Securing Rating Aggregation Systems using Statistical Detectors and Trust

Yafei Yang, *Member, IEEE,* Yan Sun, *Member, IEEE,* Steven Kay, *Fellow, IEEE,* and Qing Yang, *Member, IEEE*

***Abstract*—Online feedback-based rating systems are gaining popularity. Dealing with unfair ratings in such systems has been recognized as an important but difficult problem. This problem is challenging especially when the number of regular ratings is relatively small and unfair ratings can contribute to a significant portion of the overall ratings. Furthermore, the lack of unfair rating data from real human users is another obstacle toward realistic evaluation of defense mechanisms. In this paper, we propose a set of statistical methods to jointly detect collaborative unfair ratings in product-rating type online rating systems. Based on detection, a framework of trust-assisted rating aggregation system is developed. Furthermore, we collect unfair rating data from real human users through a Rating Challenge. The proposed system is evaluated through simulations as well as experiments using real attack data. Compared with existing schemes, the proposed system can significantly reduce negative impact from unfair ratings.**

***Index Terms*—unfair ratings, detector, trust.**

I. Introduction

The Internet, the revolutionary mass communication media, has enabled individuals to make their personal opinions accessible to the global community at almost no cost [1]. One important type of information sharing is through online opinion forums and rating systems, such as Epinions [2] and Amazon product rating. In such systems, users submit their opinions regarding products, services or other users. Then, the submitted opinions are analyzed, aggregated, and made publicly available. While feedback-based rating systems are having increasing influence on today's consumers, ensuring reliability and robustness of such systems remains as an important and challenging task [3].

Reliability of online rating systems largely depends on whether *unfair ratings* and *dishonest raters* can be detected and removed. Unfair ratings, which can be generated either intentionally or unintentionally, do not reflect the property of the items being rated. Dishonest raters are the users who intentionally provide unfair ratings. There have been many approaches proposed to deal with unfair ratings. Examples include clustering techniques [4], statistical analysis [5], endorsement-based quality estimation [3], and entropy-based detection [6]. Based on unfair rating detection, schemes are proposed to detect dishonest raters [3], [7]. The details of those approaches will be reviewed in Section II.

Most existing systems share one common property: *majority rule*. That is, to detect and remove the ratings that are far away from the majority's opinion. However, this philosophy has two limitations. *First*, the attacker can recruit dishonest raters who collaboratively manipulate ratings for one or several items. In many practical scenarios, the number of fair ratings is limited. This occurs when the items (products or services) are new or unpopular. This also occurs when rating aggregation is performed within a time window, in order to catch the dynamic change of the items. At Amazon and Epinion, it is common that a product has only a few (e.g. < 50) reviews/ratings and even fewer recent reviews/ratings. Therefore, unfair ratings may overweight fair ratings and become majority in some time intervals. *Second*, smart dishonest raters can introduce a relatively small bias and make the unfair ratings difficult to detect. For example, when there are five rating levels, as in Amazon, the dishonest raters can rate higher or lower by one level. If there is a sufficient number of dishonest raters, the attacker can make a product look better or worse than its competitors. However, it is extremely difficult for the majority-rule based methods to detect such unfair ratings without suffering a high false alarm rate.

In this paper, we investigate the problem from a new angle. Instead of trying to determine whether some specific ratings are biased, we detect the time intervals in which collaborative unfair ratings are highly likely. We focus on product-rating type applications, in which the normal users' opinion should agree with the quality of the item. The attackers insert unfair ratings in certain time intervals to mislead the system's opinion on the quality of the item. The rating values are treated as samples of a random process. Signal processing techniques are used to detect mean change, histogram change, arrival rate change in the rating values, and to perform signal modeling. Based on the signal processing techniques, we design and implement four detectors. In order to evaluate the effectiveness of the new detectors in real world, we launch a *Rating Challenge* to collect attack data from real human users. After analyzing the attack data and evaluating the performance of individual detectors, we develop a method to jointly apply different detectors. Furthermore, from the detection results, trust in raters is established. This trust information is applied to a trust-assisted rating aggregation algorithm and can assist future suspicious rating detection. Experiments show that the proposed scheme can detect smart

All the authors are with the Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI, 02881 USA e-mail: {yafei, yansun, kay, qyang}ele.uri.edu.

Manuscript received October 15, 2007; revised July 5, 2009.

and collaborative unfair ratings created by real human users. A significant reduction in the influence from unfair ratings is observed. As a summary, our contributions include

- identifying various types of unfair ratings in online rating systems through real-user experiments;
- developing statistical methods to detect suspicious ratings in the scenarios where the majority rule does not necessarily hold and/or unfair ratings have moderate or small bias;
- developing a framework that integrates trust establishment and rating aggregation.

The rest of the paper is organized as follows. Related work and attack models are discussed in Section II. An overview of the proposed system is presented in Section III, and the algorithms are described in Section IV. Simulation and experimental results are presented in Section V, followed by the conclusion in Section VI.

II. Related Work and Attack Models

A. Related Research

In the current literature, the unfair rating problem is often addressed from two angles: improving rating system to reduce bias and helping users to manage rating bias [8]. In the first type, the existing work can be roughly divided into several categories: (1) data mining and statistical analysis approaches that identify unusual rating behaviors [1], [3]–[5], [7]; (2) evaluating trust in users that reflects whether the users' rating behaviors agree with majority or follow normal pattern [3], [9]–[11]; (3) reducing randomness in rating by formalizing rating submission procedure, such as asking users a list of questions before rating submission; (4) increasing difficulty in registering new user IDs to prevent the attackers from switching identity or controlling a large number of user IDs [12], [13]; and (5) other design improvements including incorporating expert reviews, controlling anonymity of raters, and punishing violators. Interested readers can find more details in the survey paper [8] and [14]. Since our work belongs to the first and the second category, we will briefly discuss the second category and then review representative schemes in the first category.

Although it is possible to use a rater's social standing and interaction habit to determine its levels, such information is often not available. We focus on determining trust levels based on the outcome of unfair rating detection. Currently, after unfair rating detection, simple trust models are used to calculate trust in raters, such as these in [3], [10], [11]. The accuracy and effectiveness of trust evaluation largely depend on the accuracy of the detection algorithms.

Unfair rating detection can be conducted from various perspectives. For example, in [4], unfair ratings and honest ratings are separated through clustering technique. In [3], the quality of individual ratings is evaluated by an endorsement method. Particularly, a rater gives high endorsement to other raters who provide similar ratings and low endorsement to the raters who provide different ratings. The quality of a rating is the summation of the endorsements from all other raters. The unfair ratings are expected to have low quality. In [5], a statistical filtering technique based on Beta-function is presented. The ratings that are outside the q quantile and $(1-q)$ quantile of the majority opinion are identified as unfair ratings, where q is a parameter describing the sensitivity of the algorithm. In [7], if a new rating leads to a significant change in uncertainty in rating distribution, this rating is considered as unfair. All these schemes use a hidden principle, called the majority rule. The advantage is that they can detect unfair ratings, as long as the unfair rating values are far away from the majority's opinion. However, their effectiveness is greatly reduced when the smart collaborative attackers insert unfair ratings that are close to the honest ratings. The proposed scheme in this paper does not have this limitation because it detects unfair ratings using a different principle.

All above methods require that a certain number of ratings is available. In other words, when there are only a few ratings for a product, it is impossible to detect unfair ratings unless the system considers rating behavior over many products, such as investigating similarity among user rating habits, which is beyond the scope of this paper.

B. Attack Models

Inspired by the work in [4], we classify the unfair ratings into two categories.

- *Individual unfair ratings*: an individual rater provides unfairly high or low ratings without collaborating with other raters. This type of ratings often result from raters' personal preference (i.e. dispositional trust [15]), irresponsibleness, and randomness.
- *Collaborative unfair ratings*: a group of raters provide unfairly high or low ratings to boost or downgrade the overall ratings of an item. This type of rating is often due to strategic manipulation [16].

Compared with collaborative unfair ratings, individual unfair ratings are less harmful and can be handled by the existing approaches. Therefore, *our focus is to deal with collaborative unfair ratings.*

In order to investigate dishonest rating behaviors of real human users, we launched a **Rating Challenge** [17], in which participants insert collaborative unfair ratings into a regular rating data set. The participants who mislead the aggregated rating scores the most can win a cash prize. More details of the Rating Challenge will be described in Section V-A.

Unfair ratings can be considered as a random process, with a mean and variance. The difference between the mean of all unfair ratings and the mean of the fair ratings is called *bias*. Based on the data collected from real human users, we observed that there are four meaningful selections of bias and variance of unfair ratings from the attacker's point of view.

