Defending Multiple-user-multiple-target Attacks in Online Reputation Systems

Yuhong Liu[1], Yan (Lindsay) Sun[1], and Ting Yu[2]

[1]Department of Electrical and Computer Engineering University of Rhode Island, Kingston, RI 02881
[2]Department of Computer Science, North Carolina State University, Raleigh, NC 27695
Emails:{yuhong, yansun}@ele.uri.edu, tyu@csc.ncsu.edu

***Abstract*—As online reputation systems are playing increasingly important roles in reducing risks of online interactions, attacks against such systems have evolved rapidly. Nowadays, some powerful attacks are conducted by companies that make profit through manipulating reputation of online items for their customers. These items can be products (e.g. in Amazon), businesses (e.g. hotels in travel sites), and digital content (e.g. videos in Youtube). In such attacks, colluded malicious users play well-planned strategies to manipulate reputation of multiple target items. To address these attacks, we propose a defense scheme that (1) sets up heterogeneous thresholds for detecting suspicious items and (2) identifies target items based on correlation analysis among suspicious items. The proposed scheme and two other comparison schemes are evaluated by a combination of real user data and simulation data. The proposed scheme demonstrates significant advantages in detecting malicious users, recovering reputation scores of target items, and reducing interference to normal items.**

I. Introduction

As more people use the Internet for entertainment, building personal relationships, and conducting businesses, how to evaluate strangers' quality or trustworthiness in online systems becomes an important issue. Online reputation systems, also referred to as online rating systems, allow users to post their ratings/reviews on items in the system, aggregate these ratings/reviews and assign each item with a reputation score that indicates its quality. The items that receive user ratings can be *products* (e.g. in the Amazon product rating system), *services* (e.g. hotel ratings in various travel sites), *users* (e.g. sellers and buyers at eBay), and *digital content* (e.g. video clips at YouTube). Online reputation systems can help people evaluate the quality of online items before transactions, and hence greatly reduce the risks of online interactions.

More and more people get used to post their opinions in online rating/review systems and refer to those opinions before making their purchasing/downloading decisions. The Pew Internet & American Life Project has found that 26% of adult internet users in the U.S. have rated at least one product, service, or person using online rating systems [1].

However, not all the ratings/reviews are honest. Driven by the huge profits of online markets [2], attacks that attempt to mislead users' online decisions through dishonest ratings/reviews are gaining popularity. Sellers at the online marketplace boost their reputation by trading with collaborators [3]. Firms post biased ratings and reviews to praise their own products or bad-mouth the products of their competitors [4]. There are even ratings/reviews for the products that never exist. In the "Amazon window shop app" for iPad, one category, called "peculiar products", contains a lot of kooky and fake products, such as uranium ore, or a $40 Tuscan whole milk [5]. Surprisingly, these products receive thousands of ratings/reviews, although Amazon does not really sell them. Attacks against reputation systems can overly inflate or deflate item reputation scores, crash users' confidence in online reputation systems, eventually undermine reputation-centric online businesses and lead to economic loss.

In this work, we define the users providing dishonest ratings as *malicious users*, and the items receiving dishonest ratings from malicious users as *target items*. Based on malicious users' capability and goals, the attacks can be roughly classified into three categories.

1. **Independent attack**: *independent* malicious users insert dishonest ratings to mislead the reputation score of a *single* item. This is the simplest and the least organized type of attacks that mainly appears when online reputation systems were introduced in the mid-1990s.
2. **Single-target attack**: *colluded* malicious users collaboratively insert dishonest ratings to mislead the reputation score of a *single* item. These attacks are the most commonly known and investigated attacks. For instance, on IMDB (a movie site owned by Amazon), a low quality movie, Resident Evil: Afterlife, has kept an overly inflated score of 8.5 out of 10 during the first month of release, with more than 1,800 ratings [6]. Obviously, in this example, the dishonest ratings overwhelm the honest ratings in a long period of time.
3. **Multiple-target attack**: *colluded* malicious users collaboratively insert dishonest ratings to mislead the reputation scores of *multiple* items. This type of attacks, which is gaining popularity recently, is relatively new to the research community and may cause severe damage to the reputation systems. These attacks are mainly launched by some "rating companies", which provide "rating services" for different customers through their affiliate network of user IDs. For just $9.99, a company named "IncreaseYouTubeViews.com" can provide 30 "I like" ratings or 30 real user comments to your video clips on YouTube. Taobao, which is the largest Internet retail platform in China, has identified these reputation boosting services as a severe threat [7]. It is important to point out that malicious user IDs are often "reused" in the multiple-

target attacks. In other words, a subset of malicious users that attacked one target item may also be used to attack other target items.

Independent attacks can be easily addressed. In this type of attack, dishonest ratings that are far away from the honest opinions can be detected by various statistical methods [8], [9], whereas independent dishonest ratings that are close to the honest opinions usually do not cause much damage. Therefore, the current research efforts [9]–[13] mainly focus on single-target attacks that involve colluded malicious users [14]. However, very limited work has been done to address the multiple-target attacks.

Current defense schemes, mainly designed for single-target attacks, can address the multiple-target attacks to some extent. In particular, the rating statistics of each item can be examined independently by existing defense methods. If a user issues suspicious ratings to an item that is detected under attack, the *trust* of this user drops. Ideally, the malicious users who rate for multiple target items should have low trust values. However, this approach has three major limitations.

- Most of the statistical defense schemes consider each item equally in terms of setting detection parameters (e.g. rating distribution parameters, detection threshold). However, in practice, the rating statistics of different items are not homogenous. Some items may naturally receive highly diverse ratings while others may receive similar ratings. Therefore, identical detection parameters can hardly fit all items.
- Trust in raters is usually computed based on users' past good/bad behaviors. Malicious users can accumulate high trust values by providing honest ratings to the items that they are not interested in. Such type of behaviors can make trust evaluation much less effective.
- Correlation among target items is not studied. In most of the existing schemes, including our previous work [15], items under strong attacks are detected, but those under moderate or weak attacks may not be detected even if there is overlap between the malicious users in moderate/weak attacks and the malicious users in strong attacks.

As a summary, existing defense schemes are not designed for and cannot effectively address multiple-target attacks, which are becoming important threats against online reputation systems.

In this paper, we propose the defense for multiple-target attacks from a new angle. The proposed scheme does not evaluate trust in raters. Instead, it sets up heterogeneous thresholds for detecting suspicious items, and further identifies target items based on correlation analysis among suspicious items.

The proposed scheme is evaluated using a combination of real user data and simulation data. Compared with the beta-function based detection scheme [10] and the iterative refinement scheme [11], the proposed scheme achieves significantly better ROC* in the detection of malicious users, and has less impact on normal items that are not under attack. Furthermore, its effectiveness is much less sensitive to attack scenarios and parameters selections.

The rest of the paper is organized as follows. Section II discusses related work, Section III introduces details of the proposed scheme, Section IV presents experiment results, followed by discussion in Section V and conclusion in Section VI.

