An Iterative Algorithm for Trust and Reputation Management

Erman Ayday
School of Electrical and Comp. Eng.
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: erman@ece.gatech.edu

Hanseung Lee
School of Electrical and Comp. Eng.
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: hanseung.lee@ece.gatech.edu

Faramarz Fekri
School of Electrical and Comp. Eng.
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: fekri@ece.gatech.edu

***Abstract*—Trust and reputation play critical roles in most environments wherein entities participate in various transactions and protocols among each other. The recipient of the service has no choice but to rely on the reputation of the service provider based on the latter's prior performance. This paper introduces an iterative method for trust and reputation management referred as ITRM. The proposed algorithm can be applied to centralized schemes, in which a central authority collects the reports and forms the reputations of the service providers as well as report/rating trustworthiness of the (service) consumers. The proposed iterative algorithm is inspired by the iterative decoding of low-density parity-check codes over bipartite graphs. The scheme is robust in filtering out the peers who provide unreliable ratings. We provide a detailed evaluation of ITRM via analysis and computer simulations. Further, comparison of ITRM with some well-known reputation management techniques (e.g., *Averaging Scheme*, *Bayesian Approach* and *Cluster Filtering*) indicates the superiority of our scheme both in terms of robustness against attacks (e.g., ballot-stuffing, bad-mouthing) and efficiency. Furthermore, we show that the computational complexity of the proposed ITRM is far less than the Cluster Filtering; which has the closest performance (to ITRM) in terms of resiliency to attacks. Specifically, the complexity of ITRM is linear in the number of clients, while that of the Cluster Filtering is quadratic.**

I. Introduction

Trust and reputation systems have found widespread adoption in online communities, web services, ad-hoc networks, P2P computing, and in e-commerce communities. In most environments, the consumer of the service (e.g., the buyer) has no choice but to rely on the reputation of the service provider (e.g., the seller) based on the latter's prior performance. Hence, the service recipient should take a prior risk before receiving the actual service. This risk puts the recipient into an unprotected position since he has no opportunity to try the service before he receives it. A reputation-management mechanism is a promising method to protect the consumers against deceitful service providers. By using a reputation-management scheme, an individual service provider's reputation can be formed by the combination of received reports (ratings). Hence, after each transaction, a party who receives the service (referred to as the rater) provides, to the central authority, its rating about the quality of the service provided for that transaction. The central authority collects the ratings and estimates the reputation of the service provider as well as the rating trustworthiness of the raters. The rating mechanism puts the service providers in a vulnerable position as malicious raters may give unfair ratings. Hence, the success of a reputation scheme depends on the robustness of the mechanism to accurately evaluate the service providers' reputations and the trustworthiness of the raters. Despite the past progress on reputation systems, there is still a need to develop reliable, scalable, and dependable schemes that would be resilient to the various ways a reputation system can be compromised.

Focusing mainly on centralized reputation systems, the ultimate objective of our work is to develop a trust and reputation management scheme that not only provides immunity against malicious ratings but also discourages the service providers from any unfair and discriminating behaviors. Our work on reputation systems stems from our prior success in the use of iterative algorithms, such as message passing techniques and belief propagation [1], [2] in the decoding of Low-Density Parity-Check (LDPC) codes in erasure channels [3], [4]. These algorithms rely on graph-based representations of codes, where decoding can be viewed as message passing between nodes in the graph. Moreover, they are shown to perform at error rates near what can be achieved by the optimal scheme, maximum likelihood decoding, while requiring far less computational complexity (i.e., linear in the length of the code). We believe that these significant benefits offered by iterative algorithms can be tapped in to benefit the field of reputation systems. To achieve this, we propose the Iterative Trust and Reputation Mechanism (ITRM).

The main strengths of the ITRM scheme are summarized in the following.