- Type 1: large bias and small variance
- Type 2: small bias and small variance
- Type 3: moderate bias and moderate variance
- Type 4: zero bias and large variance

The goal of types 1, 2, and 3 is to boost or downgrade the aggregated rating values. The goal of type 4 is to make the overall rating have a large variation (or variance), which will make the consumers have less confidence in the aggregated rating scores and feel riskier when making purchasing decisions. Type 4 is very special. It targets psychology more than technology. In the rest of this paper, we will focus on type 1-3.

The *cost* of attack largely depends on the number of malicious users, i.e. the number of user IDs under the attacker's control. Recall that one user ID cannot provide multiple ratings to the same product. The attacker needs to acquire many user IDs. This behavior is referred to as the sybil attack [12]. The existing defense against the sybil attack can largely increase the cost of registering/controlling fake user IDs [12]. Thus, the attacker with resource constraint cannot have an unlimited number of user IDs.

Given a fixed number of user IDs, the attacker has two choices: inserting unfair ratings within very long time duration (e.g. one year) or within a relatively short time interval (e.g. a few weeks to a few months). The first choice introduces a smaller bias in the final rating score for a longer time, whereas the second choice introduces a bigger bias for a shorter time. We argue that the ultimate goal of boosting/downgrading is to make the target product look better or worse than other competing products. If the attacker only introduces a small bias, he/she is unlikely to change the relative ranking of comparable products. Even if he/she can maintain this small bias for a long time, the consequence of the attack (e.g. volume) may not be obvious. Therefore, a more effective way is to introduce large enough bias that changes the relative ranking. In this paper, we assume the attacker makes the second choice. Of course, if the attacker is really powerful, he/she can introduce a large bias during the entire lifetime of the product. We will not study this extreme case in this paper.

III. Trust-enhanced Rating Aggregation System Design

The overall design of the trustworthy rating analysis and aggregation system is shown in Figure 1. The two major components are the *Rating Aggregator* and the *Trust Manager*. In this section, we first describe the system architecture and then discuss design challenges.

A. Rating Aggregation Overview

The rating aggregation process contains four steps.

First, raw ratings are analyzed. Four analysis methods, arrival rate detection, model change detection, histogram detection and mean change detection, are applied independently. The philosophies behind these detectors are as follows.

- Since the primary goal of the attacker is to boost or reduce the aggregated rating score that is closely related to the mean of rating values, a *mean change detector* is developed to detect sudden changes in the mean of rating values.
- When the attackers insert unfair ratings, they may cause an increase in the rating arrival rate. Thus, the *arrival rate detector* is designed to detect sudden increase in the number of raters per time unit.
- A large number of unfair ratings can result in a change in the histogram of overall rating values, especially when the difference between unfair and fair ratings is large. Thus, the *histogram detector* is used.
- The fair ratings can be viewed as a random *noise*. In some attacks, the unfair ratings can be viewed as a *signal*. Thus, signal modeling technique is used to detect whether a signal (i.e. unfair ratings) is presented. This method can be viewed from another angle. When the unfair ratings are presented, the power spectrum density may change from white to colored. Therefore, a *model change detector* is designed to capture this change.

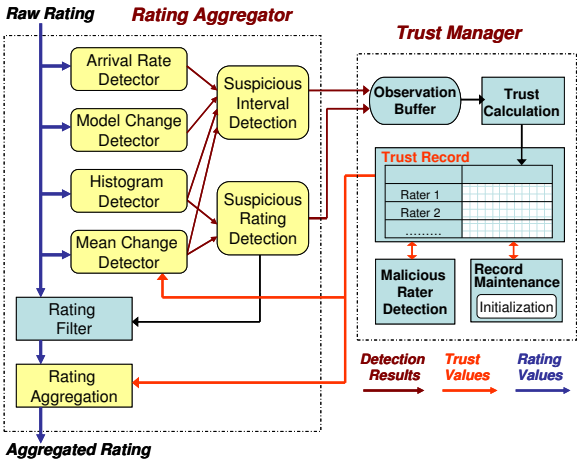

Fig. 1. Block diagram of the trust-enhanced rating aggregation system.

Second, outcomes of the above four detectors are combined to detect the time intervals in which unfair ratings are highly likely. Additionally, the suspicious rating detection module can mark some specific ratings as suspicious.

Third, the trust manager uses the output of the suspicious interval detection and the suspicious rating detection to determine how much individual raters can be trusted.

Fourth, the highly suspicious ratings are removed from the raw ratings by a rating filter. Then, the ratings are combined by the rating aggregation algorithm.

B. Trust Manager Overview

Before discussing the trust manager, we introduce the relationship between trust establishment and rating aggregation.

A *trust relationship* is always established between two parties for a specific action. That is, one party trusts the other party to perform an action. The first party is referred to as the *subject* and the second party as the *agent*. A notation $\{subject : agent, action\}$ is used to represent the trust relationship. For each trust relationship, one or multiple numerical values, referred to as *trust values*, describe the level of trustworthiness. In the context of rating aggregation,

- the *rating values* provided by the raters is the trust value of $\{rater : object,$ having certain quality$\}$;

- the *trust in raters* calculated by the system is the trust value of $\{system : rater,$ providing honest rating$\}$;
- the *aggregated rating* (i.e. the overall rating score) is the trust value of $\{system : object,$ having certain quality$\}$.

When the subject can directly observe the agent's behavior, *direct trust* can be established. Trust can also transit through third parties. For example, if A and B have established a recommendation trust relationship and B and C have established a direct trust relationship, then A can trust C to a certain degree if B tells A its trust opinion (i.e. recommendation) about C. Of course, A can receive recommendation about C from multiple parties. This phenomenon is called *trust propagation*. *Indirect trust* is established through trust propagations. The ways to calculate the indirect trust are often called *trust models*.

In the context of rating aggregation, the system, the raters, and the object obviously form trust propagation paths. Thus, the aggregated rating, i.e. the indirect trust between the system and the object, can be calculated using trust models. *One important observation is that the calculation in rating aggregation can be determined or inspired by existing trust models.*

Based on this observation, we design a trust manager by simplifying the generic framework of trust establishment proposed in [18]. As illustrated in Figure 1, this design contains the *observation buffer* that collects observations on whether specific ratings or time intervals are detected as suspicious, the *trust calculation* module and *trust record* that compute and store trust values of raters, and the *malicious rater detection* module that determines how to handle the raters with low trust values.

C. Design Challenges

1) : The **first challenge** is to design detection methods. The trust manager determines how much a rater can be trusted based on observations. However, obtaining the observations, or in other words extracting features from the raw rating data is challenging. A very popular trust calculation method is the Beta-function based trust model proposed in [19]. In this method, trust value is calculated as $\frac{S+1}{S+F+2}$, where S denotes the number of previous successful actions and F denotes the number of previous failed actions. This method has been used in various applications [18], [20], [21]. For rating aggregation, however, it is difficult to determine S and F values.

Assume that we are examining trust in rater i. In this case, S should be the number of honest ratings provided by i, and F should be the number of dishonest ratings provided by i. However, there is no way of perfectly monitoring rater i's past behavior, and we must estimate S and F values through some detection methods. To make things even more difficult, one must consider the fact that the rating values are highly discrete and the number of ratings can be small (e.g. < 100) in practical applications.

2) : The attack behaviors against rating systems can be very complicated. Several detection strategies must be used simultaneously. The **second challenge** is to understand the effectiveness of each detector against different attacks and then integrate multiple detectors.

It is important to point out that the proposed methods can detect collaborative malicious ratings that cause sudden change in rating value statistics. They cannot address the following two attack scenarios. First, the malicious users insert unfair ratings from the beginning to the end of the lifetime of the target product. Second, the malicious users introduce small but steady change gradually such that no sudden change occurs but the accumulated change can be large.

These two types of attacks can be detected by neither the proposed methods nor other existing methods. However, the attacker must invest significant more resource to launch these attacks. Obviously, the attacker must maintain the rating bias for a very long time. Since ratings for one product must be from different user IDs, the attacker needs to control a large number of user IDs. Furthermore, these user IDs cannot be reused to attack other products since we evaluate the trust in raters. As a summary, when the proposed defense methods are used, the attacker will have to spend much more resource in order to achieve the desired goal, compared to the case without the proposed defense. In our Rating Challenge, we observe the attacks that cannot be detected by the proposed scheme. These attacks, however, do not introduce large damage to the overall rating, since each participant of the Rating Challenge has a constraint on attack resource (e.g. the number of malicious user IDs).

IV. Algorithm Description

A. Mean Change Detector

The mean change detector contains three parts.