*Receiver operating characteristic

II. Related Work

To protect reputation systems, defense schemes have been proposed from four angles.

The *first* angle is increasing the cost of acquiring multiple user IDs by binding user identities with IP addresses [16] and using network coordinates to detect sybil attacks [17].

The *second* angle is endogenous discounting of dishonest ratings [18]. Methods in this category directly differentiate dishonest ratings from normal ratings based on the statistic features of the rating values. In a Beta-function based approach [10], a user is determined as a malicious user if the estimated reputation of an item rated by him/her lies outside q and $(1-q)$ quantile of his/her underlying rating distribution. An entropy based approach [9] identifies ratings that bring a significant change in the uncertainty in rating distribution as dishonest ratings. In [8], dishonest rating analysis is conducted based on Bayesian model.

The *third* angle is exogenous discounting of dishonest ratings [18]. Methods in this category evaluate the reliability of a given rating based on the trust/reliability of the user who provides this rating. The iteration refinement approach proposed in [11] assigns a weight to each rating according to the "judging power" of the user who provides this rating. In [12], a user's trust is obtained by cumulating his/her neighbors' beliefs through belief theory. REGRET reputation system, proposed in [19], calculates user trust based on fuzzy logic. Flow models, such as EigenTrust [13] and Google PageRank [20], compute trust or reputation by transitive iteration through looped or arbitrarily long chains.

The *fourth* angle is studying correlation among users to detect dishonest ratings [15], [21]. Schemes that investigate only correlation among individual users are not effective in addressing multiple-target attacks. In multiple-target attacks, many colluded malicious users may not have similar rating behaviors because they attack different target items or different sets of target items. What really matters is the correlation among target items, which is not exploited in the current schemes.

The previous schemes are designed for single-target attacks, and have inherited limitations, when used to address multiple-target attacks. The proposed scheme, which handles multiple-target attacks from a new angle, can also address single-target attacks, and is compatible with many schemes in the first, third and fourth category. Notice that some rating sites allow to have anonymous user ratings. The proposed scheme, as well as many other schemes, cannot handle attacks from anonymous users.

III. Proposed Defense Scheme

A. Overview of Defense Scheme

The proposed scheme contains two main modules: *suspicious item detection* and *item correlation analysis*. The general

idea is to first preselect suspicious items as items whose rating distributions are highly likely to be abnormal, then conduct correlation analysis among these suspicious items and determine the items with strong correlations as target items.

In the suspicious item detection module, there are three submodules: (1) change interval detection, (2) CvT analysis and (3) suspicious item selection. Here, CvT is the abbreviation for Change Interval versus Threshold, which will be explained in details in Section III-B2.

To detect suspicious items, we first detect changes in the rating distribution for each item. In many practical reputation systems, items have intrinsic quality, which should be reflected in the distribution of their ratings. If there are rapid changes in the distribution of rating values, such changes can serve as indicators of anomaly. Therefore, the first submodule is

1. *Change Interval Detection*: for each item, detect its *change interval* (CI), the time intervals in which rating values "change rapidly". Manipulation highly likely takes place in such time intervals.

However, what kind of change is "rapid change"? The answers are different for different items. As discussed in Section I, rating statistics of items are not homogenous, and an identical detection threshold cannot fit all items. The proposed scheme addresses this problem by the second and third submodules.

2. *CvT analysis*: for each item, extract its *CvT feature* (i.e. how item's CI changes with detection threshold). Then the CvT figure is constructed to visualize CvT features for all items in the system. From this figure, we observe that the CvT features of normal items follow a certain pattern, and the items whose CvT features are quite different from others are highly suspicious. This type of analysis and visualization has never been reported in the current literature.
3. *Suspicious Item Selection*: set heterogeneous detection thresholds according to the CvT figure and select suspicious items as the items with abnormal CvT features.

The suspicious item detection module can effectively detect items whose rating statistics change abnormally. However, it may not be able to avoid some false alarms when normal items also experience changes due to randomness in normal users' rating behaviors or item quality change. The changes in normal items can be even larger than the changes in target items especially when malicious users conduct moderate/weak attacks.

Recall that one important feature of multiple-target attacks is that malicious users are often "reused". As a consequence, we can find correlation between two target items when some malicious users rate both target items. This approach is particularly useful for identifying (a) target items that experience moderate/low level of manipulation and (b) unpopular target items that experience strong manipulation. To see this, let us consider two example cases.

- In Case 1, an item receives 100 honest ratings and 10 dishonest ratings, and values of these dishonest ratings are close to that of honest ratings. Whereas traditional schemes may not discover this relatively weak attack, the correlation analysis may find that this item is highly correlated with another target item under strong attack.
- In Case 2, an item receives 10 honest ratings and 20 dishonest ratings. Since the dishonest ratings overwhelm the honest ratings for this unpopular item, many traditional schemes are not effective to detect such attacks. However, the correlation analysis may reveal that this item and another target item share many common raters, which leads to suspicion.

Therefore, we propose

4. *Item Correlation Analysis (ICA)*: determine target items and malicious users by analyzing correlation among suspicious items.

It is important to point out that ICA is not a trivial task, because there may only be a very small overlap between the malicious user group attacking one target item and the malicious user group attacking another target item.

The major components of the proposed scheme will be described in details in Section III-B and III-C.

B. Suspicious Item Detection

To detect the suspicious items whose reputation scores are possibly manipulated, the suspicious item detection module is employed to (1) monitor changes in the rating distribution (using the method in [15]) for each individual item, (2) analyze the CvT features of all items in the system, and (3) identify those items with abnormal CvT features as suspicious items.

1) **Change Interval (CI) Detection:** To manipulate item reputation, dishonest ratings would inevitably cause changes in rating distribution. Therefore, change detection can be used as the first step toward anomaly detection in reputation systems. In this work, we adopt the change detector proposed in [15]. A brief introduction of this change detector is as follows.

The *rating sequence* of an item is defined as all of the ratings provided to the item ordered according to the time when these ratings are provided. For each item, a change detector is applied to its rating sequence. Let y_k denote the k^{th} rating of an item's rating sequence. Assume that the original mean value of the ratings is μ_0, and the change to be detected is the rating mean value change that exceeds a certain range ν. That is, the mean value after change is either above $\mu_1^+ = \mu_0 + \nu$ or below $\mu_1^- = \mu_0 - \nu$. The *decision functions* at the k^{th} rating are:

$$g_k^+ = max((g_{k-1}^+ + y_k - \mu_0 - \nu/2), 0), \tag{1}$$

$$g_k^- = max((g_{k-1}^- - y_k + \mu_0 - \nu/2), 0), \tag{2}$$

where g_k^+ is computed to detect positive changes and g_k^- is computed to detect negative changes. Here, g_k^+ and g_k^- can be calculated as soon as the k^{th} rating arrives.