1) The proposed algorithm computes the reputations of the service providers accurately (with a small error) in a short amount of time in the presence of attackers.
2) ITRM is a robust and efficient methodology for detecting and filtering out unreliable ratings (from malicious raters) in a short amount of time.
3) ITRM detects the malicious raters with a high accuracy, and updates their trustworthiness accordingly. Hence, ITRM enforces the malicious raters to execute low grade attacks in order to remain undercover.
4) The proposed ITRM algorithm has the computational complexity which is linear with the number of raters. Hence, ITRM is scalable and suitable for large scale implementations.

The rest of this paper is organized as follows. In the rest of this section, we summarize the related work and present major attack models considered in this study. In Section II we describe ITRM in detail. Next, in Section III, we evaluate our proposed scheme by conducting computer simulations and a detailed analysis. Furthermore, we compare ITRM with some existing schemes. Finally, we provide the concluding remarks in Section IV.

A. Related Work

Several works in the literature have focused on building reputation-management mechanisms. We may classify reputation mechanisms for centralized systems as i) global reputation systems, where the reputation of a service provider is based on the ratings from general users [5], [6], and ii) personalized reputation systems, where the reputation of a service provider is determined based on the ratings of a group of particular users, which may be different in the eyes of different users [5], [7]. We note that our work falls under the category of global reputation systems. The most famous and primitive global reputation system is the one that is used in eBay. Other well-known web sites such as Amazon, Epinions , and AllExperts use a more advanced reputation mechanism than eBay. Their reputation mechanisms mostly compute the average (or weighted average) of the ratings received for a product (or a peer) to evaluate the global reputation of a product (or a peer) [5]. Hence, these schemes are vulnerable to collaborative attacks by malicious peers. Use of the Bayesian Approach is also proposed in [6], [8]. In these systems, the *a posteriori* reputation value of a peer is computed combining its *a priori* reputation values with the new ratings received for the peer. Finally, [7] proposed to use the *Cluster Filtering* method [9] for reputation. Different from the existing schemes, the proposed ITRM algorithm is a graph based iterative algorithm motivated by our previous success on message passing techniques and belief propagation algorithms.

B. Attack models

We consider two major attacks that are common for any trust and reputation management mechanisms. Further, we assume that the attackers may collude and collaborate with each other:
Bad-mouthing: Malicious raters collude and attack the service providers with the highest reputation by giving low ratings. In addition to the malicious raters, in some applications, bad-mouthing may be originated by a group of selfish raters who attempt to weaken high-reputation providers in the hope of improving their own chances as providers.
Ballot-stuffing: Malicious raters collude to increase the reputation value of peers with low reputations. Just as in bad-mouthing, in some applications, this could be mounted by a group of selfish consumers attempting to favor their allies.

II. Description of the Proposed Algorithm

As in every trust and reputation management mechanism, we have two main goals: 1. Computing the service quality (reputation) of the peers who provide a service (henceforth referred to as Service Providers or SPs) by using the ratings from the peers who used the service (referred to as the raters), and 2. Determining the trustworthiness of the raters by analyzing their past ratings. We let TR_j be the global reputations of the SPs. Further, TR_{ij} represents the rating that the rater i reports about the j^{th} SP whenever a transaction is completed between the two peers. Finally, R_i denotes the (rating) trustworthiness of the i^{th} rater-peer. All of these parameters may evolve with time. However, for simplicity, we omitted time dependencies from the notation.