1) Mean Change Hypothesis Test: For one product, let $t(n)$ denote the time when a particular rating is given, $x(n)$ denote the value of the rating, and $u(n)$ denote the IDs of the rater. That is, at time $t(j)$, rater $u(j)$ submits a rating for the product with rating value $x(j)$, where $j = 1, 2, \cdots, N$ and N is the total number of ratings for this product.

We first study the mean change detection problem inside a window. Assume that the window contains $2W$ ratings. Let X_1 denote the first half ratings in the window and X_2 denote the second half ratings in the window. We model X_1 as an i.i.d Gaussian random process with mean A_1 and variance σ^2, and X_2 as an i.i.d Gaussian random process with mean A_2 and variance σ^2. Then, to detect the mean change is to solve the hypothesis testing problem

$$\mathcal{H}_0 : A_1 = A_2$$
$$\mathcal{H}_1 : A_1 \neq A_2.$$

It has been shown in [22] that the Generalized Likelihood Ratio Test (GLRT) is

$$\text{Decide } \mathcal{H}_1 \text{ (i.e. there is a mean change), if}$$
$$2 \ln L_G(x) = \frac{W(\hat{A}_1 - \hat{A}_2)^2}{2\sigma^2} > \gamma \quad (1)$$

where $\hat{A}_1$ is the average of X_1 and $\hat{A}_2$ is the average of X_2, and γ is a threshold.

2) Mean Change Indicator Curve: Second, the detector constructs the mean change indicator curve using a sliding window. The size of the sliding window (W), should be carefully chosen. It cannot be too small. Otherwise, the test in (1) will have poor performance, i.e. low detection rate and/or high false alarm rate. It cannot be too large because the test can only handle one mean change in the window.

Based on (1), the mean change indicator curve is constructed as $MC(k)$ versus $t(k)$, where $MC(k)$ is the value of $W(\hat{A}_1 - \hat{A}_2)^2$ calculated for the window containing ratings $\{x(k-W), \cdots, x(k+W-1)\}$. In other words, the test in (1) is performed to see whether there is a mean change at the center of the window. Here, the value k should satisfy $W < k \leq N - W + 1$.

In addition, $MC(k)$ can also be calculated for $k \leq W$ and $k > N - W + 1$, by using a smaller window size. For example, when $k \leq W$, $MC(k)$ can be calculated in the window $\{x(1), \cdots, x(2k-2)\}$ with window size $(2k-2)$. Of course, the window must contain a minimum number of data points to ensure effectiveness of the test.

Examples of mean change indictor curve are shown in Section IV-E, Figure 2 and Figure 4. The top plot shows the rating data $x(n)$ vs. $t(n)$. The blue dots represent the rating values for a flat panel TV (the first data set) in the Rating Challenge [17], the red circles represent the unfair ratings added by simulation. On the MC curves (the 2nd plots), two peaks clearly show the beginning and the end of the attack.

3) MC Suspiciousness: Based on the peak values on the mean change indicator curve, we detect the time interval in which abnormal mean change occurs. This interval is called *mean change (MC) suspicious interval*.

When there are only two peaks, the MC suspicious interval is just between the two peaks. When there are more than 2 peaks, it is not straightforward to determine which time interval is suspicious. We use trust information to solve this problem. In particular, we divide all ratings into several segments, separated by the peaks on the mean change indicator curve. Assume there are M segments. In each segment, the mean value of ratings are calculated as B_j for $j = 1, 2, \cdots, M$. And B_{avg} is the mean value of the overall ratings. A segment j is marked as *MC suspicious* if either of the following conditions is satisfied:

1) $|B_j - B_{avg}| > threshold_1$. That is, there is a very large mean change.
2) $|B_j - B_{avg}| > threshold_2$ and T_j/T_{avg} is smaller than a threshold, where T_j is the average trust value of the raters in the j^{th} segment, T_{avg} is the average trust value of the raters in all segments. Here, $threshold_2 < threshold_1$. This condition says that there is a moderate mean change and the raters in the segment is less trustworthy.

B. Arrival Rate Change Detector

1) Arrival Rate Change Hypothesis Test: For one product, let $y(n)$ denote the number of ratings received on day n. We first study the arrival rate detection problem inside a window. Assume that the window covers $2D$ days, starting from day k. We want to detect whether there is an arrival rate change at day k', for $k < k' < k + 2D - 1$.

Let $Y_1 = [y(k), y(k+1), \cdots, y(k'-1)]$ and $Y_2 = [y(k'), y(k'+1), \cdots, y(k+2D-1)]$. It is assumed that $y(n)$ follow Poisson distribution. Then, the joint distribution of Y_1 and Y_2 is

$$p[Y_1, Y_2; \lambda_1, \lambda_2] = \prod_{j=k}^{k'-1} \frac{e^{-\lambda_1}\lambda_1^{y(j)}}{y(j)!} \prod_{j=k'}^{k+2D-1} \frac{e^{-\lambda_2}\lambda_2^{y(j)}}{y(j)!}, \quad (2)$$

where λ_1 is the arrival rate per day from day k to day $k'-1$, and λ_2 is the arrival rate per day from day k' to day $k+2D-1$. To detect the arrival rate change is to solve to the hypothesis testing problem

$$\mathcal{H}_0 : \lambda_1 = \lambda_2$$
$$\mathcal{H}_1 : \lambda_2 \neq \lambda_1$$

It is easy to show that

$$p[Y_1, Y_2; \lambda_1, \lambda_2] = \frac{e^{-a\lambda_1}\lambda_1^{a\bar{Y_1}}}{\prod_{j=k}^{k'-1} y(j)!} \cdot \frac{e^{-b\lambda_2}\lambda_2^{b\bar{Y_2}}}{\prod_{j=k'}^{k+2D-1} y(j)!}. \quad (3)$$

where

$$\bar{Y_1} = \frac{1}{a}\sum_{j=k}^{k'-1} y(j), \quad \bar{Y_2} = \frac{1}{b}\sum_{j=k'}^{k+2D-1} y(j),$$
$$a = k' - k, \qquad b = k - k' + 2D.$$

A GLRT decides $\mathcal{H}_1$ if

$$\frac{p[Y_1, Y_2; \hat{\lambda_1}, \hat{\lambda_2}]}{p[Y_1, Y_2; \hat{\lambda}, \hat{\lambda}]} > \gamma, \quad (4)$$

where $\hat{\lambda_1} = \bar{Y_1}$, $\hat{\lambda_2} = \bar{Y_2}$, and $\hat{\lambda} = \frac{1}{2D}(\sum_{j=k}^{k+2D-1} y(j)) = \bar{Y}$. Taking logarithm at both sides of (4), we derive

$$\text{Decide } \mathcal{H}_1 \text{(i.e. there is an arrival rate change) if}$$
$$\frac{a}{2D}\bar{Y_1}\ln\bar{Y_1} + \frac{b}{2D}\bar{Y_2}\ln\bar{Y_2} - \bar{Y}\ln\bar{Y} \geq \frac{1}{2D}\ln\gamma. \quad (5)$$

2) Arrival Rate Change Curve: Based on (5), the Arrival Rate Change (ARC) curve is constructed as $ARC(k')$ vs $t(k')$. Here, the k' value is chosen as the center of the sliding window, i.e. $k' = k + D$. When $D < k' < N - D + 1$, $ARC(k')$ is just the left-hand side of equation (5) with $a = b = D$. When $k' \leq D$ or $k' \geq N - D + 1$, $ARC(k')$ can be calculated using a smaller window size, similar as the approach used in Section IV-A2. Examples of the ARC curve are shown in Section IV-E, Figure 2 - 4, with two peaks showing the beginning and the end of the attack.

3) ARC Suspiciousness: Based on the peaks on the ARC curve, we divide all ratings into several segments. If the arrival rate in one segment is higher than the arrival rate in the previous segment and the difference between the arrival rates is larger than a threshold, this segment is marked as *ARC suspicious*.

4) H-ARC and L-ARC: For some practical rating data, the arrival rate of unfair ratings is not very high or the poisson arrival assumption may not hold. For those cases, we design H-ARC, which detects the arrival rate change in high value ratings, and L-ARC, which detects the arrival rate change in low value ratings.

Let $y_h(n)$ denote the number of ratings that are higher than $threshold_a$ received on day n, and $y_l(n)$ denote the number of ratings that are lower than $threshold_b$ received on day n. The $threshold_a$ and $threshold_b$ are determined based on the mean of all ratings.

- H-ARC detector: replace $y(n)$ in the ARC detector by $y_h(n)$
- L-ARC detector: replace $y(n)$ in the ARC detector by $y_l(n)$.

