spying on the signature key carried by the agent so that the businessman will be liable for the deal(s).

Previous research works [Kim et al. 2001, Kotzanikolaou et al. 2000, Lee et al. 2001, Mambo et al. 1996a, Mambo et al. 1996b, Reagle 1998] have proposed some solutions to address this problem, or, in a more generic term, the problem of securing the signature key (also called *proxy key*) carried by a mobile agent. However, there are rooms for improvements in these solutions. Some of these solutions do not support the security service of non-repudiation of signature receipt [Kotzanikolaou et al. 2000, Mambo et al. 1996a, Mambo et al. 1996b, Reagle 1998], while others do not provide sufficient protection for the proxy key against misuse by a remote malicious (merchant) host [Lee 2001]. In addition, some of the solutions that provide a good level of security protection, e.g. the Kim's protocol [Kim et al. 2001], are inefficient in terms of computational costs.

Recently, Bamasak and Zhang [Obamasak & Zhang 2004] have proposed a more secure and efficient scheme for signature delegation to a mobile agent aimed at addressing the weaknesses mentioned above. This scheme makes use of a Trusted Third Party (TTP) to assist the mobile agent in signature generation and dispute resolution to support the non-repudiation of signature receipt. The scheme assumes that a single host plays the role of this TTP. The downside of this centralized TTP based approach is that it is a potential performance/reliability bottleneck introducing a single point of failure for protocol execution. The fact that it is increasingly difficult to protect any single system against the sort of attacks proliferating on the Internet today has made this assumption a weakness in our design. One way of overcoming this weakness is to distribute the role of the TTP among a set of trusted hosts rather than relying on a single host. In this way, the robustness of the protocol is strengthened as a majority of, or all of, the trusted hosts would have to be compromised before the system as a whole is compromised.

An important question raised at this stage is: how can the agent owner decide on the group of hosts that can be trusted to jointly perform the role of the TTP? In other words, what should be the selection criteria on which the agent owner makes the decision as whether or not a host should be selected to play the role of the TTP? Obviously, the agent owner should select the *TTP-hosts* that have an acceptable level of reputation. Conventional security solutions and cryptographic methods, such as the use of passwords and digital certificates, alone are not sufficient for us to evaluate the trustworthiness of a *TTP-host*. They can help us to establish whether a party is authenticated and authorized to take certain actions. They can not guarantee that a party (even authorised) will perform as promised and deliver a trusted service.

In this paper, we investigate current solutions to reputation management, and present the design of our distributed reputation scheme suited to the signature delegation model proposed in [Obamasak & Zhang 2004]. The remaining part of this paper is organized as follows. Section 2 presents an analysis of related work in reputation management. Section 3 defines the metrics for evaluating the reputation of a *TTP-host* in e-commerce context. Section 4 describes our Distributed Reputation Management scheme along with the two algorithms used by the scheme. Finally, Section 5 draws the conclusions of this paper.

2. Related Work

Reputation is defined as "the amount of trust inspired by a particular person in a specific setting or domain of interest" by Marsh [Marsh 1994]. In [Reagle 1996], reputation is defined as "asset creation and it is evaluated according to its expected economic returns". Recently, a number of trust/reputation systems and mechanisms have been proposed for on-line trading and agent systems [Ketchpel & Garcia-Molina 1996, Zacharia & Maes 2000, Xiong & Liu 2003, Bryant & Colledge 2002]. The approach used in [Ketchpel & Garcia-Molina 1996] employs one or more TTP-hosts to achieve a fair exchange security service. This work, however, assumes that the *TTP-hosts* are trustworthy, and never misbehave. The authors in [Zacharia & Maes 2000] have developed a collaborative reputation mechanism allowing personalized evaluation of ratings assigned to participating entities. Based upon these ratings, the reliabilities of the entities are estimated. However, the work does not clearly state the evaluation metrics, or on what criteria, the ratings (i.e. reputation values) are calculated. The methods presented in [Malaga 2001, Xiong & Liu 2003, Jøsang & Ismail 2002] only deal with peer-to-peer trust. They do not provide a solution for the selection of a group of *TTP-hosts* that can jointly performs a security-sensitive task.