A. Iterative Trust and Reputation Management Mechanism (ITRM)

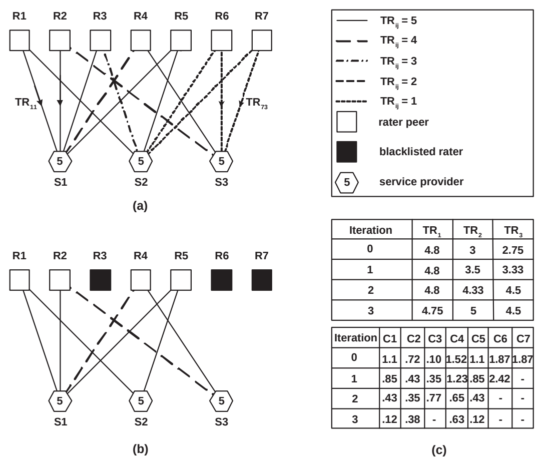
Our proposed iterative algorithm is inspired by our earlier work on the improved iterative decoding algorithm of LDPC codes in the presence of stopping sets [3], [4]. In iterative decoding of LDPC, every check-vertex (in the graph representation of the code) has some opinion of what the value of each bit-vertex should be. The iterative decoding algorithm would then analyze the collection of these opinions to decide, at each iteration, what value to assign for the bit-vertex under examination. Once the values of the bit-vertices are estimated, in the next iteration, those values are used to determine the satisfaction of the check-vertex values. The novelty of this work stems from the observation that a similar approach can be adapted to determine SPs' reputation values as well as the trustworthiness of the raters. Furthermore, the analysis of reputation systems resembles that of the code design problem. In LDPC, one of the goals is to find the decoding error for the a fixed set of check constraints. Similarly, in ITRM, our goal is to specify the regions of trust for the set of the system parameters. A region of trust is the range of parameters for which we can confidently determine the reputation values within a given error bound. We acknowledge, however, that we have a harder problem in the case of reputation systems as the adversary dynamics is far more complicated to analyze than the channel noise in the coding problem.

The first step in developing ITRM is to interpret the collection of the raters and the SPs together with their associated relations as a bipartite graph, as in Fig. 1(a). In this representation, each rater-peer corresponds to a *check vertex* in the graph, shown as a square and each SP is represented by a *bit vertex* shown as a hexagon in the graph. Assume that the graph has k check-vertices and N bit-vertices. If a rater i has a rating about the j^{th} SP, we place an edge with value TR_{ij} from the i^{th} check-vertex to the j^{th} bit-vertex. As time passes, we use the age-factored values as the edge values instead. To each edge $\{ij\}$, a value $WR_{ij} = w_{ij} TR_{ij}$ is assigned, where WR_{ij} is the age-factored TR_{ij} value. The factor $w_{ij}(t)$ is used to incorporate the time-varying aspect of the reputation of the SPs (i.e., time-varying service quality). We use a known factor $w_{ij}(t) = \lambda^{t-t_{ij}}$ where λ and t_{ij} are the fading parameter and the time when the last transaction between the rater i and the SP j occurred, respectively. If any new rating arrives from the rater i about the j^{th} SP, our scheme updates the new value of the edge $\{ij\}$ by averaging the new rating and the old value of the edge multiplied with the fading factor.

We consider slotted time throughout this discussion. At each time-slot (or epoch), ITRM will be executed using the input parameters R_i and WR_{ij} to obtain the reputation parameters (e.g., TR_j) and the list of malicious raters (referred to as the blacklist). Initially, the blacklist is set empty. Details of ITRM may be described by the following procedure at the L^{th} time-slot. Let R_i and TR_{ij} be the parameter values prior to the present execution (the L^{th} execution) of ITRM algorithm. Let also TR_j^{ν} and TR_{ij}^{ν} be the values of the bit-vertex and the $\{ij\}^{th}$ edge at the iteration ν of the ITRM algorithm. Prior to the start of the iteration ($\nu = 0$), we set $TR_{ij}^{\nu=0} = TR_{ij}$ and compute the initial value of each bit-vertex (referred to as the initial guess $TR_j^{\nu=0}$) based on the weighted average of the age-factored edge values (WR_{ij}^{ν}) of all the edges incident to the bit-vertex j. Equivalently, we compute

$$TR_j^{\nu} = \frac{\sum_{i \in A} R_i \times WR_{ij}^{\nu}}{\sum_{i \in A} R_i \times w_{ij}(t)}, \tag{1}$$