Based on experiments, we have found that H-ARC and L-ARC are more effective than the ARC detector when the arrival rate of unfair ratings is less than 2 times of the arrival rate of fair ratings.

C. Histogram Change Detector

Unfair ratings can change histogram of the rating data. In this paper, we design a histogram change detector based on clustering technique. There are two steps.

- 1. Within a time window k with the center at t_k, constructed two clusters from the rating values using the simple linkage method. The Matlab function $clusterdata()$ is used in the implementation.
- 2. The Histogram Change (HC) curve, $HC(k)$ versus t_k, is calculated as

$$HC(k) = \min\left(\frac{n_1}{n_2}, \frac{n2}{n1}\right), \tag{6}$$

where n_1 and n_2 denote the number of ratings in the first and the second cluster, respectively.

Examples of the HC curve are shown in Section IV-E, Figure 2 and 4. When the attack occurs, the $HC(k)$ increases.

D. Signal Model Change Detector

Let $E(x(n))$ denote the mean of $x(n)$, where $x(n)$ denote the rating values. When there is no collaborative raters, ratings received at different time (also from different raters) should be independent. Thus, $(x(n) - E(x(n)))$ should approximately be a white noise. When there are collaborative raters, $(x(n) - E(x(n)))$ is not white noise any more. Instead, the ratings from collaborative raters can be looked at as a *signal* embedded in the white noise.

Based on the above argument, we develop an unfair rating detector through signal modeling.

- Model-error-based detection: the ratings in a time window are fit onto an autoregressive (AR) signal model. The model error is examined. When the model error is high, $x(n)$ is close to a white noise, i.e. honest ratings. When the model error is small, there is a *signal* presented in $x(n)$ and the probability that there are collaborative raters is high.

	Unfair Rating Type 1	Unfair Rating Type 2	Unfair Rating Type 3
Bias	large	small	moderate
Variance	small	small	moderate
Cost	moderate	high	high
MC effective?	Yes	depends	depends
HC effective?	Yes	No	Yes
ARC effective?	depends	Yes	Yes
H-ARC, L-ARC effective?	Yes	Yes	Yes
ME effective?	No	Yes	No

TABLE I
EFFECTIVENESS OF DETECTORS UNDER DIFFERENT TYPES OF UNFAIR RATINGS

The *model error (ME) curve* is constructed with the vertical axis as the model error, and horizontal axis as the center time of the windows. The windows are constructed either by making them contain the same number of ratings or have the same time duration. The covariance method [23] is used to calculate the AR model coefficients and errors.

Examples of the ME curve are shown in Section IV-E, Figure 3. The curve marked with $*$ is the model error for the original rating data, the curve marked with small dots is the model error when unfair ratings are present. It can be seen that the model error drops when there is an attack. The time interval when the model error drops below a certain threshold is marked as the model error (ME) suspicious interval.

E. Comparison Among Detection Methods

We have presented four different detectors. The outputs of these detectors are intervals marked as MC suspicious, HC suspicious, ARC suspicious, and ME suspicious. No single detector can handle all types of attacks effectively. In this section, we compare these detectors from two perspectives. One is qualitative comparison under different types of unfair ratings and the other is quantitative comparison based on their Receiver Operating Characteristic (ROC) curves.

1) Comparison under different types of unfair ratings: In Table I, we list the features of type 1, 2, 3 unfair ratings and qualitatively compare different detectors. In this table, *cost* means the number of dishonest raters the attacker has to control in order to cause a noticeable change in the aggregated rating value, when the rating aggregation algorithm is simple averaging.

- The performance of the *MC detector* depends on how ambitious the attacker is. If the attacker only wants to cause a small boost or downgrade, this detector will not be very effective.
- The performance of the *HC detectors* depends on how far the unfair ratings are away from the fair ratings. A large bias in the unfair ratings will lead to effective HC detection. Therefore, HC detector is effective against type 1 and type 3 unfair ratings.
- To achieve the same effect, the attackers need to insert more unfair ratings if they use type 2 or type 3 unfair ratings, compared with using type 1 unfair ratings. Thus, the arrival rate of unfair ratings in type 2 and type 3 should

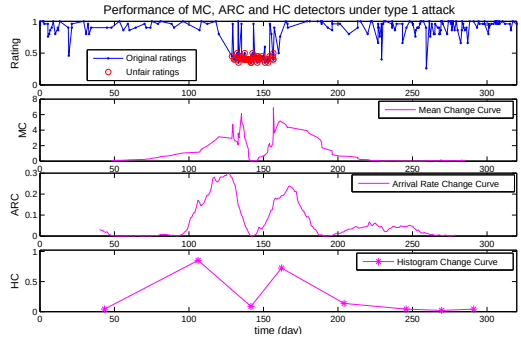

Fig. 2. Under type 1 attack (duration: 30 days, bias: 0.5, variance: 0.1 × variance of honest ratings, arrival rate: 3 × arrival rate of the honest ratings.)

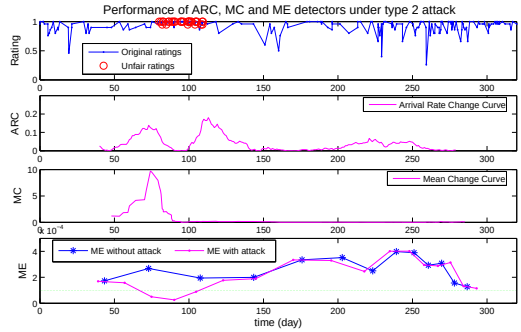

Fig. 3. Under type 2 attack (duration: 30 days, bias: 0.1, variance: 0.1 × variance of honest ratings, arrival rate: 2 × arrival rate of the honest ratings.)

be higher than that in type 1. Therefore, the *ARC detector* can detect type 2 and type 3 unfair ratings well. However, the *ARC detector cannot be the only indicator* because it has high false alarm rate. In practice, the arrival rate of fair ratings can also change rapidly. For example, after a sale event, more people come to rate a particular product.

- Through experiments, we found that the *ME detector* can detect type 2 unfair ratings, but not other types. ME detector and the ARC detector can work together to detect type 2 effectively.

In Figures 2, 3, and 4, we show the detection curve of different detectors under 3 types of unfair ratings. The curves illustrates the key points in the above discussion. These figures also serve as illustration of the detection results, as mentioned in Section IV-A to IV-D.

2) Comparison based on ROC curves: In this section, we construct the Receiver Operating Characteristics (ROC) curves for all four detectors in typical application scenarios.

After studying the rating data at Amazon.com, we find a typical rating model. That is, the arrival of the ratings with the same rating value approximately follows poisson arrival process. Let λ_i denote the arrival rate of ratings with value i. The arrival rate of all ratings can be described by $\lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5]$. Note that Amazon supports 5 rating levels. In addition, for many products, we find that $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 \approx 1 : 1 : 2 : 6 : 2$. Therefore, in the simulations, we set original $\lambda = [0.5, 0.5, 1, 3, 1]$. The simulation time is 90 days. Unfair ratings are inserted in a continuous 30-day duration. The starting

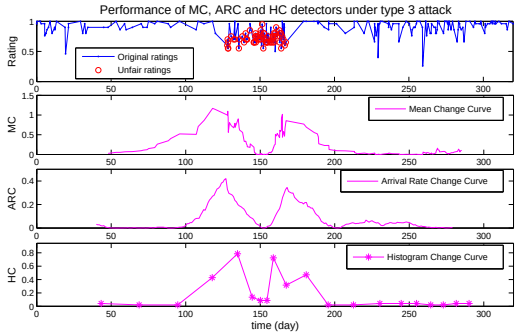

Fig. 4. Under type 3 attack (duration: 40 days, bias: 0.2, variance: 0.5 × variance of honest ratings, arrival rate: 3 × arrival rate of the honest ratings.)

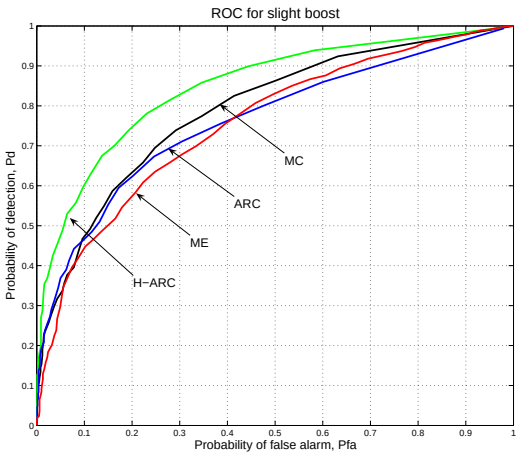

Fig. 5. Receiver operating characteristics for slight boost ($\lambda = [0.5, 0.5, 1, 3, 2]$, $mean = 3.78$)

date of 30-day duration is randomly chosen between day 31 and day 61. The unfair ratings also follow poisson arrival with arrival rate λ_a. The attackers rate 5 to boost a product, or rate 1 or 2 to downgrade a product.