In the Internet-based electronic marketplace, there are a few working on-line reputation systems. Examples include the eBay [eBay 2004] and Yahoo! Auction [Yahoo 2004] feedback systems. These systems allow participants of a transaction to rate each other with a value of +1 for a positive feedback, 0 for being neutral, and -1 for a negative feedback. Only winning bidders and sellers may submit a feedback for their completed transactions. A rating from an eBay participant only contributes once to another participant's rating. For example, if participant *A* gives 3 positive points to another participant *B* (for 3 different transactions), participant *B*'s rating can only increase by +1. However, if *A* gives two negative and one positive points to *B*, the negative rating will count once and so will the positive rating. A transaction participant is also allowed to submit a written response. The reputation value of a participant is then calculated as the sum of all points this participant has received since s/he started dealing in the marketplace. Any participant with a reputation value of -4 will be suspended from further dealings on eBay. This approach is linear and single-factor based, i.e. positive or negative feedback, and often fails to capture the behavior of the parties involved effectively. For example, a participant who has 100 positive feedback points will have the same rating as a user who has had 300 positive and 200 negative points.

The Distributed Reputation Management scheme to be proposed in this paper has addressed these weaknesses. Namely, the scheme has defined clear criteria for reputation calculation. It is also embedded with an algorithm, the TTP-hosts Subset Selection (TSS) algorithm, that selects one or more (or a subset of) *TTP-hosts* based upon the criteria such that the aggregated reputation value of these selected TTP-hosts satisfies the trust level specified by an agent owner. The trust level is, in turn, determined by the value of the transaction to be performed. In addition, the scheme has also got an algorithm, the Trust and Reliability Updating (TRU) algorithm, by which the agent owner calculates and updates reputation values associated with each of the *TTP-hosts* involved in a transaction. The algorithm uses a non-linear approach to transaction feedback sent by merchant hosts. Figure 1 shows an overview of the Distributed Reputation Management scheme.

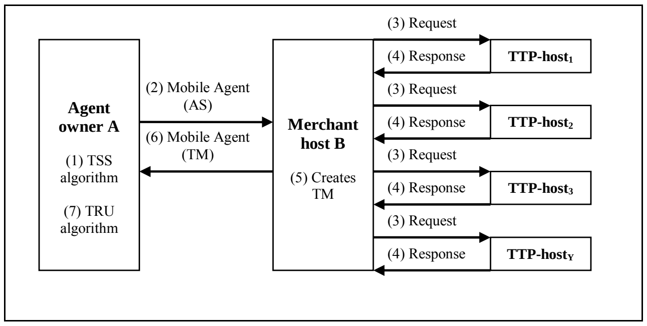

Figure 1. An Overview of the Distributed Reputation Management Scheme

3. Reputation Metrics

In our distributed trust model, a *TTP-host*'s reputation is measured by two values, a *trust level* and a *reliability level*. Both values are the results of the *TTP-host*'s aggregated transactional behaviours (reflected by its responses) over a specific past period. The trust level reflects the truthfulness of the *TTP-host* in performing the transactions and the reliability level reflects its availability in providing the TTP service. Both values are the function of the following parameters.

(1) *Transaction outcome feedback*. Upon the execution of each transaction, the remote (merchant) host assigns a feedback measured in terms of trust and reliability values to each participating *TTP-host*. The values assigned are related to the message reply sent by the *TTP-host* in relation to this transaction. For example, if the reply can pass its verification process positively, the trust value will be 'Yes'. Otherwise, if the message reply fails the verification process, perhaps due to a malicious intent by the TTP-host or due to channel errors (repeated transmissions are applied), then the trust value will be 'No'. For the reliability value, if the merchant host did not receive an expected message from the *TTP-host* after a certain period of time, the merchant will assign 'No' to the reliability value associated with the *TTP-host*. Otherwise, if the expected message is received, the reliability value will be 'Yes'. The per-transaction trust and reliability values reflect how well and reliable this *TTP-host* has fulfilled its part of the protocol execution in performing this transaction. The overall trust and reliability values of a *TTP-host* are the aggregation of the per-transaction trust and reliability values given to the host in all the transactions involved by the TTP-host over a specified past time period T_h.

