TULUNGAN: A Consensus-Independent Reputation System for Collaborative Web Filtering Systems

Alexis V. Pantola[*1], Susan Pancho-Festin[2], Florante Salvador[1]
[1]De La Salle University, 2401 Taft Avenue, Manila, Philippines
Telephone: +632-5264247
[2]University of the Philippines, Diliman, Quezon City, Metro Manila, Philippines
*Corresponding author: pong.pantola@delasalle.ph

ABSTRACT

Web filtering systems allow or prohibit access to websites based on categories (e.g., pornography, violence, sports, etc.). Categorization of websites can be done automatically or manually. Automatic categorization is prone to under- and over-blocking. On the other hand, manual approach is typically performed by a limited number of people making it not scalable.

Collaborative web filtering systems, a variation of manual categorization, allow anyone to categorize websites in order to determine which domain these sites belong (e.g., pornography, violence, sports, etc.). This attempts to solve the scalability issue of the typical manual method.

The approach offered by collaborative web filtering relies heavily on the contribution of users in order to make the system scalable and less prone to errors. However, its success is greatly dependent on user cooperation. To promote cooperation, reputation system can be used in web filtering.

A previous study called Rater-Rating promotes cooperation and explores the use of a user-driven reputation system that measures both the contributor and rater reputation of users of a collaborative web system. However, Rater-Rating is consensus dependent. If the number of malicious users are more than their good counterparts, the reputation system can be defeated. In other words, the system can mistakenly give malicious users a high reputation value.

This study discusses a reputation system called Tulungan that is consensus-independent. It can detect the presence of malicious users even if the number of their good counterparts are fewer. A simulation result that compares the effectiveness of Tulungan relative to Rater-Rating is presented in this paper. The simulation shows that Tulungan is still effective even with 25% good users while Rater-Rating requires at least 50% good users to be effective.

Keywords: Reputation System Web Filtering

INTRODUCTION

Collaborative web systems such as Wikipedia *(Wikipedia, the free Encyclopedia [Online] , 2001)* and Untangle *(Untangle - Multi-functional firewall Soft- ware - Open Source Content Filter and Spam Filter [Online]* , n.d.) allow anyone to contribute information. Such set-up allows these systems to grow since contribution can come from different parts of the world (Ebner & Zechner, 2006). However, open collaboration also is prone to issues such as low credibility and inaccuracy of contents. These can be caused by malicious contributors who vandalize contents in such systems (Chesney, 2006). In some cases, "lazy" contributors, as opposed to malicious ones, may haphazardly post entries without verifying their correctness. There are also deviant contributors, whose intentions are good, however, their view deviates from other users such that their contributions are generally considered as incorrect.

Because of such problems, reputation systems are developed to measure the credibility of contents and/ or contributors. A reputation system is composed of several interacting modules that address problems such as existence of bad contributors and contributions. To solve such problems, algorithms on distinguishing good and bad contributors and contributions are developed and implemented in these modules. These algorithms utilize information gathered from transactions in a web system to assess contributors and contributions (Kennes & Schiff, 2003). Transactions can be in the form of purchase or sale in e-commerce sites or contribution in collaborative systems.

This paper presents a reputation system called Tulungan that can detect the presence of malicious users even if the number of their good counterpart are fewer. A simulation result that compares the effectiveness of Tulungan relative to Rater-Rating is presented in this paper. Aside from this, the simulation also determines the minimum percentage of good users needed in order to ensure that the system can differentiate good from malicious users.

REVIEW OF RELATED WORK

Reputation systems can be classified into two: (a) content-driven and (b) user-driven.

2.1 Content-Driven Reputation System

Content-driven reputation system relies on the content to determine the accuracy of information. As an example, WikiTrust (Adler & Alfaro, 2007; Adler, Chatterjee, et al., 2008; Adler, Alfaro, Pye, & Raman, 2008) relies on the contents of Wikipedia pages to determine their accuracy. The stability of a content is equated to its credibility. It assumes that the less frequent an entry of a Wikipedia page is changed, the more credible it is since reviewers find it as already accurate and does not require any correction. However, such an assumption may lead to a wrong conclusion since a "non-edit" to an entry does not necessarily imply that the said entry is correct. There are cases when authors are "lazy" in reviewing and editing an entire article and focus only in entries that interest them thereby leaving other entries unedited. For example, a biography entry in Wikipedia is proven inaccurate even if it is not edited for 132 days (Cross, 2006). If the credibility of this entry is measured using WikiTrust, it is possible that it will be incorrectly labelled as accurate.

2.2 User-Driven Reputation System

User-driven reputation system relies on user rating to measure the accuracy of contents. For instance, users of Reddit can provide contents and these are assessed by other users by rating them as good or junk *(Reddit.com: Help [Online]* , n.d.). It solves the issue of its content-driven counterpart by not relying on the frequency of edits to determine if submitted contents are accurate. Its accuracy is measured based on the rating of users.

In order to make user-driven reputation system successful, cooperation among raters is needed. Without cooperation, raters may not provide any feedback (Steiner, 2003) or if ever they provide one,

it is biased or negatively influenced by external factors. In other cases, the users may even manipulate the system to improve their own reputation or degrade the reputation of others.

Encouraging raters to provide honest rating can be a challenge as seen in websites that utilize such system. In eBay, some raters refuse to give negative ratings for fear of retaliation. Such incorrect rating has negative effects to users of collaborative sites (Yao, Fang, Dineen, & Yao, 2009). There are even cases when users resort to self-rpomotion by resorting to Sybil attack (Douceur, 2002). Sellers in eBay can create alter egos or phantom user accounts and use these to give a positive rating to their original account (Zittrain, 2008; *Some eBay Users Abuse Auction Site's Feedback System, Professor Finds [Online]*, 2007). Aside from self-promotion, users may resort to other form of attacks such as whitewashing, slandering, orchestrated, and denial of service (Hoffman, Zage, & Nita-Rotaru, 2009).

There are reputation systems that encourage cooperation[1]. However, they are designed to work in mobile ad-hoc networks (MANETs) *(IETF MANET Working Group. Mobile Ad Hoc Networks (MANET). WorkingGroup charter [Online]*, n.d.) and not in collaborative web systems.

A MANET is a collection of mobile devices or nodes that communicate with each other through wireless technology. Since wireless communication has a finite and limited range, nodes in a MANET need to rely on each other to forward and route messages. Several routing algorithms such as Dynamic Source Routing (Johnson, Maltz, & Broch, 2001) and Ad-hoc On Demand Distance Vector Routing (Perkins & Royer, 1999) allow mobile nodes to determine the path a message should pass through in order for it to be routed from one node to another. These routing algorithms assume that nodes are cooperative and are expected to forward messages in behalf of other nodes (Trivedi et al., 2009). However, due to resource constraints such as battery life, nodes tend to be selfish and refuse to forward messages in order to limit battery consumption. There are also cases where nodes are programmed to be malicious and cause problems in a MANET.

Due to selfish and malicious nodes, several reputation systems were introduced to mitigate the effects of such nodes (Marti, Giuli, Lai, & Baker, 2000; Buchegger & Le Boudec, 2002a, 2002b; Michiardi & Molva, 2002; Bansal & Baker, 2003; Hu & Burmester, 2006). In a system proposed by (Marti et al., 2000), misbehaving nodes are mitigated through the use of two components, a watchdog and a pathrater. The watchdog is used to identify misbehaving nodes. This is accomplished by checking which nodes do not forward packets. Once misbehaving nodes are determined, the pathrater uses this information to know which nodes are more reliable in forwarding packets. The pathrater prevents messages from not being forwarded by avoiding a path that includes misbehaving nodes. However, such system may ironically encourage misbehavior of nodes since a misbehaving node will be off-loaded from forwarding messages. As a result, its battery life can be conserved. A modification of this system, called CONFIDANT, was presented in (Buchegger & Le Boudec, 2002a) and its result shown in (Buchegger & Le Boudec, 2002b).