In case 1, attackers insert rating value 5 with $\lambda_a = 1$. Let m denote the mean of the original ratings without the attack, and m_a denote the mean of all ratings during the attack. In this case, $m = 3.58$, $m_a = 3.78$, and the mean change is only 5.6%. Case 1 represents slightly boosting. Figure 5 shows the ROC curve (detection probability vs. false alarm probability) of ARC-H, ARC, MC and ME detectors. Considering the fact that the mean change is only 5.6%, these detectors have satisfactory performance. Among them, ARC-H is the best detector because it is designed to detect boosting only. Other detectors have similar performance.

In case 2, attackers insert rating value 5 with $\lambda_a = 2$. Here, $m_a = 3.93$ and mean change is 9.8%. Case 2 represents moderately boosting. Figure 6 shows the ROC curves. The performance of the four detectors is greatly improved. With 0.05 false alarm probability, ARC-H can detect 94% attacks. With 0.2 false alarm probability, ME (which is the worst in case 2) can detect over 90% attacks. It is noted that the HC detector does not work in the boosting scenario. Thus, its performance is not shown in Figure 5 and Figure 6.

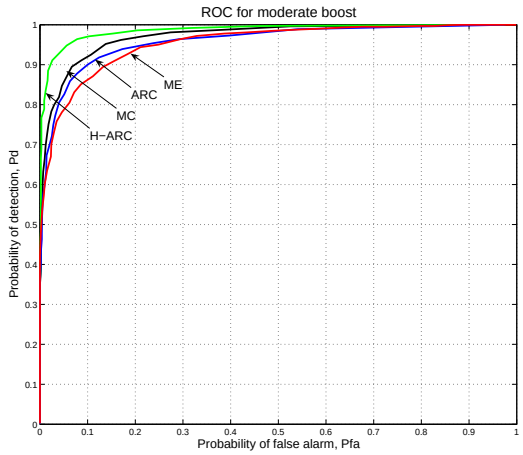

Fig. 6. Receiver operating characteristics for moderate boost ($\lambda = [0.5, 0.5, 1, 3, 3]$, $mean = 3.93$)

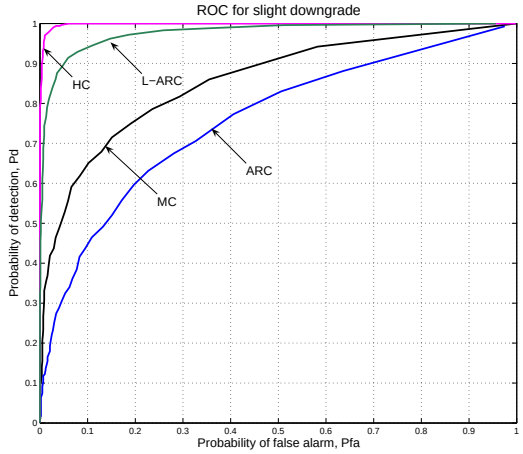

Fig. 7. Receiver operating characteristics for slight downgrade ($\lambda = [0.5, 1.5, 1, 3, 1]$, $mean = 3.36$)

In case 3, attackers insert rating value 2 with $\lambda_a = 1$ which represents slightly downgrading. Figure 7 shows the ROC curves. HC works very well. The other detectors can also detect this attack. Note that the mean change is only 6.1%. L-ARC works better than ARC because it is designed for downgrading only.

In case 4, attackers insert rating value 2 with $\lambda_a = 2$. The mean change is 10.9%. Case 4 represents moderately downgrading. Figure 8 shows the ROC curves. The performance of all detectors is greatly improved compared with that in case 3. HC and ARC-L have super performance. For MC and ARC, with 10% false alarm ratio, their detection ratio is above 92%. The ME detector does not work in downgrading scenarios and its performance is not shown in Figures 7 and Figure 8.

Through the comparisons shown in Figure 2 to Figure 8, we make the following conclusions.

- H-ARC/L-ARC is more sensitive and accurate than ARC detector. Therefore, ARC can be replaced by H-ARC/L-ARC.
- H-ARC/L-ARC is very sensitive. A small change in arrival rate will result in a detection. In the real world, the rating arrival process may not be a perfect Poisson and the arrival rate of fair ratings may have variations. In those cases, H-ARC/L-ARC will make many false alarms. Therefore, H-ARC/L-ARC should be combined with other detectors to reduce false alarm.
- Although the MC detector does not have the best performance, it works in both boosting and downgrading scenarios with stable performance.
- H-ARC/L-ARC can determine whether the attack is boosting or downgrading. The ME detector is effective to detect boosting, and the HC detector is sensitive to downgrading[1]. Therefore, H-ARC can be combined with ME, and L-ARC can be combined with HC.
- The H-ARC, L-ARC and MC detectors detect the beginning and the end of attacks. If there is an obvious 'U-shape' on the detection curve, the attack interval can easily be determined. Otherwise, they can only be used to judge the possibility of an attack but cannot decide the attack boundaries.
- The MC and HC detectors indicate the center of attack interval. They can be combined with H-ARC/L-ARC to detect the duration of the attack.

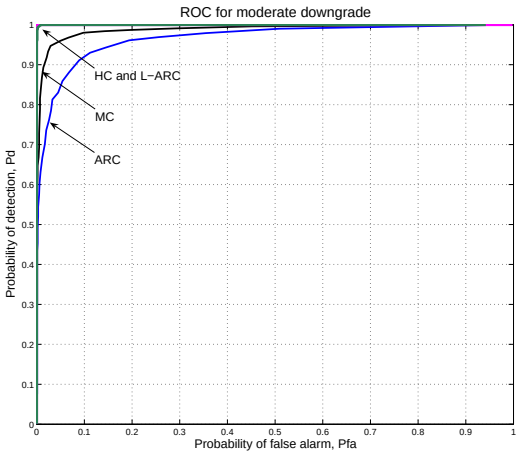

Fig. 8. Receiver operating characteristics for moderate downgrade ($\lambda = [0.5, 2.5, 1, 3, 1]$, $mean = 3.19$)

It is important to point out that the ROC curves in Figure 5 - 8 are for slight-to-moderate attacks, in which the total number of unfair ratings is not overwhelming. In other words, the arrival rate of the unfair ratings is not significant. When the arrival rate of unfair ratings is very high, the detectors will work better and the ROC curves will approach closer to the upper-left corner. The attacks become obvious, not only to the detectors, but also to the human eyes. Thus, in the simulation and experiments in

[1]This statement is true when the product quality is good. That is, there is a small room for boosting and a large room for downgrading. If the product quality is very low, we simply switch ME and HC in the above statement, in the following statements, and in Figure 9.

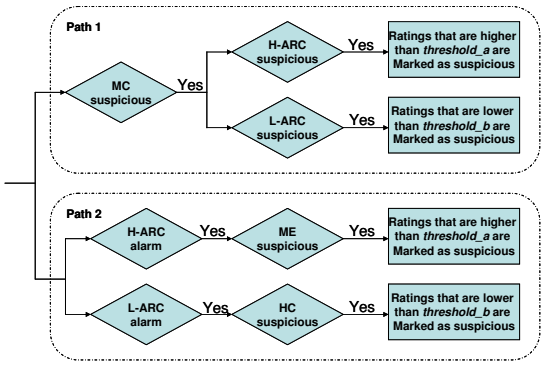

Fig. 9. Joint Detection of Suspicious Ratings

this paper, we focus on slight-to-moderate attacks by imposing a constraint on the total number of malicious user IDs.

F. Integrated Detection

We have developed detectors for mean change, arrival rate change, histogram change, and model error change. The problem formulation for individual detectors are different. This is because that attack behaviors are very diverse and cannot be described by a single model. Different attacks have different features. For example, one attack may trigger MC and H-ARC detectors, another attack may trigger only L-ARC and HC detectors.

In addition, the normal behaviors, i.e. fair ratings, are not stationary. Even without unfair ratings, the statistics of the fair ratings can have variation. In smart attacks, the changes caused by unfair ratings and the nature changes in fair ratings are sometimes difficult to differentiate. Thus, using a single detector will cause a high false alarm rate. Based on the above discussion, we develop an empirical method to combine the proposed detectors, as illustrated in Figure 9.