(2) *Total number of transactions performed*. A simple aggregation of feedbacks may fail to capture the true record of a *TTP-host*'s transactional behaviour. For example, a *TTP-host* that has performed dozens of transactions but cheated on 1 out of every 4 occasions will have a higher aggregated reputation value in comparison with a *TTP-host* that has only performed 10 transactions and has been faithful in all of these occasions. In other words, the total number of transactions that the *TTP-host* has performed over the specific past period is also an important indicator of its reputation and should be taken into account for the calculation of its reputation value. In our model, the average feedback value, measured as the ratio of the sum of the feedbacks the *TTP-host* has received over time period T_h to the total number of transactions the *TTP-host* has taken part over the same period, is used instead of a simple sum.

(3) *Transaction value*. The value of a transaction undertaken by a *TTP-host* is another important metric for its reputation evaluation [Kim & Benbasat 2003]. Helping with a transaction with a value of £1000 certainly worth more credits than that with a value of £10. Similarly, failure to perform a transaction of £1000 should be penalised more than failing a £10 one. The solution used by e-Bay fail to address this observation, which may let away a *TTP-host* that develop a sound reputation value by being honest in performing small value transactions, but behaving maliciously with large value ones. Our reputation evaluation algorithm has overcome this weakness by having an embedded risk factor in the reputation calculation. The risk factor is proportional to the transaction value, and is used to weigh the feedback of the transaction given by a merchant to a *TTP-host*.

(4) *Total number of failed incidents*. To further enhance fairness in assessing transactional behaviour of a *TTP-host*, we have also introduced a counter to record the total number of failed or incorrect responses made by the TTP-host within a specified period. A *TTP-host* with this counter value reaching a certain threshold will have its reputation value reduced to the minimum. In this case, the *TTP-host* will have to perform a considerable volume of honest transactions in order to rebuild its reputation.

(5) *Source of the feedback*. As the author in [Malaga 2001] stated: "when considering reputation information, we often account for the context and the source of the information", the feedback from a party (i.e. a merchant) who has a better reputation should be weighed more in calculating reputation.

4. A Distributed Reputation Management Scheme

To dynamically select a subset of TTP-hosts among a set of N trusted hosts, $\{TTP\text{-}host_i, i \in \{1, ..., N\}\}$, based upon their real-time transactional behaviour and reliability to assist a mobile agent to perform security sensitive tasks, two algorithms are required. The first, *called TTP-hosts Subgroup Selection (TSS) algorithm*, allows the agent owner to select a subset of Y ($<N$) most trustworthy *TTP-hosts* from N available ones. The second algorithm, called *Trust and Reliability Updating (TRU) algorithm*, allows the agent owner to evaluate and assign trust and reliability values to each *TTP-host* that has taken part in a transaction based upon the feedback received from his/her merchant host. The two algorithms constitute our distributed reputation management scheme.

4.1 Assumptions

The Distributed Reputation Management Scheme is designed based upon the following assumptions:

- The agent owner maintains a table TA (Trust Assessment) containing trust and reliability values associated with each of the *TTP-hosts* that the agent owner has dealt with in the past period T_h. An example TA is given in Table 1. In the table, each row corresponds to one *TTP-host* containing eight attributes: {TTP$_i$-ID, Trust$_i$, Rel$_i$, Sat$_i$, TotalTran$_i$, T-C$_i$, R-C$_i$, Fee}.

  The TTP$_i$-ID is the unique identifier of host *TTP-host$_i$*, e.g. its distinguished name[1] [ITU-T 1997]. Trust$_i$ and Rel$_i$ are its aggregated trust and reliability values, respectively. The Trust$_i$ attribute indicates the level of *TTP-host$_i$*'s trustworthiness (or honesty) in performing its job.

  The Rel$_i$ attribute indicates its reliability level in service provision. It is assumed that the initial values of Trust$_i$ and Rel$_i$ are set to zeros indicating that the agent owner has not yet had any experience in dealing with the *TTP-host*. These initial values are greater than the values indicating a malicious *TTP-host* ($\leq$ -1). In this way, a newly deployed *TTP-host* will not be treated unfairly.