The threshold used in the change detector is denoted by $\bar{h}$. Then the alarm time t_a, when the change detector raises an alarm, is defined as:

$$t_a = min\{k : g_k^+ \geq \bar{h} || g_k^- \geq \bar{h}\} \ , \tag{3}$$

which is the time when either g_k^+ or g_k^- is greater than $\bar{h}$.

For details of this change detection method, interested readers can refer to [15].

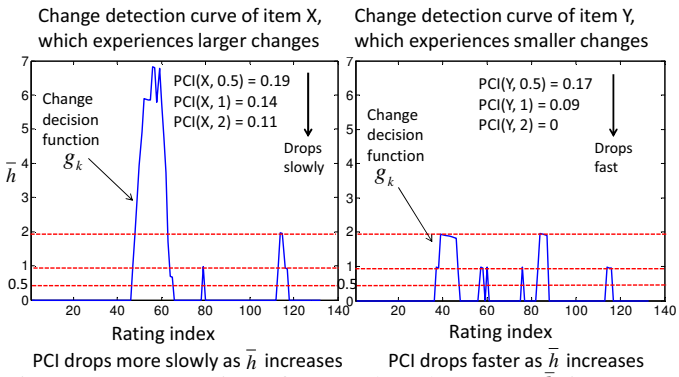

Fig. 1: Demonstration of PCI value change (as $\bar{h}$ increases, PCI drops more slowly if the signal experiences larger changes.)

2) **CvT Figure Generation:** As discussed in Section III-A, the change detection with identical threshold cannot fit all items. Then how to set up detection thresholds for different items? To solve this problem, we first define the Percentage of Change Interval (PCI) as:

$$PCI = \frac{\text{total length of all detected } CIs}{\text{length of the whole rating sequence}}.$$

PCI value describes whether the rating statistics of an item experiences long lasting changes or short random changes. As an example, assume that we detect two change intervals (CI) for item j when setting $\bar{h} = k$. The first CI starts at time 50 and ends at time 77, the second CI starts at time 117 and ends at time 201, and length of the rating sequence (time between the first rating and the last rating received for item j) is 450. Then PCI of item j at threshold k, denoted as $PCI(k, j)$, is computed as $\frac{(77-50)+(201-117)}{450} = 0.247$.

From the definition, we can see that PCI value largely depends on change detection threshold $\bar{h}$. How PCI value changes along with $\bar{h}$ is demonstrated in Figure 1, where item X receives both honest and dishonest ratings while item Y receives only honest ratings. Recall that when the change decision function g_k is above $\bar{h}$, a change interval is detected. From Figure 1, two important observations are made.

- When $\bar{h}$ increases, the changes with smaller amplitude and shorter duration, which are usually caused by normal variations, will be ignored by the change detector, which reduces the PCI value.
- How fast the PCI value changes with the threshold $\bar{h}$ reflects whether the detected changes are smaller-amplitude-shorter-duration or larger-amplitude-longer-duration. In other words, if the changes in the ratings of an item have large amplitude and sustain for sometime, which is highly suspicious, the PCI value of this item will drop slowly when $\bar{h}$ increases.

Let *CvT feature* denote how PCI value changes along with the threshold $\bar{h}$. Based on the above observations, we visualize the CvT feature of all items through the CvT figure, which is constructed in the following 3 steps.

1. For each item, calculate the PCI value for $\bar{h}_0$, which is the lowest change detector threshold. A representative value of $\bar{h}_0$ is 0. In other words, we calculate $PCI(\bar{h}_0, j)\ \forall j$. Recall that $PCI(\bar{h}_i, j)$ denote the PCI value of item j under change detector threshold $\bar{h}_i$.
2. Order items according to $PCI(\bar{h}_0, j)$ from low to high, and assign each item a new index, called C-index, according to this order. The C-index is denoted by c. For example, if item j's $PCI(\bar{h}_0, j)$ is the 2nd lowest, the C-index of item j is $c = 2$.
3. For each item, calculate the PCI values for different change detector thresholds. Let x-axis be the C-index c of items, y-axis be the threshold $\bar{h}$, and the color represent value of $PCI(\bar{h}, c)$.

The CvT figure can be viewed as an image. An example of the CvT figure is shown as Figure 2. The data used to generate Figure 2 is from a cyber competition [22], which contains ratings for 300 items. Details of this data set will be described in Section IV-A1. The change detector threshold values are chosen from 0 to 4, and $\bar{h}_0$ is 0. Figure 2 demonstrates the CvT feature of all items. Items with large PCI values at $\bar{h}_0$ lie on the right side of the figure, and PCI value of each item drops as the change detector threshold increases.

Next, we generate the contours using the *contourf* function in Matlab. These contours are marked with black lines in Figure 2. Each contour curve is composed of points with the same PCI value. Let z denote this PCI value. A point (x, y) on the contour curve with value z means that the item with C-index $c = x$ has PCI value z when the change detection threshold $\bar{h} = y$. In other words, a contour curve clearly shows the corresponding change detection thresholds that yield the same PCI value for different items.

Each contour has a slope and spikes. From Figure 2, we can observe that excluding the spikes, these contours are roughly parallel, meaning that the slopes of different contours are roughly the same. It indicates that when threshold $\bar{h}$ increases for a certain amount, most of the items' PCI values drop at similar speeds. In other words, most of the items in the system have similar CvT features. Then how would a spike appear? For example, when the change detector is sensitive (i.e. with threshold $\bar{h}_0$ and can be triggered by small changes), the PCI value detected for item c is between that for item $c-1$ and that for item $c+1$. As the threshold increases, the PCI value for item $c-1$ and $c+1$ quickly decreases, which is normal. However, the PCI value for item c does not decreases as quickly as its "neighbors" (i.e. item $c-1$ and item $c+1$). Then, a spike will be shown for item c on the CvT figure. Recall that target items tend to have larger-amplitude-longer-duration changes, leading to a slow drop in PCI values. Therefore, *the items with spikes have CvT feature different from that of most other items and are highly suspicious*.

As a summary, CvT figure is constructed to visualize the CvT features of all items from system points of view. The rich information in the CvT figure can provide us a way to detect target items that demonstrate abnormal CvT features.

3) **Suspicious Item Selection:** Due to items' heterogeneity, it is difficult to set up a universal detection threshold suitable for all items in the system. This problem is rarely solved in current reputation defense schemes.