There are two detection paths. Path 1 is used to detect strong attacks. If the MC indicator curve has a U-shape, and H-ARC or L-ARC indicator curve also has a U-shape, the corresponding high or low ratings inside the U-shape will be marked as suspicious. If for some reason, H-ARC (or L-ARC) indicator curve does not have such a U-shape, H-ARC (or L-ARC) alarm is issued. The alarm will be followed by the ME or HC detector. This is path 2. Path 2 detects suspicious intervals. Since there may be multiple attacks against one product, the ratings must go through both paths. Path 1 and Path 2 are in parallel.

G. Trust in Raters

It is noted that we cannot perfectly differentiate unfair ratings and fair ratings in the suspicious intervals. Therefore, some fair ratings will be marked as suspicious. As a consequence, one cannot simply filter out all suspicious ratings. In our work, this suspicious rating information is used to calculate trust in raters, based on the beta-function trust model [19]. The calculation is described in Procedure 1.

Procedure 1 Computing Trust in Raters

For each rater i, initialize $S_i = 0$, and $F_i = 0$
for $k = 1 : K$ **do**
  % Let $\hat{t}(k)$ denote the time when we calculate trust in raters.
  % k is the index.
  for each rater i **do**
    Set $n_i = f_i = 0$,
    Considering all products being rated during time $\hat{t}(k-1)$ and $\hat{t}(k)$, determine:
      n_i: the number of ratings that is provided by rater i
      f_i: the number of ratings from rater i and being marked as suspicious
    calculate $F_i = F_i + f_i$ and $S_i = S_i + n_i - f_i$.
    calculate trust in rater i at time $\hat{t}(k)$ as: $(S_i+1)/(S_i+F_i+2)$.
  end for
end for

H. Rating Aggregation

Several trust models have been compared for rating aggregation in [24]. In the comparison, the modified weighted average trust model outperforms other methods. The intuitive explanation is as follows. Many trust models tend to lower the trust value when trust propagates through the third party. For instance, let T_{ab} denote how much A trusts B and T_{bc} denote how much B trusts C. Many trust models will calculate the trust between A and C, denoted by T_{ac}, such that $T_{ac} < T_{ab}$ and $T_{ac} < T_{bc}$. This is reasonable in many applications, such as in P2P networks and public key certification, in which trust value represents whether you can depend on the other party to perform an action. However, in rating aggregation, the goal is to accurately estimate quality of products. It is found in [24] that many trust models make the aggregated rating value lower than the product quality, whereas the weighted average algorithm does not have this problem. Thus, with a slight modification that removes the ratings from users with low trust, the modified weighted average trust model is adopted.

Let R denote the set of raters whose ratings are the inputs to the aggregation module. If rater $i \in R$, let r_i denote the rating from rater i and T_i denote the current trust value of rater i. In addition, each rater provides only one rating for one object and R_{ag} denotes the aggregated rating. Then,

$$R_{ag} = \frac{1}{\sum_{i:i\in R} max(T_i - 0.5, 0)} \sum_{i:i\in R} r_i \cdot max(T_i - 0.5, 0) \ . \tag{7}$$

It is noted that 0.5 is used as the threshold value in (7). This is because the initial trust value is 0.5 when there is no observations (i.e. $S_i = 0$ and $F_i = 0$ in Procedure 1). In simulations, we observe that the overall performance of the proposed system is not sensitive to the choice of this threshold.

Finally, from the description of the algorithms, it is easy to see that the computation and storage complexity of the proposed scheme increases linearly with the number of products or the number of users.

V. Performance Evaluation

A. Rating Challenge and Experiment Description

For any attack-resistant on-line rating analysis and aggregation system, it is very difficult to evaluate their performance in practical settings. This is due to the lack of realistic unfair rating data. Even if one can obtain data with unfair ratings from e-commerce companies, there is no ground truth about which ratings are dishonest.

To understand human users' attacking behavior and evaluate the proposed scheme against non-simulated attacks, we designed and launched a *Rating Challenge* [17]. In this challenge,

- We collected real online rating data for 9 flat panel TVs with similar features. The data are from a well-known online-shopping website.
- The participants to the Rating Challenge download the rating dataset and control 50 biased raters to insert unfair ratings. In particular, the participants decide when the 50 raters rate, which products they rate for, and the rating values.
- The participants' goal is to boost the ratings of two products and reduce the ratings of another two products.
- The successfulness of the participants' attack is determined by the **overall manipulation power**, called MP value. For each product, we calculate $\Delta_i = |R_{ag}(t_i) - R^o_{ag}(t_i)|$ during every 30 day period, where $R_{ag}(t_i)$ is the aggregated rating value with unfair ratings, and $R^o_{ag}(t_i)$ is the aggregated rating value without unfair ratings. The overall MP value is calculated as $\sum_k (\Delta^k_{max_1} + \Delta^k_{max_2})$, where $\Delta^k_{max_1}$ and $\Delta^k_{max_2}$ are the largest and 2nd largest among $\{\Delta_i\}'s$ for product k.
- The participants that can generate the largest MP value win the competition.

As we discussed previously, in the real world, attackers may boost their own products and downgrade the rivals' products at the same time. Thus, we use MP instead of the aggregated rating score of a single product to rank the submitted attacks.

In addition, the real online rating data may contain unfair ratings. After examining the rating data manually, such as checking whether there is large variation in the aggregated rating scores in a very short time and whether the rating scores for the same product at different websites are consistent, we assume that the real online rating data do not contain collaborative unfair ratings. Recall that the goal of the collaborative unfair ratings is to introduce bias in aggregated rating scores.

In the Rating Challenge, the participants know the original online rating dataset, the rule of the competition including detailed evaluation criteria, the MP score of each of their submissions, and their rank. They do not know the underlying detection algorithms. The numbers of honest ratings for four products are 177, 102, 238, and 201, respectively. Since the number of unfair ratings for one product cannot be more than 50, the ratio between the total number of dishonest ratings and that of honest ratings is less than 0.28.

The Rating Challenge has been successful. We have collected 252 eligible submissions of unfair ratings. We evaluate the performance of the proposed system using the unfair rating data. In the experiment, all raters' trust values are initially assigned as 0.5. The window size of the MC detector, H-ARC/L-ARC detectors, HC detector, and ME detector are 30 (days), 30 (days), 40 (ratings), and 40 (ratings), respectively. In the H-ARC and L-ARC, $threshold_a = 0.5m$ and $threshold_b = 0.5m + 0.5$, where m is the mean of the ratings in the time window.

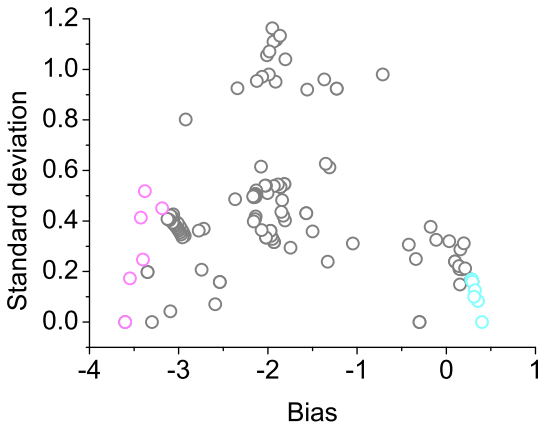

Fig. 10. Features of real users' attack behavior. (Red color indicates strong downgrading attacks for SA scheme; blue color indicates strong boosting attacks for SA scheme. About 252 attacks against product 1.)

For the purposed of comparison, we also evaluate the performance of three other schemes.

1) SA scheme - No attack detection, using simple averaging for rating aggregation.
2) BF scheme - Using the beta-function based filtering technique proposed in [5] to remove unfair ratings. Then, the trust value of rater i is calculated as $(S_i+1)/(S_i+F_i+2)$, where F_i is the number of ratings (from rater i) that have been removed, and $F_i + S_i$ is the total number of ratings provided by rater i.
3) DC scheme - Using the proposed detectors without trust establishment.