CONFIDANT implements a concept called neighbor watch. In this scheme, neighboring nodes monitor and detect malicious activities. This increases the chance of all legitimate nodes detecting bad behavior since neighbor watch allows multiple nodes to report malicious activities. Aside from this, it punishes bad behavior by not providing services to misbehaving nodes. This solves the issue presented by (Marti et al., 2000).

Two systems CORE (Michiardi & Molva, 2002) and OCEAN (Bansal & Baker, 2003) provide a similar scheme presented by (Marti et al., 2000; Buchegger & Le Boudec, 2002a). In addition, they addressed problems concerning reporting false misbehavior which can result to denial of service. CORE accomplishes this by ignoring negative reports and focusing only on positive ones. On the other hand,

[1]In reputation systems, encouraging cooperation can be accomplished by making misbehavior unattractive or by providing incentive for good behavior (Buchegger & Le Boudec, 2002b).

second-hand information regardless if the report is negative or positive.

Another reputation system called Lars (Hu & Burmester, 2006) addresses the same concern solved by CORE and OCEAN. However, Lars does not ignore negative reports. Reports on misbehavior are still processed in this scheme as long as they are reported by a significant number of nodes.This reduces the chance that a well-behaved node will be denied service.

Several concepts applied in MANET reputation systems can be applied in collaborative web systems (Pantola, Pancho-Festin, & Salvador, 2010). Users of collaborative web systems can be given an additional task of reviewing and rating the work of contributors in order to improve the accuracy of the contributions. However, such extra work may not be attractive to "lazy" users who tend not to review the contribution of others. Systems applied to mobile nodes to encourage message passing can also be applied to users to encourage them to review and rate.

Collaborative web systems may require users to review the work of others. However, there is no assurance that they will do it properly. The concept of watchdog in the system of (Marti et al., 2000) can be applied in collaborative systems in order to detect raters who "misbehave" or those who do not review and rate properly.

Once misbehaving users are detected, there should be a corresponding penalty similar to CONFIDANT. One way to implement this is to compute not only the reputation of users based on the quality of their contribution but also consider their reputation when rating the contribution of others. Bad raters can be penalized by not requiring other users to review their work. An unreviewed work should be given a low rating and thus the reputation of the author of a sub-standard work is also negatively affected. However, caution must be exercised when implementing such approach since other users of such collaborative web systems may be ultimately affected if many contributions remain unreviewed.

The implementation of the reputation system must also consider denial of service. The rating made by a good rater may be manipulated by misbehaving ones in order to bring down the reputation of the former. CORE, OCEAN, and Lars can be adopted to address such concern.

As mentioned above, concepts behind these reputation systems for MANETs can be adopted by collaborative web systems such as community-based web filtering.

Web filtering is the evaluation of web resources in relation to a set of parameters. Such evaluation can be used to regulate access to web contents (Bertino, Ferrari, & Perego, 2010) such as pornography, violence, and racism. (Bertino, Ferrari, Perego, & Zarri, 2005) presented an integrated approach to rating and filtering web content. It combines existing approaches (i.e. list-based and metadata-based) with content labelling. This mitigates problems of previous systems that enforce a restrictive and often ineffective filtering. Such ineffectiveness results to under- and/or over-blocking of web access (e.g., pornographic and gynecology sites are considered as having similar category). However, these list-based and metadata-based filtering approaches have deficiencies and are inadequate (Noll & Meinel, 2006). These methods do not scale with the rapid growth of the Internet. Manually updating a list or metadata cannot catch up with the increasing number of new websites. In 2005, there were approximately 63 million active websites, with an average increase of 1.2 million sites per month (Noll & Meinel, 2005). A survey conducted by Netcraft *(Netcraft - Internet Research, Anti-Phishing and PCI Security Services [Online]*, n.d.) in November, 2010 saw an increase on the number from 63 to 100 million active websites.

There are algorithms available that can potentially automate the process of list-based and metadata-based approaches. However, such algorithms face particular challenges. Aside from difficulty in extracting information from web documents containing different types of data (e.g., images, videos, Java applets, or Flash animation), these algorithms depend heavily on the quantity and quality of training input

(Noll & Meinel, 2006). With the current number of active websites, the training phase would either be inadequate or take too long.

Due to these challenges, social or collaborative web filtering becomes a practical alternative. This approach empowers the end users (i.e., actual recipients of web content) to categorize websites (Noll & Meinel, 2006). This is a form of crowdsourcing that utilizes the Internet *(Following the crowd [Online],* 2008). Crowdsourcing (Howe, 2006), is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call. This large group can be the community of Internet users that can collaborate in performing web filtering.

TaggyBear (Noll & Meinel, 2008) is an example of a scalable collaborative web filter. It allows end users to collaborately categorize web pages and mark them as objectionable (e.g. pornographic content, violence, racism) or dangerous (e.g. phishing, malware). It uses several types of information like user and community[2]. The user information provides the category of a particular website based on the perception of a user. An aggregate of the categories made by several users is known as "community information". As an example, a website with a community information of porn:13:68 means that 13 out of 68 users categorized the website as a pornographic site.

Other established websites such as *OpenDNS (OpenDNS - Solutions - Business/Enterprise - Web Content Filtering [Online],* 2008) and Untangle *(Untangle - Multi-functional firewall Software - Open Source Content Filter and Spam Filter [Online],* n.d.) apply a similar concept of categorizing other websites through the help of end users.

[2]The term used in TaggyBear is actually types of rating and not types of information. However, in order not to confuse with the rating that this study is focusing on, the word information is used instead.

OpenDNS started as a DNS service in 2006 and eventually widened its function by including collaborative web filtering. Registered users can categorize websites (e.g., gambling, video sharing, social networking). Other users can vote on the accuracy of the category made (Broersma, 2008). Un- like Taggbear, the web filtering service offered by OpenDNS is cloud-based and is meant for enterprise deployment *(A Radically Simpler Approach to Web Content Filtering and Security [White paper],* n.d.).

Untangle also offers web filtering service. Using its web filter submission tool, registered users can contribute by categorizing different web URLs *(Untangle - WebFilter Technical Specifications [Online],* n.d.). However, unlike Taggybear and OpenDNS, contributions made by the users are verified and approved by the technical support of Untangle *(Untangle - Web Filter submission tool [Online],* n.d.) that act as community or content managers. These managers are responsible for sustaining and nurturing the contributions (Gray, 2010). This makes its web filtering less scalable since the verification process is done by a relatively fewer number of people.

The three collaborative web filtering systems enable registered users to categorize websites. However, TaggyBear relies on the consensus of the majority to determine the accuracy of a category. On the other hand, OpenDNS depends on a voting mechanism to measure the accuracy of categories. This presents a new problem since the accuracy of the voting mechanism itself is not measured. Untangle solves the accuracy issue by relying on its technical support. But as mentioned, such approach is not scalable.

To solve issues regarding the accuracy of categories, (Ravikumar, McAFee, & Tomkins, 2009) use the same strategy offered by OpenDNS in their web filtering system. However, they proposed to solve the deficiency of OpenDNS (i.e., accuracy of the votes) by having a system that rates the raters. Reputation systems in MANETs discussed earlier can be used as a basis in developing such a strategy. Websites categorized by registered users are treated as contribution and votes made by others are treated as

ratings. By adopting reputation systems in MANET, the contributors as well as the raters are encouraged to cooperate.

Rater-Rating, a reputation system for a collaborative web-based system, attempts to encourage cooperation by adopting concepts in MANET (Pantola et al., 2010). It explores the use of a user-driven reputation system that measures both the contributor and rater reputation of users of a collaborative web system. However, Rater-Rating is consensus dependent. The effectiveness of the system relies heavily/primarily on the opinion of the majority. If the number of malicious users are more than their good counterparts, the reputation system can be manipulated. In other words, the system can mistakenly give malicious users a high reputation value relative to the reputation of good users.

THE TULUNGAN ALGORITHM

This section discusses the algorithm Tulungan. The discussion is divided into four sections: overview, notation, tuple definition, and actual algorithm.