From the explanation in the previous subsection, we can

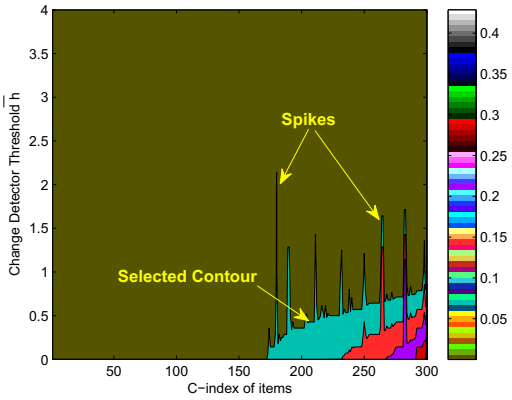

Fig. 2: An example of CvT figure

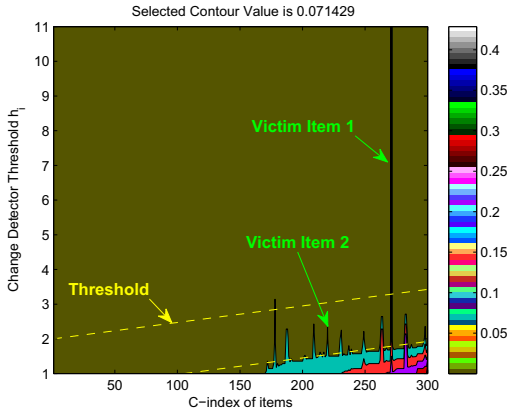

Fig. 3: CvT figure with two target items

see that a target item tends to have a spike in the CvT figure. Therefore, we propose to set heterogeneous thresholds for different items according to slopes of the contours in the CvT figure. In particular, suspicious items are selected as follows.

- After constructing the CvT figure, obtain the contour curve that goes through the largest number of items. As shown in Figure 2, this selected contour curve is "higher" than other contours. The contour value is 0.0714. That is, all the points on this contour curve have PCI value 0.0714.
- The 1st order linear regression model $y = ax+b_0$ is used to fit the selected contour curve, where a is the estimated slope of the contour curve.
- Set the heterogeneous threshold for each item c as $Th(c) = ac+b_0+b_s$, where a is the estimated slope of the selected contour curve, c is the C-index (item index after reorder) of an item, and $b_s(> 0)$ is called the *detection threshold offset*. Note that $Th(c)$ is roughly parallel to the contour curve, such that the threshold for the item with lower variation (lower C-index) is smaller than that for the item with higher variation (higher C-index). Here, b_s is the key parameter of the proposed scheme. The larger is b_s, the less sensitive is the detector. To detect more target items, a smaller b_s can be employed.
- Suspicious items are selected as items whose PCI values are larger than 0, when $\bar{h} = Th(c)$. That is, item c is marked as a suspicious item if $PCI(Th(c), c) > 0$.

We would like to emphasize that change detector threshold (i.e. $\bar{h}$, $\bar{h}_0$, $\bar{h}_i$), heterogeneous suspicious item selection thresholds (i.e. $Th(c)$), and the detection threshold offset (i.e. b_s) are different concepts. The CvT figure is constructed by varying $\bar{h}$; the suspicious items are detected by checking whether their PCI values at threshold $Th(c)$ are above 0; and $Th(c)$ is determined by b_s and the slope of the contour.

C. Item Correlation Analysis (ICA)

1) **Basic Procedure of ICA:** As discussed in Section III-A, changes in normal ratings can generate false alarms. Figure 3 shows a CvT figure with two target items: one item (with C-index $i_x = 274$) under strong attack and the other item (with C-index $i_y = 214$) under moderate attack. The specific parameters of the strong and moderate attacks will be presented in Section IV. We observe that item i_x generates an obvious spike, but the spike generated by item i_y is much less obvious. If we set up the detection threshold $Th(c)$ as shown in Figure 3, four normal items are marked as suspicious items. This false alarm problem is due to the fact that ratings of some normal items can have large variations and generate spikes on the CvT figure.

An important feature of multiple-target manipulation is that malicious users attack multiple target items. These colluded malicious users can lead to a correlation among those target items. Therefore, we introduce *item correlation analysis* (ICA) to identify target items and the corresponding malicious users. Details of ICA are shown in Procedure 1 and described as follows.

- **step 1** Let I_{sus} denote all suspicious items selected by the *suspicious item detection* module. Assume that there are s items in I_{sus} and $I_{sus} = \{i_1, i_2, \cdots, i_s\}$.
- **step 2** Calculate the item correlation for each pair of items in I_{sus}. Let $C^{item}_{i_x,i_y}$ denote the correlation between item i_x and item i_y.
- **step 3** Identify the suspicious user set, U_{i_x,i_y}, which contains the users who are suspected to attack item i_x and i_y.
- **step 4** Among all the item correlations calculated in step 2, obtain the maximum one as C^{item}_{max}.
- **step 5** Set the threshold for identifying highly correlated items as $T^{corr} = C^{item}_{max} \cdot P_h$, where P_h is a parameter between 0 and 1.
- **step 6** For any pairs of items, if $C^{item}_{i_x,i_y} > T^{corr}$, item i_x and i_y are determined as target items, and the users in U_{i_x,i_y} are determined as malicious users.

As long as it is not too large or too small, the parameter P_h in step 5 does not have major impact on the overall performance. We studied P_h value ranging from 0 to 0.9, and found that the proposed scheme performs quite stable when P_h is between [0.3, 0.7]. Thus, in later experiments, we set P_h = 0.7.

Notice that Steps 1-6 are designed to handle multiple-target attacks, and may not be able to address single-target attacks in which a target item is not highly correlated with other items. Therefore, we add one additional step.

- **step 7** For each item that is in I_{sus} but not determined as a target item yet, if its corresponding spike is higher than $Th(c) + \delta$, this item is also identified as a target

Procedure 1 Basic Procedure of ICA

1: Given $I_{sus} = \{i_x | 1 \leq x \leq s\}$ //suspicious item set
2: **for** each pair of i_x and $i_y \in I_{sus}$ **do**
3: $[C^{item}_{i_x\_i_y}, U_{i_x\_i_y}] = Item\_Correlation(i_x; i_y)$
4: **end for**
5: $C^{item}_{max} = max_{1\leq x\leq s, 1\leq y\leq s} C^{item}_{i_x,i_y}$ where $x \neq y$
6: $T^{corr} = C^{item}_{max} \cdot P_h$
7: $I_{target} = \{i_x, i_y |\ C^{item}_{i_x,i_y} \geq T^{corr}\}$//target items
8: $U_{mal} = \{U_{i_x,i_y} | i_x, i_y \in I_{target}\}$//malicious users

item. Here, δ ($\delta > 0$) determines the sensitivity of detecting the single-target attacks. The users who provide suspicious ratings to this item during its change intervals are determined as malicious users.

After this modification, the proposed scheme is extended to handle single target attacks. Even if malicious users attack only one target item, this item can be identified if the attack is strong enough and leads to a high spike in the CvT figure.

2) **User Distance, User Correlation and Group Correlation:** The most important function in the ICA analysis is $Item\_Correlation(i_x; i_y)$ in Procedure 1, which calculates the correlation between items i_x and i_y (i.e. $C^{item}_{i_x,i_y}$) and determines suspicious user sets for items i_x and i_y (i.e. U_{i_x,i_y}). Before discussing about this function, we first introduce some important related concepts: *user distance*, *user correlation* and *group correlation*.