B. Results

We have collected 252 valid submissions. Each submission contains unfair ratings for 4 products. Since each participant controls 50 user IDs, he/she can insert up to 50 unfair ratings for each product. In total, we have 1008 sets of unfair ratings and each set contains no more than 50 unfair ratings for one product. The features of the attack data are shown in Figure 10, in which each circle represents one attack against product 1. The x-axis is bias, i.e. difference between the mean of dishonest ratings and the mean of honest ratings. The rating values are between 0 and 5, and the mean of honest ratings for product 1 is around 4. Thus, the bias is between -4 and 1. Positive bias means promotion and negative bias means downgrading. The y-axis is the standard derivation of the unfair rating values. From Figure 10, we can see that the collected attack dataset

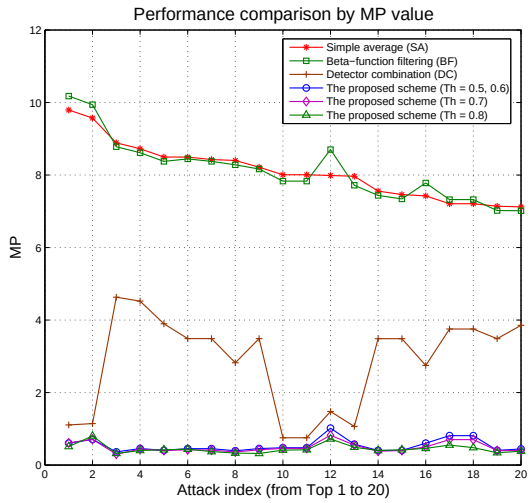

Fig. 11. Performance comparison in terms of the MP resulting from Top 20 attacks against SA

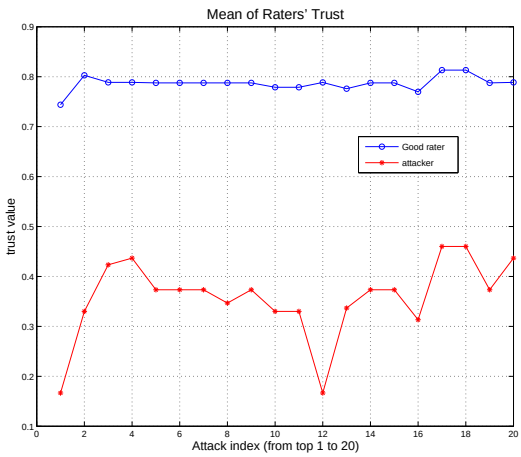

Fig. 12. Trust values of the raters in the proposed scheme

contains many type 1, 2, and 3 attacks. The bias and variance vary in a large range. The attack data against other products have similar features. The *attack duration*, which is not shown in Figure 10, also has a large variation, ranging from a few days to the entire lifetime of the products. It is important to point out that duration, bias, and variance are not the only features of an attack. Some participants inserted unfair ratings manually. With the same statistical properties, two attacks can have different strength.

There are many attack data available. We cannot show all of them due to space limitation. Instead, we compare the performance of different schemes under the strongest attacks.

In Experiment 1, we pick top 20 attacks against the SA scheme. That is, these 20 attacks generate the highest MP values when the SA scheme is used. To generate strong attacks against the SA scheme, attackers insert unfair ratings with large bias and the attack duration is not long. In Figure 11, the four schemes are compared under these 20 attacks. The horizontal axis is the index of the attack data set from top 1 to top 20. The vertical axis is the overall MP value, which is the summation of individual products' MP values in each submission. For the proposed scheme, we plot its performance when choosing different threshold values in the rating aggregation equation (7). It is seen that the performance of the proposed scheme is not sensitive to the threshold selection. From this figure, four observations are made.

First, it is clear that the proposed scheme has the best performance. It can significantly reduce the MP values resulting from real user attacks.

Second, the performance of the SA scheme is similar to that of the BF scheme. The BF method is even slightly worse than the SA in some situations. There are two reasons. When the unfair ratings are concentrated in a short time interval, the majority ratings in this time interval can be unfair ratings. Using the majority rule, the beta filter will in fact remove good ratings. This is why the beta filter performs worse. In addition, when the attacks do not have a large bias, the beta filter cannot detect unfair ratings. This is why the beta filter has almost the same performance as the simple averaging under some attacks. Therefore, the beta filter scheme, as well as other majority-rule based schemes, is not affective in detecting smart attacks from real human users.

Third, trust establishment plays an important role in the proposed scheme. Without trust model, although the proposed detectors can detect the attack and reduce MP value greatly, the performance is still worse than that of our proposed scheme with trust. No matter how good the detectors are, there is a small amount of false alarm. With trust establishment, good users and bad users can be distinguished in several rounds rather than in one shot. In Figure 12, we show the trust values of the honest raters and the trust values of the raters inserted by the attackers. Our trust model works very well in distinguishing honest raters from attackers.

Fourth, the top 1 attack has the greatest impact upon the aggregated rating when SA and BF schemes are used. However, when the proposed scheme is used, this top 1 attack cannot do much to the aggregated rating value. We studied this attack and found that the attacker concentrates all unfair ratings in short time periods regardless boosting or downgrading. This aggressive attack is caught by the proposed scheme accurately but cheats the SA and BF schemes very well.

In Experiment 2, we select the top 20 attacks that are strongest against the BF scheme. Figure 13 shows the performances of the four schemes. Again, the proposed scheme has the best performance. SA and BF have the similar performance. DC can catch most of the attacks but its performance is worse than that of the proposed scheme with trust establishment.

In this experiment, we also calculate the detection rate and false alarm rate for individual detectors and for the DC scheme. The minimum, maximum, and average values are shown in Figure 14. The detection rate and false alarm rate of a detector only count the attacks that trigger this detector. For example,

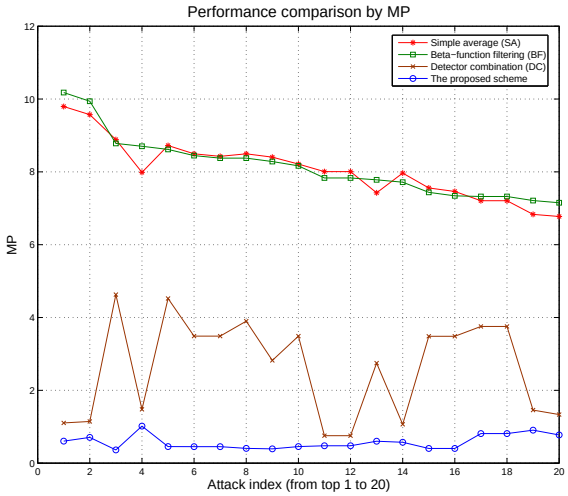

Fig. 13. Performance comparison in terms of the MP resulting from Top 20 attacks against BF

	Detection Rate			False Alarm Rate		
	Min	Max	Average	Min	Max	Average
L-ARC	80%	94%	**86%**	5%	19%	**14%**
H-ARC	72%	88%	**82%**	8%	23%	**17%**
MC	70%	86%	**78%**	12%	32%	**25%**
HC	76%	94%	**84%**	10%	26%	**20%**
ME	42%	82%	**66%**	15%	37%	**28%**
Combination	42%	96%	**90%**	5%	16%	**12%**

Fig. 14. Performance of individual detectors and their combination

a downgrade attack will not trigger H-ARC, and this attack is not counted in the calculation of the false alarm/detection rate of H-ARC. It is seen that combining the detectors can improve the detection rate and reduce the false alarm rate. The L-ARC detector has relatively good performance because the strong attacks against the BF scheme often introduce a large change in arrival rate such that the dishonest ratings become majority in the decision window. Another detector may have relative good performance when the test is for another type of attacks. Combining the detectors has two advantages: (1) improving detection rate and reducing false alarm rate; and (2) handling a variety of attacks.

In Experiment 3, we select the top 20 attacks that are strongest against the DC scheme. These attacks often have moderate bias and moderate variance. Figure 15 shows the MP values. Even in the tough cases, DC's performance is still much better than SA and BF schemes as shown in this figure.

Finally, in Experiment 4, we pick the top 20 attacks against the proposed scheme. The performance comparison is shown in Figure 16. The proposed scheme is still better than SA and BF except for two cases (attack 3 and 6), which are the only two cases where SA and BF perform better than or close to the proposed scheme. Another important observation is that trust evaluation does not necessarily improve the performance in Figure 16.