3.1 Overview

The algorithm considers several entities such as URLs, categories, users, contributions, and ratings.

A URL, *w* represents a webpage or website (e.g., www.nba.com). Each URL has a set of categories.

A category *a* specifies a category that a URL can have such as porno- graphic site, violent site, search engine, sports site, etc. A category for a particular URL has three possible levels: positive (+) negative (-), and unknown (0). As an example, if the URL www.nba.com has a category of sports site with a level α of positive, it means that www.nba.com is categorized as a sports site. In the same way, if it has a category of pornographic with a level of negative, it means that www.nba.com is not a pornographic site.

There are two types of URLs used in the algorithm: control and unverified. The control URLs are those URLs whose category levels are known already (e.g. α is positive or negative). Unverified URLs have category levels that are unknown (e.g., α is unknown).

A user *u* provides contribution *c* and rating *r* in the algorithm.

The contribution provided by a user is in the form of a URL, category, and answer. The answer represents the level of the category that the user perceives that the URL possesses. As an example, if a user contributes the following: www.nba.com (URL), violent site (category), and negative (answer), it means that the user perceives www.nba.com as a non-violent site.

The rating provided by a user is also in the form of a URL, category, and answer. However, unlike a contribution, the answer in a rating signifies if a rater agrees or disagrees with the contribution. As an example, if a user rates the following: www.nba.com (URL), violent site (category), and negative (answer), it means that the user disagrees that www.nba.com is a non-violent site.

Ratings performed by users are grouped into three URLs. Two of the URLs are controls while the third one is unverified. The purpose of such grouping is discussed in the succeeding sections.

3.2 Notation

The different entities (e.g., user and contribution) used in the algorithm are tuples. A tuple is denoted by an italicized small letter (e.g., *u* and *c*). A set of tuples is denoted by a bold capital letter (e.g., **U** and **C**).

An element of a tuple is referred by specifying the symbol of the tuple, followed by a dot (.), and followed by the symbol of the element. As an example, to refer to the contributor reputation (ρ_C)of the user *u*, the following notation is used: $u.\rho_C$.

Unlike elements which are preceded by the symbol of the tuple, constants are not preceded by any symbol. As an example ρ_{cmax}, which is the maximum user contributor reputation, is denoted without the symbol *u*.

Functions are denoted by characters followed by parameters enclosed in parentheses (e.g., $cs(s, \alpha, t_i, t_f)$ and $\mathbf{R(u, s, \alpha, t_i, t_f)}$). Two types of functions are defined in this paper: functions that return a scalar value and functions that return a set. Functions that return a scalar value such as $cs(s, \alpha, t_i, t_f)$ use italicized small letters. Functions that return a set use a bold capital letter for its name such as $\mathbf{R(u, s, \alpha, t_i, t_f)}$. Aside from this, the name of the function determines the superset of the set returned by the function. As an example, $\mathbf{R(u, s, \alpha, t_i, t_f)}$. is a function that returns a set that is a subset of **R**.

3.3 Tuple Definition

The tuples used in this section are summarized in Table 1.

Table 1. Tuple Definition

Name	Symbol
user	*u*
URL category	s
contribution	*c*
potential rater	*p*
rating group	*g*
rating	*r*

A **user** *u* is a tuple defined as

$$u = <\rho_c, \rho_r, \rho_o> \quad (1)$$

where

ρ_c contributor reputation
ρ_r rater reputation
ρ_o overall reputation

The value of the contributor reputation penalty multiplier should satisfy the condition below.

$$\rho_{cpm} < \frac{1}{\rho_{crm}} \quad (2)$$

This discourages contributors to randomly provide contributions since a randomly made contribution has a 50% chance of being correct.

Table 2 summarizes the user-related constants used in this document.

Table 2. User-Related Constants

Symbol	Description
ρ_{crm}	user contributor reputation reward multiplier
ρ_{cpm}	user contributor reputation penalty multiplier
ρ_{ci}	initial user contributor reputation
ρ_{cmax}	maximum user contributor reputation
ρ_{cmin}	minimum user contributor reputation
ρ_{rrm}	user rater reputation reward multiplier
ρ_{rlpm}	user rater reputation lazy penalty multiplier
ρ_{ripm}	user rater reputation incorrect penalty multiplier
ρ_{ri}	initial user rater reputation
ρ_{rmax}	maximum user rater reputation
ρ_{rmin}	minimum user rater reputation

Similarly, the value of the rater reputation lazy penalty multiplier ρ_{rlpm} should satisfy the following condition.

$$\rho_{cpm} < \left(\frac{1}{\rho_{crm}}\right)^0 \quad (3)$$

This makes sure that even if only one of the three URLs (refer to the discussion of the rating group tuple) that needs to be rated is incorrect, its corresponding penalty is more than the effect of a single reward. This will discourage malicious raters to intentionally make the rating of two URLs correct and the third one incorrect.

In addition, the rater reputation incorrect penalty multiplier should be smaller than the rater reputation lazy penalty multiplier ($\rho_{ripm} < \rho_{rlpm}$) in order to encourage raters to rate.

A **URL category** *s* is a 5-tuple defined as

$$s = < w, a, \mathrm{j}_c, \mathrm{j}_r, \alpha > \quad (4)$$

where

- w URL
- a category
- j_c contribution reputation
- j_r rating reputation
- α level

Table 3 summarizes the URL category-related constants used in this document.

Table 3. URL Category-Related Constants

Symbol	Description
j_{cat}	URL category contribution reputation absolute threshold
j_{rct}	URL category rating reputation credibility threshold
j_{rdt}	URL category rating reputation difference threshold

A **contribution** c is a 4-tuple defined as

$$c =< u, s, \alpha | t > \quad (5)$$

where

- u user who provides the contribution
- s URL category
- α answer (i.e., +1 means $s.w$ is categorized as $s.a$, -1 otherwise)
- t time the contribution was made

A **potential rater** p is a 6-tuple defined as

$$p =< u, w_{ca}, w_{cb}, w_x, t_i, t_f > \quad (6)$$

- u user who can provide a rating
- w_{ca} 1st URL used as control
- w_{cb} 2nd URL used as control
- w_x URL with unverified categories
- t_i time user is requested to rate
- t_f expiration time of request

A **rating group** g is a 5-tuple defined as

$$g =< u, w_{ca}, w_{cb}, w_x, t > \quad (7)$$

where

- u user who provides the rating
- w_{ca} 1st URL used as control
- w_{cb} 2nd URL used as control
- w_x URL with unverified categories
- t time rating was made

The group rating accuracy threshold â is the only rating group-related constant used in the algorithm.

A **rating** r is a triple defined as

$$r =< g, s, \alpha > \quad (8)$$

where

- g rating group
- s URL category
- α answer

3.4 Algorithm

Algorithm 1 enumerates the steps involved in the Tulungan Reputation System.

Algorithm 1 Tulungan Algorithm

1: Initialize the contribution reputation φ_c, rating reputation φ_r , and level α of all URL categories s in **S**.
2: Initialize the contributor reputation ρ_c and rater reputation ρ_r of user u and add it in **U.**
3: Allow all users u to add contribution c in **C.**
4: Determine potential raters p and add them in **P.**
5: Allow all users u that are potential raters to add rating group g in **G** and rating r in **R.**
6: Update the rating reputation φ_r of all URL categories s in **S.**
7: Update the contribution reputation φ_c of all URL categories s in **S.**
8: Update the level α of all URL categories s in **S.**
9: Update the contributor reputation ρ_c of all users u in **U.**
10: Update the rater reputation ρ_r of all users u in **U.**
11: Update the overall reputation ρ_o of all users u in **U.**

1. Initialize the contribution reputation φ_c, rating reputation φ_r, and level α of all URL categories *s* in **S.**

   Since the URL categories *s* are not yet determined (i.e., α is unknown), the contribution reputation φ_c and rating reputation φ_r of all URL categories s in **S** are initialized to 0. Aside from this, the URL category level α is also initialized to 0 to denote that the level for the URL category is unknown.

$$si.\varphi_c = 0 \forall si \in \mathbf{S} \quad (9)$$

$$si.\varphi_r = 0 \forall si \in \mathbf{S} \quad (10)$$

$$si.\alpha = 0 \forall si \in \mathbf{W} \quad (11)$$

2. Initialize the contributor reputation ρ_c and rater reputation ρ_r of user *u* and add it in **U.**

   Unlike the contribution reputation and rating reputation of the URL category, the contributor reputation ρ_c and rater reputation ρ_r of a new user *u* are initialized to ρ_{ci} and ρ_{ri}

$$u.\rho_c = \rho_{ci} \quad (12)$$

$$u.\rho_r = \rho_{ri} \quad (13)$$

   Once the reputation of user *u* is initialized, it is added to **U**.