Let D_{u_p,u_q} denote the *user distance* between user u_p and user u_q. Assume these two users have provided ratings to m common items, denoted by item $j^{p,q}_1, j^{p,q}_2, \cdots, j^{p,q}_m$. For item $j^{p,q}_w$, assume that user p provides rating value $r_{p,w}$ and user q provides rating value $r_{q,w}$. Then, the distance between user u_p and u_q is calculated as

$$D_{u_p,u_q} = \frac{1}{m}\sqrt{\sum_{w=1}^{m}(r_{p,w} - r_{q,w})^2}. \quad (4)$$

Note that the Euclidean distance calculation in (4) only considers the ratings to items that are rated by both user u_p and u_q. When two users do not rate any items in common, the distance between them is set as ∞.

User correlation can be converted from *user distance* as:

$$C^{user}_{u_p,u_q} = \begin{cases} 0 & D_{u_p,u_q} > \alpha \\ \frac{(D_{u_p,u_q} - \alpha)^2}{\alpha^2} & D_{u_p,u_q} \leq \alpha \end{cases} \quad (5)$$

Here, $C^{user}_{u_p,u_q}$ denotes the correlation between two users u_p and u_q, and α is a parameter set by system. This conversion is nonlinear because we care more about users with small distances than users with large distances. Therefore, correlation of user pairs with large distances (above α) is set to 0, and $\frac{(D_{u_1,u_2} - \alpha)^2}{\alpha^2}$ is used to emphasize users with small distances.

Based on user correlation $C^{user}_{u_p,u_q}$, we calculate the *group correlation*. Let $C^{group}(S_1; S_2)$ denote the correlation between two user groups S_1 and S_2.

$$C^{group}(S_1, S_2) = \sum_{u_p \in S_1} \sum_{u_q \in S_2} C^{user}_{u_p,u_q} \quad (6)$$

As shown in (6), correlation between two groups is computed as the sum of pairwise user correlation.

C-index	178	220	237	250	271	284
178	n/a	1.14	0.86	0.81	5.55	0.86
220	1.14	n/a	4.05	4.55	**111.00**	3.30
237	0.86	4.05	n/a	0.88	7.80	0.89
250	0.81	4.55	0.88	n/a	8.15	0.83
271	5.55	**111.00**	7.80	8.15	n/a	6.84
284	0.86	3.30	0.89	0.83	6.84	n/a

TABLE I: Correlation calculation illustration

3) **Item Correlation Calculation and Malicious User Identification:** Next, we describe how the function $Item\_Correlation(i_x; i_y)$ calculates item correlation and determines malicious users. There are 5 major steps.

First, for item i_x, select all users who have rated item i_x during the change intervals of i_x. Recall that the change intervals of an item are determined by the change detector (see Section III-B1). Let $S^{i_x,CI}$ denote these selected users. Since dishonest ratings are likely to occur in change intervals, $S^{i_x,CI}$ is expected to contain most of malicious users if not all, as well as some normal users who happen to rate in the change intervals.

Second, calculate the distance between any pairs of users in $S^{i_x,CI}$, and according to those distances, divide users in $S^{i_x,CI}$ into two clusters, using the Macnaughton-smith [23] method. Users "close" to each other are put into one cluster. Let $S^{i_x,CI}_1$ and $S^{i_x,CI}_2$ denote these two clusters, respectively. Recall that malicious users tend to have similar behavior (i.e. shorter distance among them) and be grouped into the same cluster. Therefore, one cluster should contain mainly malicious users and the other should contain mainly normal users.

Third, repeat the above two steps for item i_y, and generate two clusters denoted by $S^{i_y,CI}_1$ and $S^{i_y,CI}_2$.

Fourth, if some malicious users attack both item i_x and item i_y, the cluster containing malicious users of item i_x and that of item i_y should have larger correlation. Thus, we calculate the following 4 group correlation values using (6).

$$g_{1,1} = C^{group}(S^{i_x,CI}_1, S^{i_y,CI}_1),\ g_{1,2} = C^{group}(S^{i_x,CI}_1, S^{i_y,CI}_2)$$

$$g_{2,1} = C^{group}(S^{i_x,CI}_2, S^{i_y,CI}_1),\ g_{2,2} = C^{group}(S^{i_x,CI}_2, S^{i_y,CI}_2).$$

Then, the largest correlation value should correspond to the two malicious user clusters. Specifically, let $g_{a,b}$ be the largest one among $\{g_{1,1}, g_{1,2}, g_{2,1}, g_{2,2}\}$. Item correlation is obtained as $C^{item}_{i_x,i_y} = g_{a,b}$.

Finally, the users in the clusters that yield the largest correlation are marked as candidates of malicious users for item pair i_x and i_y. That is

$$U_{i_x,i_y} = S^{i_x,CI}_a \bigcup S^{i_y,CI}_b \quad (7)$$

The output of the function $Item\_Correlation(i_x; i_y)$ is $C^{item}_{i_x,i_y}$ and U_{i_x,i_y}. Later, if item pair i_x and i_y is detected as target item pair, as in Procedure 1, U_{i_x,i_y} will be considered as malicious users.

As a summary, the main idea of ICA is to compute correlation among different suspicious items, and identify highly correlated items as target items. The computation of the item correlation depends on the rating behavior similarity among users, user clustering, and the "shortest" distance between user clusters.

Using the same data that generate Figure 3, we illustrate the results of item correlation analysis in Table I. In this case, item with C-index 178, 220, 237, 250, 271 and 284 are in the suspicious item set (I_{sus}), and items with C-index 220 and 271 are target items. Table I shows the pairwise item correlation for items in I_{sus}. It is clearly seen that the correlation between item 220 and 271 is much higher than that of other pairs of items.

IV. Experiment Results

A. Testing Data Preparation and Experiment Setup

1) **Attack Data Set:** From 05/12/2008 to 05/29/2008, we hold a cyber competition for collecting attack data against an online rating system [22]. This data set has the following features:

- The normal rating data was extracted from a real e-commerce website, and contained ratings from 300 users to 300 products during 150 days. The rating values are integer values ranging from 1 to 5.
- In the cyber competition, real human participants (a) downloaded the normal rating data set, (b) created attacks that aimed to mislead the reputation score of item v_0 (a specific item determined by the competition rule), with no more than 30 malicious user IDs and 100 dishonest ratings, (c) submitted their attack data, and (d) received feedback about the strength of their attacks and their ranking among all participants.

Each submitted attack, referred to as a *manipulation profile*, contains the following information.

- The *number of malicious users* used in this attack, denoted by N_m.
- All *ratings* provided by the malicious user IDs. Note that the dishonest ratings can be given to many items, not limited to item v_0.
- The *Mean Offset* (M_o) computed as the fair reputation of item v_0 (i.e. the mean of honest ratings) minus the mean of dishonest ratings given to item v_0.

The N_m and M_o values can roughly describe the level of manipulation. Generally speaking, larger N_m and/or larger M_o values mean stronger manipulation.

To construct representative data set for testing, we selected 3 subsets of attack data.