where A is the set of all check-vertices connected to the bit-vertex j. It is interesting to note that the initial guess-values resemble the received information from the channel in the channel coding problem. Then, the first iteration starts (i.e., $\nu = 1$). We first compute the average inconsistency factor C_i^{ν} of each check-vertex i using the values of the bit-vertices (i.e., $TR_j^{\nu-1}$) for which it is connected to. That is, we compute $C_i^{\nu} = [1/\sum_{j \in B} \lambda^{t-t_{ij}}] \sum_{j \in B} d(TR_{ij}^{\nu-1}, TR_j^{\nu-1})$ where B is the set of bit vertices connected to the check-vertex i and $d(\cdot,\cdot)$ is a distance metric used to measure the inconsistency. We use the $\mathcal{L}^1$ norm (absolute value) as the distance metric. Hence, $d(TR_{ij}^{\nu-1}, TR_j^{\nu-1}) = |TR_{ij}^{\nu-1} - TR_j^{\nu-1}| \lambda^{t-t_{ij}}$. After computing the inconsistency factor for every check-vertex, we order them by rank. Then, the check-vertex i with the highest inconsistency is selected and placed in the blacklist if its inconsistency is greater



Iteration	TR_1	TR_2	TR_3
0	4.8	3	2.75
1	4.8	3.5	3.33
2	4.8	4.33	4.5
3	4.75	5	4.5

Iteration	C1	C2	C3	C4	C5	C6	C7
0	1.1	.72	.10	1.52	1.1	1.87	1.87
1	.85	.43	.35	1.23	.85	2.42	-
2	.43	.35	.77	.65	.43	-	-
3	.12	.38	-	.63	.12	-	-

Fig. 1: Illustrative example of ITRM

than or equal to a definite threshold τ (whose choice will be discussed later). If there is no check-vertex with inconsistency greater than or equal to τ, the algorithm stops its iterations. Once the check-vertex i is blacklisted, we delete its rating TR_{ij}^{ν} for all the bit-vertices j it is connected to. Then, we update the values of all the bit-vertices using (1). This completes the first iteration of ITRM. The iterative algorithm proceeds to other rounds exactly in the same way as the first round, updating the values of the bit-vertices and blacklisting some other check-vertices as a result. However, once a check-vertex is placed in the blacklist, for the remaining iterations it is neither used for the evaluation of TR_js nor for the inconsistency measure of the check-vertices. We stop the iterations when the inconsistencies of all the check-vertices (excluding the ones already placed in the blacklist) fall below τ.

As an example, ITRM is illustrated in Fig. 1 for $k = 7$ raters, $N = 3$ SPs, and $\tau = 0.7$. It is assumed that the rates are integer values from $\{1, \ldots, 5\}$ and the actual reputations, $\hat{TR_j}$, are equal to 5. For simplicity, we assumed w_i's to be equal to 1 and R_i's to be equal for all raters. Furthermore, we assumed that the peers 1, 2, 3, 4, and 5 are honest but 6 and 7 are malicious raters. It is expected that (as we have assumed) the raters, although honest, may have different ratings about a particular SP. Hence, we choose rater 3 to be honest but giving incorrect (unreliable) ratings. The malicious raters (6 and 7) are mounting the bad-mouthing attack in this example. Fig. 1(a) shows the TR_{ij} values (illustrated by different line-styles) prior to the execution of ITRM. The TR_j values and the individual inconsistencies of the raters after each iteration are also illustrated in Fig. 1(c). We note that the algorithm stops at the third iteration when all the raters have inconsistencies less than τ. Fig. 1(c) indicates how ITRM gives better estimates of TR_j's compared to the weighted averaging method (which is correspond to the zero iteration). Fig. 1(b) illustrates the edges after the final iteration of ITRM. It is worth noting that the malicious raters 6 and 7 are blacklisted and their ratings are accordingly deleted. Moreover, rater 3, although honest, is also blacklisted at the third iteration. We note that this situation is possible when an honest but faulty rater's rating have a large deviation from the other honest raters.