  Sat$_i$ refers to the average value of Trust$_i$ and Rel$_i$, i.e. $Sat_i = \Omega \times Trust_i + \lambda \times Rel_i$, where $\Omega + \lambda = 1$. The parameters, Ω and λ, represent the impacts of Trust$_i$ and Rel$_i$ in calculating the value of Sat$_i$, respectively. The choice of values given to these parameters is of the agent owner's preferences. For example, if an agent owner feels that Trust$_i$ should weigh more than Rel$_i$, then he may assign 0.7 for Ω and 0.3 for λ (these values are used in the example given in table 1). The higher the value of Sat$_i$, the more confidence the agent owner has in the *TTP-host$_i$*.

  TotalTran$_i$ refers to the total number of transactions taken part by the *TTP-host$_i$* with the agent owner during the past period T_h. T-C$_i$ is the total number of transactional responses sent by *TTP-host$_i$*, which have failed to pass the integrity verification. R-C$_i$ is the total number of occasions when *TTP-host$_I$* fails to respond during the period.

  Fee$_i$ specifies the amount of money *TTP-host$_i$* charges for providing the TTP service [Wang et al. 2005]. This attribute may influence an agent owner's decision as whether or not a particular *TTP-host* should be chosen to join the AS (Active Subgroup) list. We also assume that table TA is sorted in a descending order according to the satisfaction (Sat) values. Thus, the *TTP-host* with the highest satisfaction value shall be in the first row of the table. The agent owner may decide an upper-limit for the trust and reliability values according to his/her preferences.

Table 1. An example of a single row in table TA

TTP$_i$ -ID	Trust$_i$	Rel$_i$	Sat$_i$	TotalTran$_i$	T-C$_i$	R-C$_i$	Fee$_i$
1.www.verisign.com 2.VeriSign Limited 3.IT Department 4.South Melbourne 5.Victoria 6.AU	2	3	5.1	10	0	1	100

- The agent owner also maintains a table MR (Merchant Reputation) containing reputation values associated with each of the merchants that the agent owner has dealt with in the past time period T_m. As shown in table 2, each row of the MR table corresponds to one merchant containing two attributes: {Merchant$_i$-ID, Rep$_i$}. Merchant$_i$-ID

[1] The distinguished name specified in [3] consists mainly of six fields: Common name (CN), Organisation Name (O), Organisation Unit (OU), Locality (L), State (S), and Country (C).

is the merchant's unique identifier. Similarly, the identifier can be the distinguished name of the merchant. Rep_i specifies the level of reputation $Merchant_i$ has accumulated from its previous dealings with the agent owner during period T_m. The initial value assigned to Rep_i is zero (neutral), which indicates that the agent owner has no experience in dealing with the $Merchant_i$ yet. The value of Rep_i is updated by the agent owner as follows: +1 is added if the transaction outcome is positive, 0 if no response is received from the merchant, and -1 if the transaction outcome is negative.

Table MR, maintained by the agent owner, represents only the agent owner's opinion on the merchants he has dealt with. Alternatively, table MR may be stored at a publicly accessible server, in which case the value Rep_i can be modified by multiple authorized agent owners or customers. In the latter case, Rep_i will represent the accumulated opinion about $Merchant_i$ among the community. The choice of the location of the MR table, i.e. at the agent owner side or in a public server, is left to the users' preferences. In our scheme, we adopt the former approach where the agent owner maintains its own table MR.

Table 2. An example of a table MR

$Merchant_i$ ID	**Rep_i**
1. www.dixons.co.uk 2. DSG Retail Limited 3. Sales Department 4. Hemel Hempstead 5. Hertfordshire 6. UK	2

- The merchant, once agreed on a deal with the mobile agent, creates a table, TM, containing the trust and reliability values for all the participating *TTP-hosts*. Each entry in the table, corresponding to a *TTP-host*, consists of three attributes: the first refers to the *TTP-host*'s ID; the second is its trust value; and the third is its reliability value. The trust attribute will be assigned with one of the following values (Yes, No, Unknown) and the reliability attribute will be assigned with either 'Yes' or 'No', depending on the outcome of the transaction involved with the *TTP-host*. Table 3 gives an example of value settings for table TM. Table 4 summarizes exemplar scenarios for the transaction outcomes and the trust and reliability values associated to these outcomes.