   This step is performed everytime a new user registers in the reputation system.

3. Allow all users *u* to add contribution *c* in **C.**

   All users are allowed to contribute a contribution c regardless of his/her contributor and rater reputation. However, the elements *c.u* and *c.s* of the contribution *c* should have values that satisfy the condition $C(\mathbf{u}, \mathbf{s}) = \emptyset$. This ensures that a particular user *u* will only contribute a single contribution that involves a particular URL category *s*. This prevents users from contributing the same contribution.

$$\mathbf{C}(u, s) = \{c | c.u = \iota \wedge c.s = s\} \quad (14)$$

4. Determine potential raters *p* and add them in **P.**

   It is essential that the contributions that will be rated are prioritized such that those contributions with a higher probability of being correct are assigned with enough raters. To have enough raters for contributions involving a particular URL w_x and time frame t_i to t_f, the condition below must be satisfied.

$$2\alpha_{rct} \leq \sum (p_i.u.\rho_r)^2 \quad \text{where } p_i \in \mathbf{P(wx, ti, tf)} \quad (15)$$

$$\mathbf{P}(w_x, t_i, t_f) = \{p | p.w_x = w_x \wedge p.t_i = t_i \wedge p.t_f = t_f\} \quad (16)$$

   In order to determine which contributions have higher probability of being correct, the contribution reputation of each contribution can be computed (please refer to step 7 on computing the contribution reputation). However, since computing the contribution reputation requires that the contributions are rated already, computations that involve information on ratings are ignored. Specifically, equations in step 7 that refers to the rating reputation update condition (refer to Eq. 18) assumes that *z(s)* results to a value of *true.*

5. Allow all users *u* that are potential raters to add rating group *g* in **G** and rating *r* in **R.**

   Take note that the previous step determines the contributions that a particular user should rate. This prevents a user from choosing a particular contribution to rate. This decreases the chance that a particular user will intentionally and incorrectly rate a particular contribution (e.g., giving a +1 to a wrong contribution just because it is contributed by a friend).

   Aside from this, for each rating group, categories of three URLs are rated (i.e., two control URLs w_{ca} and w_{cb}, and an unknown URL w_x). The two control URLs, as the name implies, are used only as controls. The URL categories of these controls are actually known by the system. Since the URL categories are already known, the reputation

system can determine already if the ratings made by a user to the control URLs are correct. This aids the reputation system in gauging if a user is performing the rating seriously. If the ratings of the controls are wrong, then the reputation system will assume also that the user's rating for the unknown URL is also wrong. This approach is based from reCAPTCHA, where in users are asked to type the two words (i.e., an unverified word and a control word) that are presented to them *(reCAPTCHA - Stop Spam, Read Books[Online]* , n.d.).

Similar to reCAPTCHA, the three URLs are presented to the user for rating in a way that the user has no idea which are the controls and whichis the unknown. This prevents malicious raters from making the rating of the two controls correct while intentionally making the rating of the unknown URL wrong.

6. Update the rating reputation φ_r of all URL categories *s* in **S.**

A summary of the functions that will be used in computing the rating reputation is presented in Table 4.

The rating reputation φ_r of the URL category *s* is computed as follows:

$$s.\,\mathrm{j}_r = \begin{cases} r\,\mathrm{j}_w(s, t_i, t_f) & \text{if } z(s) \\ s.\,\mathrm{j}_r & \text{otherwise} \end{cases} \qquad (17)$$

where

ti initial time to consider
tf final time to consider

The function $r\,\mathrm{j}_w(s, t_i, t_f)$ computes the winning rating reputation. Users provide a set of ratings *r* on a particular URL category *s*. The set of ratings is divided into two groups: those that are positive ratings (i.e., $r.\alpha = +1$) and those that are negative ratings (i.e., $r.\alpha =$ "1). The rating reputation of both groups of ratings are computed. The higher (and therefore the winning) rating reputation is returned by the function $r\varphi_w(s, t_i, t_f)$.

Take note that the function $r\alpha_w(s, ti, tf)$ serves as the value of the rating reputation $s.\,\mathrm{j}_r$ only if the conditions in $z(s)$ are satisfied otherwise the current value of $s.\,\mathrm{j}_r$ is retained. The function $z(s)$ is defined as

$$\begin{aligned} z(s) = {} & (r\,\mathrm{j}_w(s, t_i, t_f) \geq \mathrm{j}_{rct}) \wedge \\ & (r\,\mathrm{j}_w(s, t_i, t_f) - r\,\mathrm{j}_l(s, t_i, t_f) \geq \mathrm{j}_{rdt}) \wedge \\ & (rv_w(s, t_i, t_f) = +1) \end{aligned} \qquad (18)$$

Table 4. Rating Reputation Function Description

Name	Description
$z(s)$	rating reputation update condition
$r\alpha_w(s, t_i, t_f)$	winning ratingreputation
$r\alpha(s, t_i, t_f)$	losing rating reputation
$rc_w(s, t_i, t_f)$	winning rating count
$rc_l(s, t_i, t_f)$	losing rating count
$rc_p(w_x, t_i, t_f)$	positive rating count
$rc_n(s, t_i, t_f)$	negative rating count
$rc(s, \alpha, t_i, t_f)$	rating count
$v(g)$	URL controls correct condition
$ra(g, w)$	rating accuracy
$rv_w(s, t_i, t_f)$	winning rating vote
$rs_w(s, t_i, t_f)$	winning rater reputation summation
$rs_l(s, t_i, t_f)$	losing rater reputation summation
$rs_p(s, t_i, t_f)$	positive rater reputation summation
$rs_n(s, t_i, t_f)$	negative rater reputation summation
$rs(s, \alpha, t_i, t_f)$	rater reputation summation

The conditions are explained as follows:

(i) $r\,\mathrm{j}_w(s, t_i, t_f) \geq \mathrm{j}_{rct}$
This ensures that the rating reputation of the winning rating $r\alpha_w(s, t_i, t_f)$ is credible enough to be considered since it reaches the URL category rating reputation credibility threshold j_{rct}.

(ii) $r\,\mathrm{j}_w(s, t_i, t_f) - r\,\mathrm{j}_l(s, t_i, t_f) \geq \mathrm{j}_{rdt}$
This ensures that difference between the rating reputation of the winning rating and the losing

rating is large enough such that a false winner is prevented. The URL category rating reputation difference threshold φ_{rdt} is used to determine if the difference is large enough.

(iii) $rv_w(s, t_i, t_f) = +1$
This ensures that the rating winning vote $rv_w(s, t_i, t_f)$ agrees with the contribution.