Each malicious user provides up to 4 ratings. If all these ratings are unfair ratings, the malicious users can cause bigger

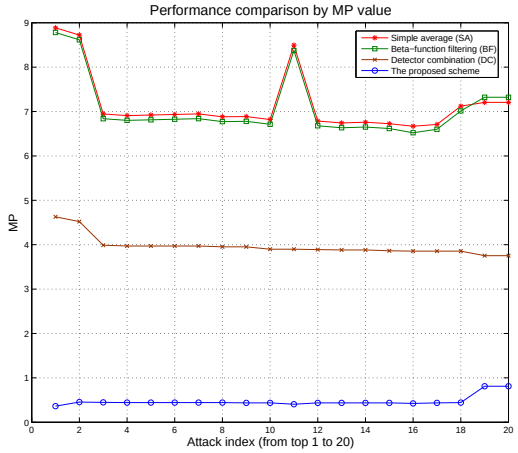

Fig. 15. Performance comparison in terms of the MP resulting from Top 20 attacks against DC

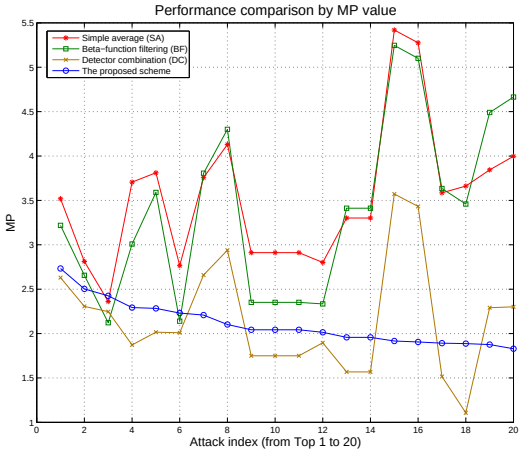

Fig. 16. Performance comparison in terms of the MP resulting from Top 20 attacks against the proposed scheme

change in the average rating scores. In previous experiments, we pick 20 attacks that are strongest against SA, BF, and DC schemes, respectively. The malicious users in these attacks often provide 4 unfair ratings. When the malicious users each provide 4 unfair ratings and honest users each provide a few honest ratings, the trust evaluation mechanism will lower the trust scores of malicious users, which makes trust evaluation very effective. This is why the proposed scheme has large performance advantage over the DC scheme (detector only) in Figure 11, 13, and 15.

In Figure 16, we pick strongest attacks against the proposed scheme. After carefully examining the attack data, we find that the malicious users in these attacks are smart enough to boost their trust values. Specifically, they give honest ratings to some products to boost their trust value, and then give 1$\sim$2 unfair ratings to attack the target products. Furthermore, some participants even find a way to attack the unfair rating detection algorithms. They insert high ratings when there are high ratings from honest raters, and/or insert low ratings when there are

Fig. 17. Trust values of the raters in the proposed scheme

	Against top 20 strongest attacks			Against all attacks
	Min	Max	Average	Average
Simple Average	7.12	9.79	8.10	4.8
Beta-function filtering	7.15	10.18	8.14	4.3
Detector Combination	3.75	4.63	3.96	2.1
The proposed scheme	1.83	2.73	2.11	0.9

Fig. 18. MP values resulting from top 20 strongest attacks and all attacks

low ratings from honest raters. By doing so, they make the honest ratings look suspicious to the ME detector. This causes false positive and reduces the trust value of honest users. Both strategies hurt the effectiveness of trust evaluation.

Figure 17 shows the average trust values of honest and dishonest users in this experiment. Clearly, the average trust value of malicious users is slightly higher than that of honest raters in attack 3 and attack 6. As a consequence, the performance of the proposed scheme is sometimes better and sometimes worse than the DC scheme. This represents the case in which trust evaluation is not effective.

However, this is not a big problem because the attack power has been diluted and MP under this type of attack is not high. In Figure 16, the MP value of attack 3 and attack 6 is around 2, which is much lower compared with the MP values without the proposed scheme in previous experiments. Even if the proposed scheme cannot catch those unfair ratings accurately, the rating system is reliable with stable performance.

In Figure 11, 13, 15 and 16, we have shown show the MP values of the four aggregation algorithms under their respective worst case scenarios. That is, we have picked the top 20 strongest attacks against each of the four algorithms. In the first four columns in the table in Figure 18, we show the minimum, maximum and average MP values of each individual method when they are facing these strongest attacks against them. In the last column, we show the average MP value when the defense methods are tested against *all attacks* in the attack dataset. The advantage of the proposed scheme is clearly shown. Compared with the majority-rule based methods and simple averaging, the proposed scheme reduces the MP value by a factor of 3 or more.

From Experiments 1 through 4 and feedbacks from the Rating Challenge winners, we realize that game theory might be used to describe the interaction between attackers and the defense mechanisms. If the attacker does not know that there is a detector, he/she would be aggressive and pursue the greatest MP by concentrating unfair ratings with a high bias, which can be detected accurately by the proposed scheme. If the attacker is afraid of being detected, he/she would be cautious and insert the unfair ratings with a small bias. He/she may even attack the detection algorithms directly. These issues are worth further investigation. Although our detector cannot detect some smart attacks, the MP values of these attacks are also low. The proposed scheme can greatly reduce the influence of collaborative unfair ratings, and therefore make the rating system reliable and robust.

VI. Conclusion

In this paper, we address the problem of detecting and handling unfair ratings in on-line rating systems. In particular, we design a comprehensive system for integrating trust into rating aggregation process. For detecting unfair ratings, we develop a model error based detector, two arrival rate change detectors, a histogram change detector, and adopt a mean change detector. These detectors cover different types of attacks. A method for jointly utilizing these detectors is developed, based on the classification of human users' attacking behavior and ROC analysis. The proposed solution can detect dishonest raters who collaboratively manipulate rating systems. This type of unfair raters is difficult to be caught by the existing approaches. The proposed solution can also handle a variety of attacks. For performance evaluation, we design and launch a Rating Challenge to collect real users' attacking data. The proposed system is evaluated against attacks created by real human users and compared with majority-rule based approaches. Significant performance advantage is observed.

Acknowledgment

The authors would like to sincerely thank Jin Ren and Nitish Reddy Busannagari for administrating Rating Challenge website, and all participants to the Rating Challenge.

References

[1] C. Dellarocas, "The digitization of word-of-mouth: Promise and challenges of online reputation systems," *Management Science*, vol. 49, no. 10, pp. 1407–1424, October 2003.

[2] Epinion, "http://www.epinions.com/," Shopping.com, Inc.

[3] M. Chen and J.P. Singh, "Computing and using reputations for internet ratings," in *Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001.

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

[5] A. Whitby, A. Jøsang, and J. Indulska, "Filtering out unfair ratings in Bayesian reputation systems," in *Proc. 7th Int. Workshop on Trust in Agent Societies*, 2004.

[6] J. Zhang and R. Cohen, "Trusting advice from other buyers in e-marketplaces: the problem of unfair ratings," in *Proceedings of the 8th international conference on Electronic commerce*, 2006.

[7] J. Weng, C. Miao, and A. Goh, "An entropy-based approach to protecting rating systems from unfair testimonies," *IEICE TRANSACTIONS on Information and Systems*, vol. E89-D, no. 9, pp. 2502–2511, September 2006.
[8] Robin S. Poston, "Using and fixing biased rating schemes," *Communications of the ACM*, vol. 51, no. 9, pp. 105–109, 2008.
[9] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.
[10] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, July 2004.
[11] K. Fujimura and T. Nishihara, "Reputation rating system based on past behavior of evaluators," in *Proceedings of the 4th ACM conference on Electronic commerce*, 2003.
[12] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *In Proceedings of IEEE Symposium on Security and Privacy*, May 2008.
[13] N. Tran, B. Min, J. Li, and L. Submaranian, "Sybil-resilient online content voting," in *In Proceedings of the 6th Symposium on Networked System Design and Implementation (NSDI09)*, Apr. 2009.
[14] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys (to appear)*, vol. 14, no. 4, Dec. 2009.
[15] D. H. McKnight and N. L. Chervany, "The meanings of trust," MISRC Working Paper Series, Technical Report 94-04, arlson School of Management, University of Minnesota, 1996.
[16] C. Dellarocas, "Strategic manipulation of internet opinion forums: Implications for consumers and firms," *Management Science*, October 2006.
[17] University of Rhode Island, "Etan rating challenge," www.etanlab.com/rating.
[18] Y. Sun and Y. Yang, "Trust establishment in distributed networks: Analysis and modeling," in *Proceedings of IEEE ICC'07*, 2007.
[19] A. Jøsang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th Bled Electronic Commerce Conference*, June 2002.
[20] Y. Sun, Z. Han, W. Yu, and K. J. Ray Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *Proc. IEEE INFOCOM'06*, April 2006.
[21] S. Buchegger and J-Y Le Boudec, "The effect of rumor spreading in reputation systems in mobile ad-hoc networks," in *Proceedings of Wiopt'03*, 2003.
[22] Steven M. Kay, *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*, Prentice Hall, 1998.
[23] M.H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley and Sons, 1996.
[24] Y. Yang, Y. Sun, J. Ren, and Q. Yang, "Building trust in online rating systems through signal modeling," in *Proceedings of IEEE ICDCS Workshop on Trust and Reputation Management*, 2007.

PLACE PHOTO HERE

Yafei Yang Biography text here.

Yan Sun Biography text here.

Steven Kay Biography text here.

Qing Yang Biography text here.