Table 3. An example of table TM

	Trust	**Rel**
TTP_1-ID	No	Yes
TTP_2-ID	Unknown	No
….		
TTP_N-ID	Yes	Yes

Table 4. The value setting for Trust and Reliability in various scenarios

Metric	**Values**	**Scenarios**
$Trust_i$	Yes	*TTP-host$_i$* sends a valid expected data to the merchant host.
	No	*TTP-host$_i$* sends an invalid expected data, or an unexpected data, or simply "a token that cannot pass a specified verification", to the merchant host.
	Unknown	The merchant has not received a response from the *TTP-host$_i$* during the protocol run. Therefore, he cannot make judgment whether this *TTP-host$_i$* is trustworthy or not.
Rel_i	Yes	The merchant host has received a response, either positive or negative, from *TTP-host$_i$*.
	No	The merchant host has not received any response from *TTP-host$_i$* during the protocol run even if repeated requests have been made. This may be due to that the *TTP-host$_i$* is out of service or the communication link between the merchant host and the *TTP-host$_i$* is broken down, etc.

Once the values in the table are set, the merchant then passes it to the agent owner, via the mobile agent, for him to update table TA accordingly.

- Tables TA and MR are controlled by the validity periods T_h and T_m, respectively, to maintain the freshness of the relevant data and to reduce memory and computational expenses.

4.2. The TTP-hosts Subset Selection (TSS) Algorithm

When an agent owner is ready to delegate a signature-signing task to his agent, the first thing s/he needs to do is to decide a subgroup of *TTP-hosts* that will take part in performing the task collectively and jointly with the agent. This subgroup is called the active subgroup (AS). To select the AS, the agent owner specifies a reputation threshold *Thr1* that is proportional to the transaction value to be undertaken. Table 5 shows exemplar settings of *Thr1* in relation to transaction values.

Table 5. Exemplar settings of *Thr1*

Transaction value	**Thr1**
< £10	1
£10- £50	2
£50 - £100	3
£100 - £200	4
…..	..
£2000 - £2100	20

From the table, it can be seen that the higher the transaction value the higher the threshold *Thr1* should be. The TSS algorithm then takes *Thr1* and table TA as its parameters to generate one or more (*N*>a subset >*1*)) *TTP-hosts* such that their aggregated Satisfaction value, computed using equation (1), is equal to or greater than *Thr1*. That is,

$$Agg = \sum_{i=1}^{j} Sat_i \qquad (1)$$

where, *j* is the number of *TTP-hosts* in the AS list. The selection process uses a top-down approach choosing the most trustworthy host first. This decision is made intuitively and other orders of selection are possible depending on customers' preferences. If the first *TTP-host* chosen has the satisfaction level that matches with the threshold corresponding to the transaction value, then this single *TTP-host* is sufficient. Otherwise, more *TTP-hosts* are chosen until their aggregated Satisfaction value reaches or exceeds *Thr1*.

The pseudo code for the TSS algorithm is given below.

TTP-hosts Subset Selection (TSS) Algorithm

```
Input:  table TA, Thr1
Output: AS member(s)
Method:
Initialize Agg to 0
For each row i in TA do
    Compute Agg = Agg + $Sat_i$ /* the Satisfaction value for $TTP\text{-}host_i$ is added to the aggregated
    average Agg */
    Insert $TTP_i$-ID in AS list
    Increment $TotalTran_i$
    If Agg equals to or greater than Thr1 then
        Exit the loop
    Else
        Increment i and start the next loop iteration
```

As the members of the AS list are chosen as the most trusted and reliable among all *N TTP-hosts*, they are most likely to perform the transaction securely and reliably. In addition, according to the algorithm, the higher the transaction value, the higher the threshold Thr_1 will be. As a result, the more *TTP-hosts* will be selected to execute the transaction. It is therefore more difficult for any single *TTP-host* to successfully forge a proxy signature or to manipulate the transactional process. In other words, the transaction outcome will be more likely to come to a satisfactory conclusion.