- Strong attack set, denoted by AS_{str}, contains all manipulation profiles that has $N_m = 30$ and M_o between (2.5, 3]. There are 12034 such manipulation profiles.
- Moderate attack set, denoted by AS_{mod}, contains all manipulation profiles that has $N_m = 15$ and M_o between (1.5, 2]. There are 4202 such manipulation profiles.
- Weak attack set, denoted by AS_{weak}, contains all manipulation profiles that has $N_m = 10$ and M_o between (1, 1.5]. There are 726 such manipulation profiles.

2) **Construction of Multiple-Target Attacks:** Since there is only one target item (i.e. item v_0) in the cyber competition, we need to create attacks against multiple items in order to test the proposed scheme against multiple-target attacks.

Besides item v_0, we select one additional target item that is similar to v_0, in terms of the number/mean/variance of honest ratings. This additional target item is referred to as item v_1. We then create several typical multiple-target attack scenarios that have two target items as shown in Table II. The first column is attack scenario type. We choose 4 typical types as strong-strong, strong-moderate, strong-weak, and moderate-moderate, each of which describes manipulation levels of these two target items. The other two columns explain how to construct multiple-target attacks based on the manipulation profiles.

Attack Scenario Index	Manipulation profile for item v_0	Manipulation profile for item v_1
	is randomly selected from	
strong-strong	AS_{str}	AS_{str}
strong-moderate	AS_{str}	AS_{mod}
strong- weak	AS_{str}	AS_{weak}
moderate- moderate	AS_{mod}	AS_{mode}

TABLE II: Typical attack scenarios with two target items

For example, strong-moderate means that item v_0 experiences a "strong" attack and item v_1 experiences a "moderate" attack. To construct a strong-moderate attack, we first randomly select a manipulation profile, denoted by MP_1, from attack data subset AS_{str}. Recall that the manipulation profile fully describes how dishonest ratings are given to all items, but the attack only targets item v_0. Therefore, we need to modify MP_1 such that item v_1 is also a target item. To achieve this, we randomly select another manipulation profile, denoted by MP_2, from AS_{mod}. For all dishonest ratings that are given to item v_0 in MP_2, we modify their target (i.e. the ID of the item that receives these ratings) from item v_0 to item v_1. For all other ratings that are not given to item v_0, we keep them unchanged. These modified ratings are added to MP_1. After such modification, item v_0 and v_1 are both under attack in the modified MP_1.

Finally, we adjust the malicious user IDs of the dishonest ratings for v_1 such that $r\%$ of malicious users that attack item v_0 also provide ratings to item v_1. The typical $r\%$ value is chosen as 30%. For example, in the strong-moderate attack scenario, $30 \times 0.3 = 9$ malicious users attack both items. Notice that the above adjustment and modification will not hurt the randomness of the constructed multiple-target attack files. This is because (1) manipulation profiles MP_1 and MP_2 are randomly chosen, and (2) ratings of each malicious user are randomly assigned by participants according to their personal preferences. Therefore, the constructed multiple-target attack profiles can still represent the diverse attack scenarios.

As a summary, we carefully construct the testing data set based on real user attack data to make it as realistic as possible. (1) By selecting one item similar to item v_0 as an additional target item, it is reasonable to assume that malicious users play similar strategies in attacking this additional target item. (2) By randomizing attack profiles, the testing data set can cover diverse attack strategies. Therefore, this testing data is much closer to reality than simulated attack data, which rarely represents real human users' diverse attack behavior.

B. Performance Measurement

The reputation defense schemes are evaluated from several angles.

- The defense scheme should detect malicious users accurately. We use DR_{user} and FA_{user} to represent the detection rate and false alarm rate of malicious user detection, respectively.

We also need to know whether the undetected malicious users cause large distortion to the final reputation scores. Therefore, we introduce *Recovered Reputation Offset* (RRO). The RRO of item i_x is defined as the fair reputation of i_x (i.e. the mean of normal ratings) minus the recovered reputation score of i_x (i.e. the mean of remaining ratings after removing detected dishonest ratings). Then, we define two additional performance measure.

- RRO_v is defined as the RRO value averaged over all target items.
- Distribution of RRO values of all items, which can show whether the reputation scores of normal items are affected by the defense scheme.

As a summary, the defense schemes will be evaluated by their capability of detecting anomaly, recovering reputation scores, and minimizing their interference to normal items.

C. Experiments and Results

In this subsection, we evaluate the performance of the proposed scheme and compare it with two other representative schemes. The experiments are done under the 4 typical attack scenarios listed in Table II. All the experiments are repeated 1000 times and the average results are shown.

1) **Advantage of Heterogeneous Thresholds:** One of the innovative components of the proposed scheme is to use heterogeneous thresholds for suspicious item detection. We show the advantage of this component by comparing

- proposed scheme: determining items in I_{sus} using the proposed threshold $Th(c) = ac + b_0 + b_s$
- scheme Flat: determining items in I_{sus} using a fixed threshold $Th'(c) = b'_s$.

The performance criteria are defined as

$$DR_{sus} = \frac{\text{\# of target items in } I_{sus}}{\text{total \# of target items}},$$

$$FA_{sus} = \frac{\text{\# of normal items in } I_{sus}}{\text{total \# of normal items}}.$$

Then, ROC curves (DR_{sus} vs. FA_{sus}) are constructed for the proposed scheme by varying detection threshold offset b_s and are constructed for scheme Flat by varying parameter b'_s. Figure 4 shows the ROC curves for the four typical attack scenarios. We observe that both schemes work well in the strong-strong attack scenario, whereas the proposed scheme works better than scheme Flat in all other three attack scenarios.

When the attacks against both target items are strong, the variation in the ratings of target items is much higher than that of normal items. In this "easy" case, flat threshold works as good as the heterogeneous thresholds. However, if a target item is under moderate/weak attack, the variation in its ratings can be similar or even smaller than that of some normal items. In this case, to achieve the same detection rate, scheme Flat has to introduce more false alarms. Therefore,

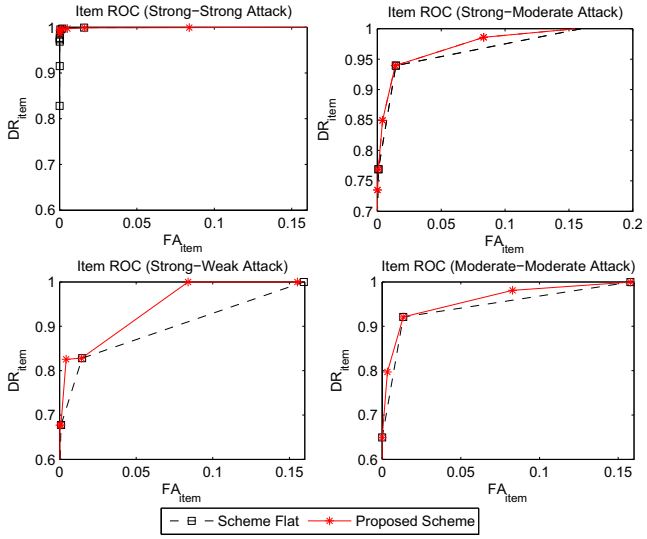

Fig. 4: Comparison between heterogeneous threshold and flat threshold in terms of target item detection.

the proposed scheme has obvious advantage in the strong-weak attack scenario, and noticeable advantage in strong-moderate and moderate-moderate attack scenarios. Obviously, the proposed heterogeneous threshold is more suitable for addressing diverse attack scenarios.