B. Raters' Trustworthiness

We now explain how the R_i values are to be updated using the check vertices that are placed in the blacklist. The idea is to use the set of all past blacklists together in a Beta distribution [6]. Initially, prior to the first time-slot, for each rater-peer i, the R_i value is set to 0.5 ($\alpha_i = 1$ and $\beta_i = 1$). Then, if the rater-peer i is blacklisted, R_i is decreased by setting $\beta_i(t+1) = \hat{\lambda}\beta_i(t) + (C_i + 1 - \tau)^{\delta}$, otherwise, R_i is increased by setting $\alpha_i(t+1) = \hat{\lambda}\alpha_i(t) + 1$, where $\hat{\lambda}$ is the fading parameter and δ denotes the penalty factor for the blacklisted raters. We note that updating R_i values via the Beta distribution has one major disadvantage. An existing malicious rater with low R_i could cancel its account and sign in with a new ID. This problem may be prevented by updating R_i's using the method proposed in [5].

III. Security Evaluation of ITRM

In order to facilitate future references, frequently used notations are listed below.

D	Number of malicious raters
H	Number of honest raters
N	Number of service providers
m	Rating given by an honest rater
n	Rating given by a malicious rater
X	Total number of malicious rates TR_{ij} per a victim SP
d	Total number of newly generated outgoing edges, per time-slot, by an honest rater
b	Total number of newly generated outgoing edges, per time-slot, by a malicious rater
$\hat{b}$	Total number of newly generated attacking edges, per time-slot, by a malicious rater
Δ	$\hat{b}/b$ (i.e., fraction of attacking edges per time-slot)
μ	Total number of un-attacked SPs rated by an honest rater

A. Analytic Evaluation

We adopt the following models for various peers involved in the reputation system. We acknowledge that although the models are not inclusive of every scenario, they are good illustrations to present our results. We assume that the quality of service providers remains unchanged during time-slots. We provide the evaluation for the bad-mouthing attack only, as similar results hold for ballot-stuffing. Hence, without loss of generality, we will assume that $TR_j = 5$ for all the SPs. Moreover, we assume that ratings (i.e., TR_{ij}) generated by the non-malicious raters are distributed uniformly among the SPs. Furthermore, we assume that the rating m (provided by the non-malicious raters) is a random variable with folded normal distribution (mean 5 and variance 0.5), however, it takes only discrete values from 5 to 1. Moreover, we assume that the values of R_i for all the raters are set to the highest value (i.e., $R_i = 1$) for simplicity (which reflects the worst case). Finally, we assume that d is a random variable with Yule-Simon distribution, which resembles the power-law distribution used in modeling online systems, with the probability mass function (PMF) $f_d(d; \rho) = \rho B(d, \rho + 1)$, where $B(.,.)$ is the Beta function. For modeling the adversary, we make the following assumptions. We assume that the malicious raters initiate bad-mouthing and collude while attacking the SPs. Further, the malicious raters attack the same set Γ of SPs at each time-slot. In other words, we denote by Γ the set of size $\hat{b}$ in which every attacked SP has one edge from each of the malicious raters. The following discussions are developed for the time-slot t.

Let the random variable Θ be the number of unique raters who rated a specific SP in t elapsed time-slots. Also let Q be a random variable denoting the exponent of the fading parameter λ at the t^{th} time-slot. It can be shown that the probability distribution of both of these random variables can be easily obtained from the distribution of the number of ratings received by a SP in t elapsed time-slots. Using this, we can establish the following results.

A τ-eliminate-optimal Scheme: We declare a reputation scheme to be τ-eliminate-optimal if it can eliminate all the malicious raters whose inconsistency (measured from the averaging scheme) exceeds the threshold τ. Hence, such a scheme would compute the reputations of the SPs by just using the honest raters. Naturally, we need to answer two questions: First, for a fixed τ, what are the conditions to have a τ-eliminate-optimal scheme? Second, among all the eliminate-optimal schemes, which scheme (i.e., which value of τ) should we choose for the best performance? In the following discussion, we try to answer these two questions.