4.3. The Trust and Reliability Updating (TRU) Algorithm

The TRU algorithm is used by an agent owner to update trust and reliability values for each *TTP-host* upon the completion of each transaction. A merchant, once agreed on a deal with the mobile agent, creates a table TM and fills it with the trust and reliability values of all the participating *TTP-hosts* depending on the transaction outcome, as described in Section 4.1. The merchant then passes table TM to the mobile agent.

The agent then comes back from the shopping trip with the TM table and submits it to the agent owner. The agent owner refreshes the contents of table TA according to the values received in table TM by executing the TRU algorithm that takes tables TM, TA and the merchant's ID (Merchant-ID) as its input parameters. The algorithm performs a search to find a *TTP-host* in table TA until a matching *TTP-host* is found. Once found, the algorithm updates the Trust and Reliability values associated with the *TTP-host* in TA according to the values in TM and the reputation value Rep_k of the merchant Merchant_k-ID (extracted from table MR) using the equations given in table 6.

Table 6. Equations for updating Trust and Reliability value in TA

	Values in TM	**Updates in table TA**
Trust_i	Yes	$NewTrust_i = OldTrust_i + ((\partial \times Rep_k)/TotalTran_i)$
	No	$NewTrust_i = OldTrust_i - ((\partial \times Rep_k)/TotalTran_i)$
	Unknown	$NewTrust_i = OldTrust_i$
Rel_i	Yes	$NewRel_i = OldRel_i + ((\partial \times Rep_k)/TotalTran_i)$
	No	$NewRel_i = OldRel_i - ((\partial \times Rep_k)/TotalTran_i)$

NewTrust_i is the latest value assigned to Trust_i in table TA. OldTrust_i refers to the aggregated Trust_i value from all previous transactions. In other words, it is the value maintained in table TA before this transaction takes place. The same interpretation is applicable to NewRel_i and OldRel_i. If the total number of untrustworthy or unreliable transactions, as indicated by T-C_i or R-C_i, reaches a certain threshold Thr_2, then the maximum penalty γ, e.g. $\gamma = -5$, is applied to the values of NewTrust_i or NewRel_i, accordingly. ∂ indicates a risk factor specified by the agent owner and it is in proportional to the transaction value. The pseudo code for the TRU algorithm is given as follows.

```
**Trust and Reliability Updating (TRU) algorithm**
**Input**: table TM, table TA, $\partial$, $Thr_2$, $\gamma$, Merchant-ID
**Output**: updated table TA
**Method:**
Initialize k and i to 0
For each row k in MR
  Search for $Merchant_k$-ID that matches Merchant-ID
  If Found then
  /* Fetch the reputation value associated with $Merchant_k$-ID */
    Rep = $Rep_k$
    Exit the loop
Increment k and start the next loop iteration
For each row i in TA
  Search for $TTP_i$-ID that matches $TTP_i$-ID in TM
  If Found then
  /* Update the $Trust_i$ value in table TA */
  If $Trust_i$_TM is Yes then
      $NewTrust_i$_TA = $OldTrust_i$_TA + (($\partial$ × Rep)/$TotalTran_i$)If $Trust_i$_TM is No then
      Increment $T-C_i$
  If $T-C_i$ = $Thr_2$ then
      $Trust_i$_TA = $\gamma$
  Else $NewTrust_i$_TA = $OldTrust_i$_TA - (($\partial$ × Rep)/$TotalTran_i$)
  If $Trust_i$_TM is Unknown
      $NewTrust_i$ = $OldTrust_i$
  /* Update the Reliability value in table TA */
  If $Rel_i$_TM is Yes then
      $NewRel_i$_TA = $OldRel_i$_TA + (($\partial$ × Rep)/$TotalTran_i$)If $Trust_i$_TM is No then
      Increment $R-C_i$
```

```
If R-C_i = Thr_2 then
    Rel_i_TA = $\gamma$
Else
    NewRel_i_TA = OldRel_i_TA - (($\partial$ × Rep)/TotalTran_i)
```