2) **Comparison Schemes:** To demonstrate the overall performance, we compare the proposed scheme with

- scheme Beta: beta-function based scheme [10].
- scheme IR: iterative refinement scheme [11].

In scheme Beta, the item reputation is computed as expectation of all its ratings which are modeled as a beta distribution. A user is determined as a malicious user if the reputation score of any item rated by him/her lies outside q and $(1-q)$ quantile of his/her underlying rating distribution. Here q is the sensitivity parameter of scheme Beta. A smaller q leads to a lower detection rate and a lower false alarm rate, while a larger q leads to a higher detection rate and a higher false alarm rate.

In scheme IR, the estimated quality of each item is computed as weighted average of all ratings. Initially all of the ratings given to an item are assigned with equal weights. Then a user's judging power is estimated. A user having rating values closer to items' estimated qualities is considered as a user with larger judging power. In the next iteration, the weight assigned to each rating is computed as $w_i = J_i^{\beta}$, where w_i is the weight for user i, J_i is the judging power of user i, and β is the key parameter of scheme IR. As β goes up, weights of ratings provided by users with large judging power increase. In the following rounds, the judging power, weights, and item reputation are continuously updated until the reputation score converges.

Scheme Beta and scheme IR represent typical strategies in anomaly detection/reputation recovery for online reputation systems.

3) **Malicious User Detection:** In this subsection, we compare the malicious user detection ROC of the proposed scheme and that of scheme Beta. The ROC of the proposed scheme is obtained by changing the detection threshold offset b_s, and

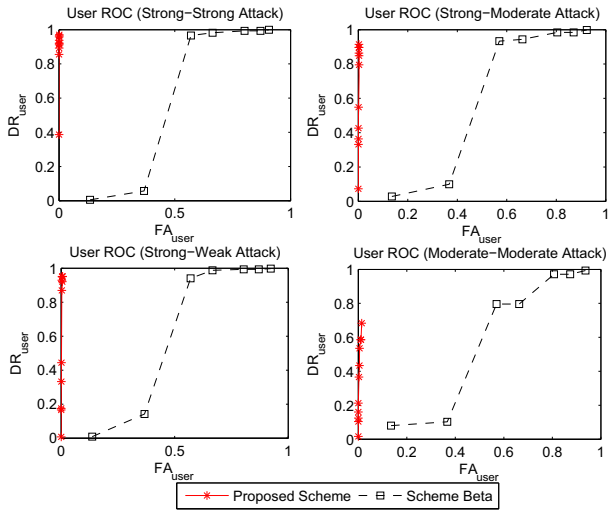

Fig. 5: Comparison between the proposed scheme and scheme Beta, in terms of malicious user detection.

the ROC of scheme Beta is obtained by varying the parameter q.

We would like to point out that the scheme IR can hardly detect malicious users. After the reputation score converges, only a few users have high weights, and the majority of users have very small weights. Thus, scheme IR cannot be used to differentiate honest users and malicious users by using weights, and is not included in this comparison.

As shown in Figure 5, the advantage of the proposed scheme is significant. For example, with the false alarm rate 0.0036, the proposed scheme achieves the detection rate of 1.0 in strong-strong attacks, 0.9023 in strong-moderate attacks, 0.9350 in strong-weak attacks. In moderate-moderate attacks, the highest detection rate is only 0.7224, because some malicious users provide dishonest ratings very close to normal ratings and are hard to detect. The scheme Beta does not have good performance because of the features of real user attack data. The attack data was generated by real human users, who were smart enough to create dishonest ratings that are not too far away from the majority's opinion. Scheme Beta, as well as other defense schemes that focus only on examining statistics of the rating values, cannot effectively detect such dishonest ratings without suffering a high false alarm rate.

4) **Recovered Reputation Offset of Target Items:** In this subsection, we first examine RRO_v (i.e. average RRO of target items) of all three schemes. Figure 6 shows the RRO_v of the proposed scheme, scheme Beta and scheme IR for different parameter selections. Each subplot demonstrates RRO_v for one scheme. One subplot contains 4 curves, each of which is for one attack scenario. The x-axis is the key parameter of the scheme. For the proposed scheme (upper subplot), parameter b_s, which affects the heterogeneous threshold value $Th(c)$, ranges from 0 to 10. For scheme Beta (middle subplot), parameter q ranges from 0.1 to 0.4. For scheme IR (lower subplot), parameter β ranges from 0 to 1.

From Figure 6 we can find the best achievable RRO_v, referred to as the *minimal* RRO_v, by varying those parameters. We make the following observations.

- Among all three schemes, the proposed scheme can achieve the smallest RRO_v in most attack scenarios. These RRO_v values are between 0.01 and 0.04, which is very low. The fair reputation of target items in the experiments is around 4. That is, with the proposed scheme, the undetected malicious users can mislead the reputation score by $< 1\%$.
- The best achievable RRO_v for scheme Beta and scheme IR is also very low. However, their RRO_v values are sensitive to parameter selection. For example, for slightly different q value (e.g. q=0.175, q=0.15, in strong-strong attack), the RRO_v of scheme Beta can change from 0.5839 to 0.0289.
- The RRO_v values of scheme IR and scheme Beta are sensitive to different attack types. They may not perform consistently in different attack scenarios, especially when the optimal parameters are difficult to estimate in practice.
- The RRO_v of the proposed scheme is stable for either different parameter settings or different attack scenarios. Even if parameter b_s is not optimal, the RRO_v will not increase dramatically.

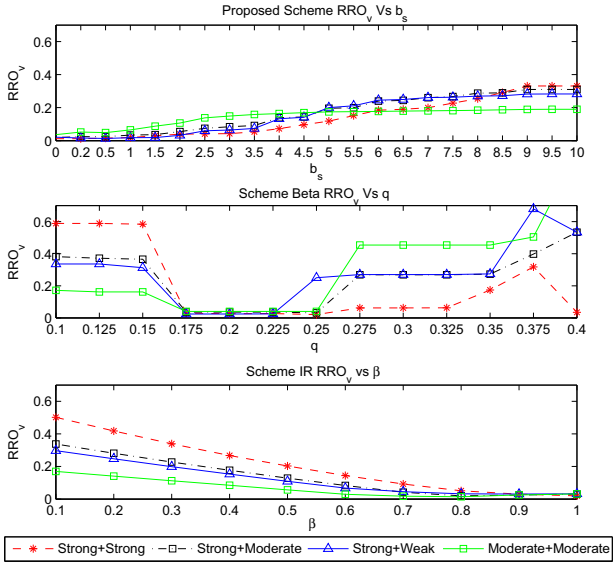

Fig. 6: Comparison among the proposed scheme, scheme Beta, and scheme IR in terms of RRO_v.