The winning rating reputation $r\varphi_w(s, t_i, t_f)$ is defined as follows:

$$r\varphi_w(s, t_i, t_f) = \begin{cases} log(x)\, y & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where
$x = rc_w(s, t_i, t_f)$
$y = rs_w(s, t_i, t_f)$ (19)

This function uses the winning rater reputation summation $rs_w(s, t_i, t_f)$. However, using this as the only basis is not sufficient. A single malicious rater that intentionally increase its rater reputation to eventually cause problems in the future can manipulate the value of $r\varphi_w(s, t_i, t_f)$ if $rs_w(s, t_i, t_f)$ is the only basis. To ensure that no single user can manipulate the result, the winning rating reputation function also considers the winning rating count $rc_w(s, t_i, t_f)$. The winning rating count is used as a parameter in a logarithmic function to ensure two things. First, if the count is less than 10 (e.g., only one rater), then the effect of the rating is very small. As mentioned earlier, this prevents few raters from manipulating the result. Second, even if there are many new malicious users (e.g., 20), the count is not enough to provide a significant effect. This makes it harder for malicious users to plan a Sybil attack since they still need to increase their rater reputation before they have a significant effect.

The losing rating reputation $r\varphi_l(s, t_i, t_f)$ is defined as follows:

$$r\varphi_l(s, t_i, t_f) = \begin{cases} log(x)\, y & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where
$x = rc_l(s, t_i, t_f)$
$y = rs_l(s, t_i, t_f)$ (20)

The computation of the losing rating reputation is similar to its winning counterpart.

The winning rating count $rc_w(s, t_i, t_f)$ as well as the losing rating count $rc_l(s, t_i, t_f)$ are computed by comparing the positive and the negative rater reputation summation. The rater count of the higher rater reputation summation is returned by the winning rating count function while the lower one is returned by the losing rating count function. The two functions are defined as follows:

$$rc_w(s, t_i, t_f) = \begin{cases} rc_p(s, t_i, t_f) & \text{if } rs_p(s, t_i, t_f) > rs_n(s, t_i, t_f) \\ rc_n(s, t_i, t_f) & \text{if } rs_n(s, t_i, t_f) > rs_p(s, t_i, t_f) \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

$$rc_l(s, t_i, t_f) = \begin{cases} rc_p(s, t_i, t_f) & \text{if } rs_p(s, t_i, t_f) < rs_n(s, t_i, t_f) \\ rc_n(s, t_i, t_f) & \text{if } rs_n(s, t_i, t_f) < rs_p(s, t_i, t_f) \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

The positive rating count $rc_p(s, t_i, t_f)$ and its inverse, the negative rating count $rc_n(s, t_i, t_f)$, use the rating count function $rc(s, \alpha, t_i, t_f)$ to compute the number of raters that provide positive and negative rating. These functions are shown below.

$$rc_p(s, t_i, t_f) = rc(s, +1, t_i, t_f) \quad (23)$$

$$rc_n(s, t_i, t_f) = rc(s, -1, t_i, t_f) \quad (24)$$

The rating count function $rc(s, \alpha, t_i, t_f)$ is indicated below.

$$rc(s, \alpha, t_i, t_f) = |\mathbf{U_r}(\mathbf{s}, \alpha, \mathbf{t_i}, \mathbf{t_f})| \quad (25)$$

The rating count function simply counts the number of elements returned by the $\mathbf{U_r\ (s,\ \alpha,\ t_i,\ t_f)}$ function.

The $\mathbf{U_r\ (s,\ \alpha,\ t_i,\ t_f)}$ function returns a set of users that rated the URL category s with an level α. This function is defined as follows:

$$\mathbf{U_r\ (s,\ \alpha,\ t_i,\ t_f\ )} = \{u \mid \mathbf{R(u,\ s,\ \alpha,\ t_i,\ t_f\ )} \div \mathbf{\emptyset}\} \quad (26)$$

The function $\mathbf{R(u,\ s,\ \alpha,\ t_i,\ t_f)}$ is utilized by the function above. This function returns all ratings rated by user *u*, and rated the URL category *s* with an answer *α*. This function is defined below.

$$\mathbf{R(u,\ s,\ \alpha,\ t_i,\ t_f\ )} = \{r \mid r.g.u = u \wedge\ r.s = s \wedge\ r.\alpha = \alpha \wedge\ r.g.t\ [t_i\ ..t_f\ ) \wedge\ v(r.g)\} \quad (27)$$

The R(u, s, α, t_i, t_f) function ensures that only ratings that correctly answered the control URLs (i.e., r.g.w_{ca} and r.g.w_{cb}) are considered. This is checked by the function v(g) shown below.

$$v(g) = ra(g,\ g.w_{ca}) \geq \beta \dot{e}\ ra(g,\ g.w_{cb}) \geq \beta \quad (28)$$

The function above is based on the rating accuracy $ra(g,\ w)$ of the two control URLs: $g.w_{ca}$ and $g.w_{cb}$. The rating accuracy for both controls are compared to the group rating accuracy threshold β.

The rating accuracy $ra(g,\ w)$ function is indicated below.

$$ra(g,\ w) = \begin{cases} \frac{|\mathbf{R}_{correct}(g,w)|}{|\mathbf{R}_{all}\ (g,w)|} & \text{if } |\mathbf{R_{all}(g,\ w)}| > 0 \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

The function considers the number of correct ratings performed on a particular URL w in a rating group g. This can be accomplished by getting the number of elements of the set below.

$$\mathbf{R_{correct}(g,\ w)} = \{r \mid r.g = g \wedge\ r.s.w = w \wedge\ r.s.\alpha \div 0 \wedge\ r.s.\alpha = r.\alpha\} \quad (30)$$

Aside from this, it also considers all the ratings (i.e., correct and incorrect) on the same URL w in the same rating group g. Similarly, this is accomplished by getting the number of elements of the set below.

$$\mathbf{R_{all}(g,\ w)} = \{r \mid r.g = g \wedge\ r.s.w = w \wedge\ r.s.\alpha \div 0\} \quad (31)$$

The winning rating vote $rv_w\ (s,\ t_i,\ t_f\ )$ determines which group of rating (i.e., positive or negative) has the higher rater reputation summation. The function is defined below.

$$rv_w\ (s,\ t_i,\ t_f\ ) = \begin{cases} +1 \text{ if } rs_p\ (s,\ t_i,\ t_f) > rs_n\ (s,\ t_i,\ t_f) \\ -1 \text{ if } rs_n\ (s,\ t_i,\ t_f\ ) > rs_p(s,\ t_i,\ t_f) \\ 0 \quad \text{otherwise} \end{cases} \quad (32)$$

The winning rater reputation $rs_w\ (s,\ t_i,\ t_f)$ returns the rater reputation summation of the winning rating vote. The function is shown below.

$$rs_w\ (s,\ t_i,\ t_f) = \begin{cases} rs_p(s,\ t_i,\ t_f) & \text{if } rs_p\ (s,\ t_i,\ t_f) > rs_n\ (s,\ t_i,\ t_f) \\ rs_n(s,\ t_i,\ t_f) & \text{if } rs_n\ (s,\ t_i,\ t_f) > rs_p\ (s,\ t_i,\ t_f) \\ 0 & \text{otherwise} \end{cases} \quad (33)$$

The losing rater reputation rs$_l$(s, t_i, t_f) provides the opposite result of the winning rater reputation. The function is defined as follows:

$$rs_l\ (s,\ t_i,\ t_f) = \begin{cases} rs_p(s,\ t_i,\ t_f) & \text{if } rs_p\ (s,\ t_i,\ t_f) < rs_n\ (s,\ t_i,\ t_f) \\ rs_n(s,\ t_i,\ t_f) & \text{if } rs_n\ (s,\ t_i,\ t_f) < rs_p\ (s,\ t_i,\ t_f) \\ 0 & \text{otherwise} \end{cases} \quad (34)$$

The functions $rs_p\ (s,\ t_i,\ t_f)$ and $rs_n(s,\ t_i,\ t_f)$, which are the positive and negative rater reputation summation functions, respectively, use the rater reputation summation function $rs(s,\ \alpha,\ t_i,\ t_f)$ to compute the summation of the rater reputation of raters that provide positive and negative rating. These functions are defined below.

$$rs_p(s,\ t_i,\ t_f\ ) = rs(s,\ +1,\ t_i,\ t_f) \quad (35)$$

$$rs_n(s,\ t_i,\ t_f\ ) = rs(s,\ -1,\ t_i,\ t_f) \quad (36)$$

The rater reputation summation $rs(s, \alpha, t_i, t_f)$ is defined below.

$$rs(s, \alpha, t_i, t_f) = \sum_{1} (u_i.\rho_r)^2$$

$$\text{where } u_i \in \mathbf{U_r}\ (\mathbf{s}, \alpha, \mathbf{t_i}, \mathbf{t_f}) \quad (37)$$

Instead of adding only the rater reputation of the users ($u_i.\rho_r$), their square ($(u_i.\rho_r)^2$) are added. This makes the effect of low (i.e., reputation less than 1) reputation less significant and in the same way increases the effect of high reputation (i.e., reputation greater than 1).