An important feature of this algorithm is that the step value of award/penalty for Trust and Reliability is not linear. It gets smaller as the number of transactions performed by the *TTP-host* gets larger. In addition, the step value is also linked to the transaction value through risk factor ∂. The larger the transaction value, the higher the risk the agent owner has to endure, and the more reward/penalty the *TTP-host* will get shall the transaction succeed/fail. If a *TTP-host* repeatedly misbehaves or are unreliable, say, for Thr_2 times, its Trust or Reliability values will drop to γ. Furthermore, the step value is weighed according to the reputation (Rep_k) of the source of the feedback ($Merchant_k$). A feedback from a highly reputable merchant weighs more than that from a less reputable one. We believe that these features have captured users' expectation and acceptance in real life. The following example illustrates the implications of these features. When a customer has performed many valid deals with a specific merchant, the customer will be less likely to move away from this merchant if only a couple of deals with small values (small ∂) have ended up in an unsatisfactory manner. However, having a transaction with a large value gone wrong may completely put the customer off. This is why the effect of a transaction outcome on the overall Trust or Reliability value and subsequently the satisfaction (i.e. the reputation) of a *TTP-host* should be dependent on the three factors: the value of a transaction; the total number of transactions that have been performed in the past period concerned; and the reputation of the merchant host.

The two algorithms mentioned above jointly facilitate this non-linear reputation model. According to our best knowledge, there has not been any such non-linear reputation model proposed in the literature.

4.4 A working example

In this section, we use a working example to illustrate the use of the TSS and TRU algorithms. The example will show the state of the TA table before and after performing a particular transaction by the agent owner.

Let us assume the agent owner has been dealing with four *TTP-hosts* in past time period *T*. The parameters Ω and λ are each assigned with the value of 0.5. Table 7 shows the information corresponding to each of the four *TTP-hosts* in table TA.

Table 7. table TA prior to performing the transaction

TTP-ID	Trust	Rel	Sat	TotalTran	T-C	R-C
TTP_1	8	12	10	5	0	1
TTP_2	6	10	8	10	4	0
TTP_3	5	5	5	7	0	0
TTP_4	3	3	3	2	1	1

The agent owner now wants his agent to perform a transaction worth of £150 on his behalf, so the value of Thr_1 is set to 4. The TSS algorithm in this case will generate an AS list, containing only TTP_1 as this *TTP-host*'s satisfaction value (Sat) alone is greater than Thr_1. (It is worth noting that, if the transaction value is £2000, then Thr_1 would be 20, and the outcome of executing TSS algorithm will be an AS list containing the first three *TTP-hosts* in table TA as their aggregated Satisfaction value (23) is greater than Thr_1). The agent owner will set the values of the required parameters as follows: Thr_2 = 5, ∂ = 4, and γ = -5. Once the transaction is performed, the agent comes back with a TM table (given by $Merchant_i$ that the agent has performed the transaction with). Table 8 shows the contents of table TM.

Table 8. Table TM received from $Merchant_i$

TTP-ID	Trust	Rel
TTP_1	Yes	Yes
TTP_2	No	Yes
TTP_3	Unknown	No

The agent owner will pick up $Merchant_i$'s reputation value Rep_i from table MR, which is in this example set as 6. Then the agent owner will update table TA by executing the TRU algorithm. The updated TA is shown in table 9.

Table 9. Table TA after performing the transaction

TTP-ID	Trust	Rel	Sat	TotalTran	T-C	R-C
TTP_1	12.8	16.8	14.8	6	0	1
$TTPB_2$	-5	12.4	3.7	11	5	0
TTP_3	5	1.58	3.29	8	0	0
TTP_4	3	3	3	2	1	1

5. Conclusion

In this paper, we have critically analysed the related work in the topic area of distributed reputation management. Based upon the analysis, five important reputation metrics are defined to achieve a fairer evaluation of one's reputation. A novel Distributed Reputation Management scheme is subsequently designed based upon these metrics. The scheme allows a party *A*, e.g. a customer, to distribute a security sensitive task among several most trustworthy *TTP-hosts*. This is achieved by first choosing a subset of *TTP-hosts* with the highest trust and reliability levels. The scheme then credits/penalizes each *TTP-host* according to the feedback received by *A* from party *B*, e.g. a merchant. Incorporating this reputation management scheme in e-commerce applications can not only support secure delegation of security sensitive tasks, such as signature signing, but also deter cheating or misbehaving by e-commerce participants, thus improving our e-business environment. In addition, the distributed idea used in the scheme allows the provision of robust security services

As our future work, we plan to build an agent-based signature delegation framework by integrating the Distributed Reputation Management scheme with the agent-based threshold proxy signature scheme proposed in [Obamasak 2004] to facilitate robust and secure agent-based e-commerce activities.