At each iteration, ITRM blacklists the rater-peer i with the highest inconsistency C_i if $C_i \geq \tau$. Obviously, the blacklisted rater should be a malicious one in all iterations. This criteria is given by the following lemma:

Lemma 1: Let Θ_j be the number of unique raters for the j^{th} SP. Then, a sufficient condition for the inconsistency C_i, at the first iteration, to exceed the threshold τ for all malicious raters is given by

$$\sum_{r \in \Lambda} \Psi_r \geq (\hat{b}m + b\tau) \quad (2)$$

Here, $\Psi_r = \frac{mX + n\Theta_r\lambda^Q}{X + \Theta_r\lambda^Q}$ for $r \in \Lambda$, where Λ is the index set of the set Γ.
Given $C_i \geq \tau$ for a malicious rater i, for a τ-eliminate-optimal scheme, we require that the inconsistency of the malicious rater exceeds the inconsistencies of all of the honest raters.

Lemma 2: (τ-eliminate-optimal condition): Let d_t be the total number of outgoing edges from an honest rater in t elapsed time-slots. Then, provided that Lemma 1 is met, ITRM would be a τ-eliminate-optimal scheme if the condition

$$\frac{\mu}{d_t} > 1 - \frac{\Theta\lambda^Q\Delta}{D} \quad (3)$$

is satisfied with high probability at the t^{th} time-slot.

The design parameter τ should be selected based on the highest fraction of malicious raters to be tolerated. To determine which scheme (i.e., which value of τ) we should choose for the best performance among all the eliminate-optimal schemes, we start with Lemma 2. We use a waiting time t such that (3) is satisfied with high probability. Then, among all τ values that satisfy (2) with high probability, we select the highest τ. In the following example, we designed the scheme to tolerate up to $W = 0.3$ (i.e., 30% malicious raters). For the given parameters $D+H = 200$, $N = 100$, $\Delta = 1$, $\rho = 1$ and $\lambda = 0.9$, we obtained the optimal $\tau = 0.4$. As shown in Fig. 2, for W lower than 0.3, the waiting time becomes shorter to have a τ-eliminate-optimal scheme for $\tau = 0.4$. However, the scheme may also blacklist a few non-malicious raters in addition to all the malicious ones when W is actually less than 0.3. This is because the optimal value of τ is higher for a τ-eliminate-optimal scheme when W is actually less than 0.3.

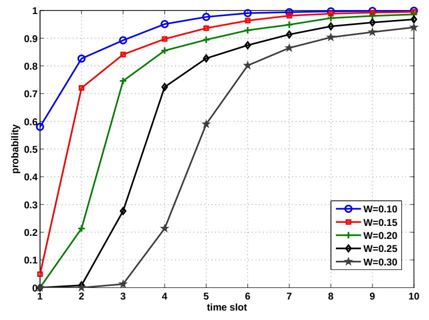


Fig. 2: Waiting time for τ-eliminate-optimal

B. Simulations

We have evaluated the performance of ITRM in the presence of bad-mouthing and ballot-stuffing. Here, we provide an evaluation of the bad-mouthing attack only, as similar results hold for ballot-stuffing. We compared the performance of ITRM with three well-known and commonly used reputation management schemes: 1) *The Averaging Scheme*, 2) *Bayesian Approach*, and 3) *Cluster Filtering*. The Averaging Scheme is widely used in well-known web sites such Amazon and AllExperts. The Bayesian Approach [6] updates the TR_j values using a Beta distribution. For this scheme, we assumed a deviation threshold of 0.4 and a trustworthiness threshold of 0.4, which is consistent with our τ selection (for details refer to [6]). Cluster Filtering [7], [9] performs a dissimilarity test among the raters and then updates the TR_j values using only the honest raters.