7. Update the contribution reputation j_c of all URL categories s in **S.**

A summary of the functions that will be used in computing the contribution reputation is presented in Table 5.

The contribution reputation j_c of the URL category s is equal to the winning contribution reputation $c j_w (s, t_i, t_f)$. However, two conditions must be satisfied before the winning contribution reputation is used:

Table 5. Contribution Reputation Function Description

Name	Description
$c j_w (s, t_i, t_f)$	winning contribution reputation
$cc_w (s, t_i, t_f)$	winning contribution count
$cc_l (s, t_i, t_f)$	losing contribution count
$cc_p (s, t_i, t_f)$	positive contribution count
$cc_n (s, t_i, t_f)$	negative contribution count
$cc(s, \alpha, t_i, t_f)$	contribution count
$cv_w (s, t_i, t_f)$	winning contribution vote
$cs_w (s, t_i, t_f)$	winning contributor reputation summation
$cs_l (s, t_i, t_f)$	losing contributor reputation summation
$cs_p (s, t_i, t_f)$	positive contributor reputation summation
$cs_n (s, t_i, t_f)$	negative contributor reputation summation
$cs(s, \alpha, t_i, t_f)$	contributor reputation summation

$z(s)$ and the current contribution reputation of the URL category $s.j_c$ should be less than the URL category contribution reputation absolute threshold j_{cat}.

$$s.j_c = \begin{cases} c j_w(s, t_i, t_f) & \text{if } s.j_c < j_{cat} \cdot z(s) \\ s.j_c & \text{otherwise} \end{cases} \quad (38)$$

The winning contribution reputation $c j_w (s, t_i, t_f)$ is defined as follows:

$$c j_w (s, t_i, t_f) = \begin{cases} [log(x)]^x \ y & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where (39)

$x = cc_w (s, t_i, t_f)$

$y = cs_w (s, t_i, t_f)$

The equation above uses another function $cs_w(s,t_i, t_f)$, which is the winning contributor reputation summation. However, using this as the only basis is not sufficient. A single malicious contributor that intentionally increase its contributor reputation to eventually cause problems in the future can manipulate the value of $c\varphi_w(s, t_i, t_f)$, if $cs_w(s, t_i, t_f)$ is the only basis. To ensure that no single user can manipulate the result, the winning contribution reputation function also considers the winning contribution count $cc_w(s, t_i, t_f)$. Similar to its rating counterpart, the winning contribution count is used as a parameter in a logarithmic function to ensure two things. First, if the count is less than 10 (e.g., only one contributor), then the effect of the contribution is very small. As mentioned earlier, this prevents few contributors from manipulating the result. Second, even if there are many new malicious users (e.g., 20), the count is not enough to provide a significant effect. This makes it harder for malicious users to plan a Sybil attack since they still need to increase their contributor reputation before they have a significant effect.

The winning contribution count $cc_w(s, t_i, t_f)$ and losing contribution count $cs_l(s, t_i, t_f)$ are computed by comparing the positive and the negative contributor reputation summation. The contributor count of the higher contributor reputation summation is returned by the winning contribution count function while the lower one is returned by the losing contribution count function. The two functions are shown below.

$$cc_w (s, t_i, t_f) = \begin{cases} cc_p(s, t_i, t_f) & \text{if } cc_p (s, t_i, t_f) > cc_n(s, t_i, t_f) \\ cc_n (s, t_i, t_f) & \text{if } cc_n (s, t_i, t_f) > cc_p (s, t_i, t_f) \\ 0 & \text{otherwise} \end{cases} \quad (40)$$

$$cc_l (s, t_i, t_f) = \begin{cases} cc_p(s, t_i, t_f) & \text{if } cc_p (s, t_i, t_f) < cc_n(s, t_i, t_f) \\ cc_n (s, t_i, t_f) & \text{if } cc_n (s, t_i, t_f) < cc_p (s, t_i, t_f) \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

The positive contribution count $cc_p\ (s,\ t_i,\ tf)$ and negative contribution count $cc_n\ (s,\ t_i,\ tf)$ functions use the contribution count function $cc(s,\ \alpha,\ t_i,\ tf)$ to compute the number of contributors that provide positive and negative contributions. These functions are defined below.

$$cc_p(s,\ t_i,\ t_f) = cc(s,\ +1,\ t_i,\ tf) \tag{42}$$

$$cc_n(s,\ t_i,\ t_f) = cc(s,\ -1,\ t_i,\ tf) \tag{43}$$

The contribution count function $cc(s,\ \alpha,\ t_i,\ tf)$ is defined below.

$$cc(s,\ \alpha,\ t_i,\ t_f) = |\ \mathbf{U_c(s,\ \alpha,\ t_i, t_f)}\ | \tag{44}$$

The contribution count function simply counts the number of elements returned by the $\mathbf{U_c(s,\ \alpha,\ t_i,\ tf)}$ function.

The $\mathbf{U_c(s,\ \alpha,\ t_i,\ t_f)}$ function returns a set of users whose contribution involved URL category s, answered α, within the time frame t_i to tf.

$$\mathbf{U_c(s,\ \alpha,\ t_i,\ t_f)} = \{u |\ \mathbf{C(u,\ s,\ \alpha,\ t_i,\ t_f)} \neq \varnothing\} \tag{45}$$

The function $\mathbf{C(u,\ s,\ \alpha,\ t_i,\ tf)}$ is utilized by the function above. This function returns all contributions contributed by user u, pertaining to URL category s with an answer α. This function is defined below.

$$\begin{aligned}\mathbf{C(u,\ s,\ \alpha,\ t_i,\ tf)} = \{c\ /\ & c.u = u \wedge \\ & c.s = s \wedge \\ & c.\alpha = \alpha \wedge \\ & c.t\ [t_i\ ..tf)\}\end{aligned} \tag{46}$$

The winning contributor reputation $cs_W\ (s,\ t_i,\ tf)$ returns the contributor reputation summation of the winning contribution vote. The function is defined below.

$$cs_W\ (s,\ t_i,\ tf) = \begin{cases} cs_p(s,\ t_i,\ tf) & \text{if } cs_p\ (s,\ t_i,\ tf) > cs_n(s,\ t_i,\ tf) \\ cs_n\ (s,\ t_i,\ tf) & \text{if } cs_n\ (s,\ t_i,\ tf) > cs_p\ (s,\ t_i,\ tf) \\ 0 & \text{otherwise} \end{cases} \tag{47}$$

The losing contributor reputation $cs_l\ (s,\ t_i,\ tf)$ provides the opposite result of the winning contributor reputation. The function is shown below.

$$cs_l\ (s,\ t_i,\ tf) = \begin{cases} cs_p(s,\ t_i,\ tf) & \text{if } cs_p\ (s,\ t_i,\ tf) < cs_n(s,\ t_i,\ tf) \\ cs_n\ (s,\ t_i,\ tf) & \text{if } cs_n\ (s,\ t_i,\ tf) < cs_p\ (s,\ t_i,\ tf) \\ 0 & \text{otherwise} \end{cases} \tag{48}$$

The functions $cs_p\ (s,\ t_i,\ tf)$ and $cs_n\ (s,\ t_i,\ tf)$, which are the positive and negative contributor reputation summation functions, respectively, use the contributor reputation summation function $cs(s,\ \alpha,\ t_i,\ tf)$ to compute the summation of the contributor reputation of contributors that provide positive and negative contribution. These functions are defined below.

$$cs_p(s,\ t_i,\ tf) = cs(s,\ +1,\ t_i,\ tf) \tag{49}$$

$$cs_n\ (s,\ t_i,\ tf) = cs(s,\ -1,\ t_i,\ tf) \tag{50}$$

The contributor reputation summation $cs(s,\ \alpha,\ t_i,\ tf)$ is shown below.

$$cs(s,\ \alpha,\ t_i,\ tf) = \sum (u_i.\rho_C)^2$$

$$\text{where } u \in \mathbf{U_c\ (s,\ \alpha,\ t_i,\ tf)} \tag{51}$$

Similar to its rating counterpart, instead of adding only the contributor reputation of the users $(u_i.\rho_C)$ their square $((u_i.\rho_C)^2)$ are added.