REFERENCES

Bamasak, O. and N. Zhang, "A Secure Proxy Signature Protocol for Agent-Based M-Commerce Applications," Proceedings of the 9th IEEE Symposium on Computer and Communications, IEEE Computer Society, pp. 399-406, Alexandria, Egypt, July 2004.

Bryant, A. and B. Colledge, "Trust in Electronic Commerce Business Relationships," *Journal of Electronic Commerce Research*, Vol. 3, No. 2, pp. 32-39, 2002.

eBay, 2004. Available at http://www.ebay.com (accessed 20-8-2004).

ITU-T Recommendation X.500, ISO/IEC 9594-1, 1997, Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services.

Jøsang, A. and R. Ismail, "The Beta Reputation System," Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia, 2002.

Ketchpel, S. and H. Garcia-Molina, "Making Trust Explicit in Distributed Commerce Transactions," Proceedings of the 16th IEEE International Conference on Distributed Computing Systems (ICDCS96), pp. 270-281, 1996.

Kim, H., J. Baek, B. Lee and K. Kim, "Secret Computation with Secrets for Mobile Agent using One-Time Proxy Signature," Proceedings of the 2001 Symposium on Cryptography and Information Security (SCIS2001), pp. 845-850, 2001.

Kim, D. and I. Benbasat, "Trust-Related Argument in Internet Stores: A Framework for Evaluation," *Journal of Electronic Commerce Research*, Vol. 4, No. 2, pp 49-64, 2003.

Kotzanikolaou, P., M. Burmester, and V. Chrissikopoulos, "Secure Transactions with Mobile Agents in Hostile Environments,"*Proceedings* of the Fifth Australian Conference on Information Security and Privacy (ACISP 2000), LNCS, Vol. 1841, pp. 289-297, 2000.

Lee, B., H. Kim and K. Kim, "Strong Proxy Signature and its Applications," Proceedings of the 2001 Symposium on Cryptography and Information Security (SCIS2001), pp. 603-608, 2001.

Malaga, R., "Web-based Reputation Management Systems: Problems and Suggested Solutions," *Electronic Commerce Research*, Vol. 1, pp. 403-417, 2001.

Mambo, M., K. Usuda and E. Okamoto, "Proxy Signatures for Delegating Signing operation," Proceedings of the Third ACM Conference on Computers and Communications Security, pp. 48-57, 1996a.

Mambo, M., K. Usuda and E. Okamoto, "Proxy Signature: Delegation of the Power to Sign Messages," *IEICE Trans. Fundamentals*, E79-A, pp. 1338-1353, 1996b.

Marsh, S. P., "Formalizing trust as a computational concept, " Ph.D. Thesis, University of Stirling, Stirling, UK, 1994.

Reagle, J. M., "Trust in a cryptographic economy and digital security deposits: Protocols and policies," Master Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1996.

Sander, T. and C. Tschudin, “Protecting Mobile Agents against Malicious Hosts,” *Mobile Agents and Security,* LNCS, Vol. 1419, pp. 44-60, 1998.

Wang, C. L., L. Ye, Y. Zhang and D. Nguyen, "Subscription to Fee-Based Online Service: What Makes Consumer Pay for Online Content?," *Journal of Electronic Commerce Research*, Vol. 6, No. 4, pp. 304-311, 2005.

Xiong, L. and L. Liu, “A Reputation-Based Trust Model for Peer-to-Peer eCommerce Communities,” Proceedings of the IEEE International Conference on E-Commerce (CEC’03), pp. 275-284, 2003.

Yahoo! Auctions, 2004. Available at http://auctions.yahoo.com. (Accessed 20/8/2004).

Zacharia, G. and P. Maes, “Trust Management through Reputation Mechanisms,” *Applied Artificial Intelligence*, Vol. 14, pp. 881-908, 2000.