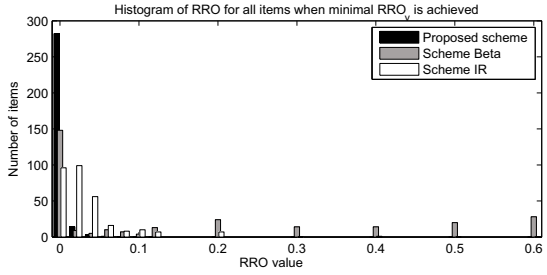

Fig. 7: Histogram of RRO for all items.

5) Recovered Reputation Offset of All Items: In this subsection, we examine whether the normal items are affected by the defense schemes.

Figure 7 shows the histogram of RRO value of all items when the key parameters in three schemes are chosen to minimize RRO_v (i.e. $b_s = 0$, $q = 0.25$ and $\beta = 0.8$), in

	Measured by	The proposed scheme
Detection of malicious users	ROC (Fig. 5)	is overwhelmingly better
Reputation recovery of target items	RRO_v (Fig.6)	achieves small minimal RRO_v in most scenarios; much less sensitive to parameter and attack type change
Interference to normal items	RRO histogram (Fig. 7)	causes much less interference to

TABLE III: Comparison result summary

the strong-moderate attack scenario. The following important observations are made.

- With the proposed scheme, more than 275 items have RRO values close to 0, and 99.67% of items have RRO value below 0.05. This means that the proposed scheme removes dishonest ratings (which lead to low RRO of target items), but do not remove honest ratings of normal items (which leads to low RRO of normal items).
- The other two schemes, however, have much worse performance. Many normal items have non-negligible RRO values. In scheme Beta, there are only 54% of items have RRO value lower than 0.05. In scheme IR, this number is 83.7%. The reputation scores of many normal items are affected because the defense schemes wrongly remove honest ratings.

Finally, we summarize the comparison results in Table III.

V. Discussion

The computation complexity of the proposed scheme mainly comes from (1) change detection and (2) item correlation analysis. The change detection, which checks all ratings in the system, only requires very simple calculation. The amount of calculation is similar to computing the reputation scores of items. Recall that the reputation system needs to go through all ratings and calculate the reputation scores of all items, even without any defense scheme. The item correlation analysis is only performed for items in the suspicious item set, which contains a small number of items. The amount of computation is not significant as long as the majority of items in the system are not under attack. Therefore, the proposed scheme is expected to have lower computation complexity than other defense schemes (e.g. scheme Beta and IR) that perform statistical analysis or even iterative computation for all items in the system.

VI. Conclusion

In this paper, we designed an anomaly detection scheme for online reputation systems. The propose scheme relies on the integration of several components: time-domain change detection, system-level visualization, heterogeneous threshold selection, and item correlation analysis. The proposed scheme represents a new philosophy of anomaly detection in online reputation systems, and can address both multiple-target and single-target attacks. Compared with scheme Beta and IR, which do not consider heterogeneity/correlation of items, the proposed scheme has significantly better performance in terms of detecting malicious users and reducing impacts on normal items. The proposed scheme also yields more stable performance in different parameter settings and attack scenarios.

Acknowledgment

This work is partially supported by NSF award #0643532 and #0931820.

References

[1] Rainie Lee and Hitlin Paul. *Use of Online Rating Systems*. http://www.pewinternet.org/Reports/2004/Use-of-Online-Rating-Systems.aspx.

[2] comScore. http://ir.comscore.com/releasedetail.cfm?ReleaseID=539354.

[3] J. Brown and J. Morgan. Reputation in online auctions: The market for trust. *California Management Review*, 49(1):61–81, 2006.

[4] A. Harmon. *Amazon glitch unmasks war of reviewers*. The New York Times, February 14,2004.

[5] Riyad Kalla. *Apparently it is easy to game amazons reviews*. http://www.thebuzzmedia.com/apparently-it-is-easy-to-game-amazons-reviews/. The Buzz Media.

[6] Riyad Kalla. *IMDB, Whats Going on With Your Scores?* http://www.thebuzzmedia.com/imdb-whats-going-on-with-your-scores/. The Buzz Media.

[7] *Taobao fights reputation spam in e-business boom*, BeijingToday, Sept. 12 2009. http://www.beijingtoday.com.cn/feature/taobao-fights-reputation-spam-in-e-business-boom.

[8] A. Whitby, A. Jøsang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proc. of the 7th Int. Workshop on Trust in Agent Societies*, 2004.

[9] J. Weng, C. Miao, and A. Goh. An entropy-based approach to protecting rating systems from unfair testimonies. *IEICE TRANSACTIONS on Information and Systems*, E89–D(9):2502–2511, Sept 2006.

[10] A. Josang and R. Ismail. The beta reputation system. In *Proc. of the 15th Bled Electronic Commerce Conference*, 2002.

[11] P. Laureti, L. Moret, Y.-C. Zhang, and Y.-K. Yu. Information filtering via iterative refinement. In *Europhysics Letters*, volume 75, pages 1006–1012, 2006.

[12] B. Yu and M. Singh. An evidential model of distributed reputation management. In *Proc. of the Joint Int. Conference on Autonomous Agents and Multiagent Systems*, pp. 294-301 2002.

[13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of the 12th Int. conference on World Wide Web*, May 2003.

[14] Y. Yang, Q. Feng, Y. Sun, and Y. Dai. Reputation trap: An powerful attack on reputation system of file sharing p2p environment. In *the 4th International Conference on Security and Privacy in Communication Networks*, 2008.

[15] Y. Liu and Y. Sun. Anomaly detection in feedback-based reputation systems through temporal and correlation analysis. In *Proc. of 2nd IEEE Int. Conference on Social Computing*, Aug 2010.

[16] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

[17] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies,architectures, and protocols for computer communications*, pp. 267-278, 2006.

[18] A. Joang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, Mar 2007.

[19] J. Sabater and C. Sierra. Social regret, a reputation model based on social relations. *SIGecom Exchanges*, 3(1):44–56, 2002.

[20] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of the 7th international conference on World Wide Web (WWW)*, 1998. http://dbpubs.stanford.edu:8090/pub/1998-8.

[21] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proc. of the 2nd ACM conference on Electronic commerce*, 2000.

[22] *CANT Cyber Competition*. http://www.ele.uri.edu/nest/cant.html.

[23] P. Macnaughton-smith, W. T. Williams, M. B. Dale, and L. G. Mockett. Dissimilarity analysis: A new technique of hierarchical sub-division. *Nature*, pages 1034–1035, 1964.