In all simulations, we considered the worst-case scenario in which the victims are chosen among the newcomer SPs in order to have the most adverse effect. We assumed that there were already 100 rater-peers and 50 SPs. Moreover, a total of 50 time-slots had passed since the initialization of the system, and ratings generated during those time-slots were distributed among the SPs in proportion to their reputation values (SPs with high quality of service made more transactions). After this initialization process, at time-slot zero, we introduced 100 more rater-peers as well as 50 more SPs as newcomers. Hence, we had $k = D + H = 200$ raters and $N = 100$ SPs in total. Further, we assumed that d is a random variable with Yule-Simon distribution as discussed in the analysis. At each time-slot, the newly generated ratings from honest raters are assigned to the SPs in proportion to the present estimate of their reputation values, TR_j. Let $\hat{TR_j}$ be the actual value of the reputation. Then, we obtained the performance of ITRM, for each time-slot, as the mean absolute error (MAE) $|TR_j - \hat{TR_j}|$, averaged over all the SPs that are under attack. We used the following parameters throughout our simulations: $b = \hat{b} = 5$, $\rho = 1$, $\lambda = \hat{\lambda} = 0.9$, the penalty factor $\delta = 10$, and $\tau = 0.4$ (the choice of τ is based on our analytical results).

We assumed that the malicious raters attack the same set Γ of SPs in each time-slot. Hence, at each time-slot, the malicious raters choose SPs from Γ and rate them as 4. The malicious raters do not deviate very much from the actual $\hat{TR_j} = 5$ values to remain undercover as many time-slots as possible (while still attacking). Note that we also tried higher deviations from the $\hat{TR_j}$ value and observed that the malicious raters were easily detected by ITRM in fewer time-slots. In Figure 3, the performance of ITRM is illustrated against the "bad mouthing" for $W = \frac{D}{D+H} = 0.10$ (10% malicious peers). The graph is obtained for different $\Delta = \hat{b}/b$ values. We observe that as the malicious peers attack with small number of edges (for low values of $\hat{b}$), it requires more time slots to have negligibly low error values. On the other hand, when the $\hat{b}$ values becomes very small ($\hat{b} = 1, 2$), it is hard to detect the malicious peers, which is consistent with our analytical results. Although the malicious peers stay undercover when they attack with very small number of edges, this type of an attack limits the malicious peers' ability to make a serious impact (they can only attack to a small number of SPs). We note that throughout the remaining simulations, we set $\Delta = 1$.

In Figure 4, we show the performance of ITRM for different W values when the malicious peers are employing the "bad mouthing". We note that ITRM guarantees significantly low errors regardless of the fraction of the malicious consumers. As W becomes larger, it takes more time to get negligibly small error values (which is consistent with our analysis).

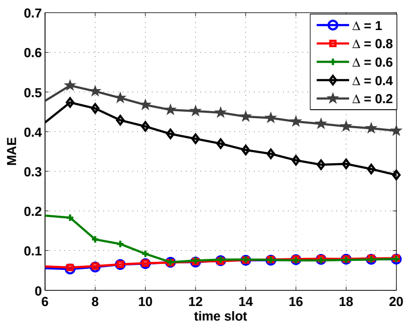


Fig. 3: MAE performance of ITRM versus time for bad mouthing when $W = 0.10$ and varying Δ