8. Update the level α of all URL categories s in **S**.

The level α of the URL category s is equal to the winning contribution vote $cv_W\ (s,\ t_i,\ tf)$. Similar to the φ_C of the URL category s, the condition involving $z(w)$ and the URL category contribution reputation absolute threshold φ_{Cat} must be met.

$$s.\alpha = \begin{cases} cv_W\ (s,\ t_i,\ tf) & \text{if } s.\varphi_C < \varphi_{Cat} \wedge z(w) \\ s.\alpha & \text{otherwise} \end{cases} \tag{52}$$

The winning contribution vote winning contribution vote $cv_W\ (s,\ t_i,\ tf)$ determines which group of contribution (i.e., positive and negative) has the higher contributor reputation summation. The function is defined below:

$$cv_W(s, t_i, t_f) = \begin{cases} +1 & \text{if } cs_p(s, t_i, t_f) > cs_n(s, t_i, t_f) \\ -1 & \text{if } cs_n(s, t_i, t_f) > cs_p(s, t_i, t_f) \\ 0 & \text{otherwise} \end{cases} \quad (53)$$

9. Update the contributor reputation ρ_C of all users u in **U.** The contributor reputation ρ_C of user u is computed as follows:

$$u.\rho_c = u.\rho_c \bar{e}_i \begin{cases} \rho_{crm} & \text{if } c_i.\alpha = c.s.\alpha \dot{e} \; c.s.\alpha \neq 0 \\ \rho_{crm} & \text{if } c_i.\alpha \neq c.s.\alpha \dot{e} \; c.s.\alpha \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (54)$$

where $c_i \in$ **C(u, ti , tf)**

To update the contributor reputation, contributions made by user u are considered. In addition, only contributions whose URL category s is classified (i.e., $c.s.\alpha \div 0$) are considered. If $c_i.\alpha$ matches c.s.α, then the contribution is considered correct and the user is rewarded by increasing its contributor reputation by multiplying it with the user contributor reputation reward multiplier ρ_{crm}. Otherwise, it is penalized with the user contributor reputation penalty multiplier ρ_{cpm}.

To ensure that the contributor reputation is within the range, it is compared with the maximum user contributor reputation ρ_{cmax} and the minimum user contributor reputation ρ_{cmin} as shown below.

$$u.\rho_C = max(\rho_{cmin}, \min(\rho_{cmax}, u.\rho_C)) \quad (55)$$

To get the set of contributions made by a particular user u, the following function is used:

$$\mathbf{C(u, t_j, t_f)} = \{c \mid c.u = u \; \dot{e} \; c.t \in [t_i .. t_f)\} \quad (56)$$

10. Update the rater reputation ρ_r of all users u in **U.** The rater reputation ρ_r of user u is computed as follows:

$$u.\rho_r = u.\rho_r \bar{e}_i \; r\rho(p_i)$$

where $p_i \in$ **P(u, ti, tf)** (57)

Similar to the contributor reputation, to ensure that the rater reputation is within the range, it is compared with the maximum user rater reputation ρ_{rmax} and the minimum user rater reputation ρ_{rmin} as shown below.

$$u.\rho_r = max(\rho_{rmin}, min(\rho_{rmax}, u.\rho_r)) \quad (58)$$

To update the rater reputation, the potential ratings that should be made by user u are considered. These potential ratings may be derived from the subset of **P** and is defined in the following function:

$$\mathbf{P(u, t_i, t_f)} = \{p \mid p.u = u \dot{e} \; p.t_i \in [t_i .. t_f) \dot{e} \; p.t_f \in [t_i .. t_f) \} \quad (59)$$

Each potential rating is examined if the user did in fact perform the rating. If the potential rating is not found in **G,** then it means the user failed to rate. The user is penalized by multiplying its current rater reputation with the user rater reputation lazy penalty multiplier ρ_{rlpm}. The penalty multiplier is cubed ($(\rho_{rlpm})^3$) since in a single potential rating, three URLs need to be rated (i.e., w_{ca}, w_{cb}, and w_x). If the user successfully rated, its rating is examined. This process is summarized in the two function below.

$$r\rho(p) = \begin{cases} rm(g_i) & \text{if } \{g \mid g \in \mathbf{G} \dot{e} \; m(g, p)\} \neq \emptyset \\ (\rho_{rlpm})^3 & \text{otherwise} \end{cases}$$

where $g_i \in G \dot{e} m(g_i, p)$

$$m(g, p) = (g_i.w_{ca} = p.w_{ca}) \dot{e} \; (g_i.w_{cb} = p.w_{cb}) \dot{e} \; (g_i.w_x = p.w_x) \dot{e} \; (g.u = p.u) \quad (61)$$

The function $rm(g)$ is used to examine the rating. The three URLs involved in the rating are examined separately.

$$rm(g) = rmc(g, g.w_{ca}) rmc(g, g.w_{cb}) \bar{e}_i \; rmx(r_i) \quad (62)$$

where $r_i \in$ **R(g, g.wx)**

The function rmc(g, w) is used to examine the two control URLs. It simply checks if the rating accuracy ra(g, w) meets the group rating accuracy threshold β. If it is greater than or equal the threshold, the user is rewarded by multiplying its rater reputation with the user rater reputation reward multiplier ρ_{rrm}. Otherwise, it is multiplied with the user rater reputation incorrect penalty multiplier ρ_{ripm}.

$$rmc(g,w) = \begin{cases} \rho_{rrm} & \text{if } ra(g,w) \geq \beta \\ \rho_{ripm} & \text{otherwise} \end{cases} \quad (63)$$

Similar to *rmc(g, w),* the function *rmx(r)* examines the unverified URL. However, since it is unverified, it is not appropriate to rely on the rating accuracy function *ra(g, w)*. The set of ratings that is related to rating group *g* and the unknown URL w_X must be considered. This can be derived using the function below.

$$\mathbf{R(g, w)} = \{r / r.g = g \text{ ė } r.s.w = w\} \quad (64)$$

For each rating *r* in $\mathbf{R(g, g.w_X)}$, a user is rewarded or penalized based on the $rv_W(r.s, t_i, t_f)$, which is the winning rating vote. This is defined in the function below.

$$rmx(r) = \begin{cases} \rho_{rrm} & \text{if } rv_W.(r.s, t_i, t_f) = \text{r. } \alpha \text{ ė } \\ & rv_W(r.s, t_i, t_f) \leq 0 \\ \rho_{ripm} & \text{otherwise} \end{cases} \quad (65)$$

11. Update the overall reputation ρ_O of all users *u* in **U.**

Tulungan ensures that if a user fails to be good in at least one of its roles as a contributor and rater, its overall reputation will be affected. This is accomplished by getting the product of the contributor and rater reputations of a user. As an example, if a user is good as a contributor and has a high contributor rating, but is delinquent in being a rater and thus gets a low rater rating, the user's overall reputation is also low.

The equation in deriving the overall reputation of a user is shown below.

$$u.\rho_o = \text{Ě } \overline{u.\rho_c\, u.\rho_r} \quad (66)$$

EVALUATION OF THE REPUTATION SYSTEM

4.1 Evaluation Set-up

The system is evaluated using a simulation. The different types of users are modelled based on the behaviour they are expected to have. The behaviour of a user is dictated by the type of contributor and rater he or she is. A contributor or a rater can either be good or bad (i.e., lazy, deviant, and malicious).