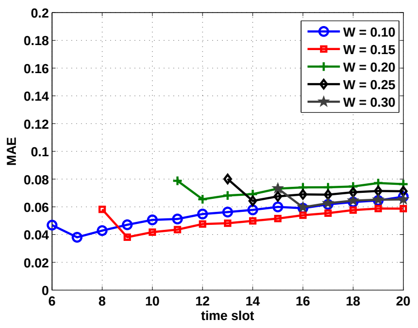


Fig. 4: MAE performance of ITRM versus time for bad mouthing and varying W

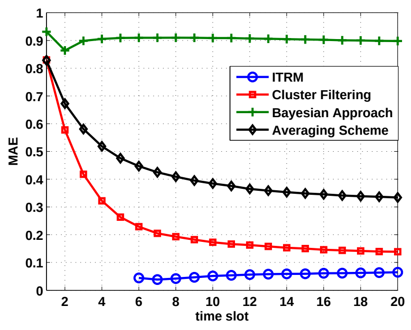


Fig. 5: MAE performance of various schemes for bad-mouthing when $W = 0.10$

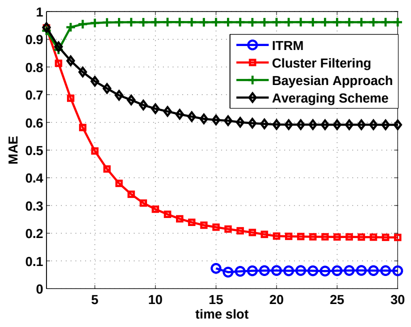


Fig. 6: MAE performance of various schemes for bad-mouthing when $W = 0.30$

Figures 5 and 6 illustrate the comparison of ITRM with the other schemes for the bad-mouthing attack when the fraction of malicious raters are 0.10 and 0.30, respectively. However, it is worth noting that for different values of Δ,we observed that ITRM still keeps its superiority over the other schemes. The lags in the plots of ITRM in Figs. 5 and 6 correspond to waiting times to include the newcomer SPs into the execution of ITRM, computed based on our analytical results presented in Fig. 2. On the other hand, we executed the other 3 schemes starting from the first time-slot, since we observed that their performances were better that way. From these simulation results, we conclude that ITRM significantly outperforms the Averaging Scheme and the Bayesian Approach in the presence of attacks. We identify that the reputation management scheme with the closest performance to ITRM is Cluster Filtering. However, as illustrated in Figs. 5 and 6, the error obtained by using Cluster Filtering is significantly high for the first set of time-slots (which is undesirable because it would introduce error into the SP selection). More importantly, the computational complexity of Cluster Filtering is much higher than ITRM. Specifically, assuming k rater-peers in the system, the number of operations required in both methods is illustrated in Table I. Therefore, while Cluster Filtering introduces quadratic complexity, the computational complexity of ITRM is linear with the number of raters. As a result, our proposed scheme is more scalable and suitable for large scale reputation systems.

	ITRM	Cluster Filtering
Addition	$O(k)$	$O(k^2)$
Multiplication	$O(k)$	$O(k^2)$

TABLE I: Complexity of Cluster Filtering and ITRM.

IV. Conclusions

In this paper, we introduced an "Iterative Trust and Reputation Management Scheme" (ITRM). Our work is a graph based iterative algorithm motivated by prior success on message passing techniques and belief propagation algorithms for decoding LDPC codes. The proposed ITRM is a robust mechanism to evaluate the quality of the service of the service providers from the ratings received from the recipients of the service (raters). Moreover, it effectively evaluates the providers' reputations and the trustworthiness of raters while introducing a linear computational complexity with respect to the number of raters. We studied ITRM by a detailed analysis, and showed the robustness using computer simulations. Besides, we compared ITRM with some well-known reputation management schemes and showed the superiority of our scheme both in terms of robustness and efficiency.

References

[1] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[2] J. Zhang and M.Fossorier, "Shuffled belief propagation decoding," *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*, pp. 12–15, Nov. 2002.

[3] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Trans. on Information Theory*, vol. 53, no. 2, pp. 599–614, Feb. 2007.

[4] B. N. Vellambi and F. Fekri, "Results on the improved decoding algorithm for low-density parity-check codes over the binary erasure channel," *IEEE Trans. on Information Theory*, vol. 53, no. 4, pp. 1510–1520, April 2007.

[5] G. Zacharia, A. Moukas, and P. Maes, "Collaborative reputation mechanisms in electronic marketplaces," *HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8*, 1999.

[6] S. Buchegger and J.Boudec, "Coping with false accusations in misbehavior reputation systems for mobile ad-hoc networks," *EPFL-DI-ICA Technical Report IC/2003/31*, 2003.

[7] C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pp. 150–157, 2000.

[8] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation," *HICSS '02: Proceedings of the 35th Hawaii International Conference on System Science*, 2002.

[9] P. Macnaughton-Smith, W. T. Williams, M. B. Dale, and L. G. Mockett, "Dissimilarity analysis: A new technique of hierarchical sub-division," *Natue(202)*, pp. 1034–1035, 1964.