The behaviour of the different types of contributors and raters are discussed below.

A good contributor provides correct categorization of URLs most of the time. In the simulation, there is only 1 in 5000 chances a good contributor provides a wrong contribution.

A lazy contributor randomly chooses whether a URL falls under a particular category. As an example, a URL that has pornographic content has a 50% chance of being categorized as pornographic.

A deviant contributor provides incorrect categorization of URLs most of the time. It has a 1 in 5000 chances of providing a correct contribution.

A malicious contributor has the same behaviour as the deviant contributor.

The behaviour of the different types of raters are modelled similarly to their contributor counterpart. However, instead of providing contribution, the raters rates the contributions made by the contributors.

Only the malicious rater is modelled differently from its contributor counterpart. Since the rating process requires rating the URL categories of three URLs (i.e., two control URLs w_{ca} and w_{cb}, and an unknown URL w_x), a malicious rater needs to rate correctly the two control URLs and intentionally make an incorrect rating for the unknown. Giving a correct rating for the control URLs is essential, otherwise, the reputation system can detect that the rater is giving

a wrong rating and may be doing something malicious. However, since the malicious rater has no idea which of the three URLs are the controls and the unknown, it needs to make a guess. There is 1 in 3 chances that a malicious rater will correctly guess which is the unknown URL. Therefore, it also has 1 in 3 chances of being successful in giving a wrong rating for the unknown URL while remaining undetected by the reputation system.

To keep track of the user type (e.g., a good contributor and rater), the simulation uses two fields, *contributor_type* and *rater_type*, for each user. This is essential to determine the contribution and rating behaviour that a user will perform in the simulation. However, these fields are not used in the actual computation of the contributor, rater, and overall reputation of each user.

The simulation does not only cover the Tulungan reputation system but as well as Rater-Rating in order to compare the two systems in terms of differentiating good and bad users based on their reputation calculation. The simulation is divided into four (4) parts as illustrated in Table 6.

Table 6: Simulation Experiments

Type of Users Involved	Rater-Rating	Tulungan
Good vs. Bad	Part 1	Part 2
Good vs. Malicious	Part 3	Part 4

For each part, the simulation is executed with a varying percentage of good users relative to their bad counterpart. It starts with 5% good users and 95% bad (in the case of the 3rd and 4th part of the simulation, only the malicious users are used among the bad users). It is executed again with10% good and 90% bad users. This is repeated with an increment of 5% in the number of good users until the percentage of good users reach 95%.

For each execution, the simulation is composed of 500 users and runs for 366 simulated days (i.e., January 1 to December 31). Everyday, each user in the simulation provides one contribution. On the 1st day of each month, the potential raters are determined by the reputation system. On the 2nd day of each month, each user provides a rating based on the potential rater list generated on the 1st day. On the 28th day of each month, the contributor reputation *(ρc)* and rater reputation *(ρr)* of all the users are determined. At the end of the 366th day, the average of the contributor reputation, rater reputation, and overall reputation of all the good users is computed. The same thing is done with the reputation of each type of bad user. To get the average reputation for each user type, the fields *contributor_type* and *rater_type* are used. Take note that fields are used in getting the averge reputation and not in the computation of individual reputation of the users.

The maximum user contributor reputation (ρ_{cmax}) and rater reputation (ρ_{rmax}) are both set to 10.0. Similarly, the minimum user contributor reputation (ρ_{cmin}) and rater reputation (ρ_{rmin}) are both set to 0.001. The initial user contributor reputation (ρ_{ci}) and rater reputation (ρ_{ri}) are both set to 0.5.

4.2 Results and Analysis

4.2.1 Good vs. Bad Users (Parts 1 and 2)

Figures 1a, 1c, and 1e are the results of good versus bad users using Rater- Rating reputation system (part 1 results). This can be compared with Figures 1b, 1d, and 1f, which are the results of good versus bad users using Tujunga reputation system (part 2 results).

The contributor reputation and rater reputation of good users in part 1 are dependent on their number. The more good users there are, the higher their reputation compared to the bad users. Before the simulation result was available, it was expected that 50% of good users is needed to overcome the reputation of the bad users when Rater-Rating is used. However, as seen in Figures 1a and 1c, only 40% of good users is needed in order to give the former a higher reputation relative to their deviant and malicious counter- part. This relatively good performance of Rater-Rating can be attributed to the presence of lazy users. Since the lazy users randomly choose the category of URLs and randomly rate contributions, there may be cases that the contribution and rating done by lazy users

support the contributions and rating made by good users.

When it comes to Tulungan, the contributor reputation of good users are already higher than their bad counterpart when there are only 25% good users (refer to Figure 1b). In addition, even if there are only 5% good users, the rater reputation of good users are already higher than the lazy, deviant, and malicious users. This can be seen in Figure 1d.

There are cases where the rater reputation of good users in Rater-Rating and Tulungan decreases to a certain level (e.g., when there are 70% and 85% good users). This can be attributed to the composition of potential raters (e.g., a set of good raters are grouped with some bad users to rate a particular unknown URL). Aside from this, good raters have a 1 in 5000 chance of making incorrect rating which may contributed to the decrease in rater reputation.

In terms of overall reputation, when there is at least 60% bad users, the minimum average overall reputation given by Rater-Rating to deviant and malicious users is 1.5 and it goes as high as 4 (refer to Figure 1e). Tulungan, on the other hand, limits the overall reputation to less than 2.5 even if there are only 25% good users. This can be seen in Figure 1f.

Although Rater-Rating allows good users to outperform the deviant and malicious users even if there are less than 50% good users (i.e., only 40% good users are needed), this can be attributed to the presence of lazy users, as explained earlier. In order to confirm this, the good users are compared against only their malicious counterpart in parts 3 and 4 of the simulation.

4.2.2 Good vs. Malicious Users (Parts 3 and 4)

Figures 2a, 2c, and 2e are the results of good versus malicious users using Rater-Rating reputation system (part 3 results). This can be compared with Figures 2b, 2d, and 2f, which are the results of good versus malicious users using Tulungan reputation system (part 4 results).

As expected, the contributor reputation and rater reputation of good users in part 3 are dependent on their number. The more good users there are, the higher their reputation compared to their malicious counterpart. It can be noted in Figures 2a and 2c, both the contributor reputation and rater reputation of the good users are lower than the malicious users when the population of good users is less than 50%. This means that the Rater-Rating reputation system is effective only when the good users are more than the malicious users.

In part 4, a significant improvement in the contributor and rater reputation can be seen when the Tujunga reputation system is used. As seen in Figures 2b and 2d, even if there is only 1 good user for every 3 malicious users (or 25% good users), the Tujunga reputation system is still effective in giving the good users a higher reputation than their malicious counterpart.

Similar to parts 1 and 2, there are cases where the rater reputation of good users in Rater-Rating and Tujunga decreases to a certain level (e.g., when there are 85% good users in Rater-Rating and 50% and 65% in Tulungan). This can be attributed to the composition of potential raters and the small possibility of good raters in making incorrect rating.

In terms of overall reputation, when there are more malicious users compared to good users, the minimum average overall reputation given by Rater- Rating to malicious users is 3.7 and it goes as high as 6.2 (refer to Figure 2e). Tulungan, on the other hand, limits the overall reputation of malicious users to less than 1 even if there are only 25% good users. This can be seen in Figure 2f.

CONCLUSION AND FUTURE WORK

The simulation shows that Tulungan is capable of distinguishing good users from their bad counterpart even if majority of the population of users are bad. When good users are pitted against malicious users, Tulungan requires only 25% good users to be effective. This is a 100% improvement relative to Rater-Rating that requires 50% of the population to be good.

In terms of overall reputation, Tulungan limits the reputation of bad users (i.e., lazy, deviant, and malicious) to 2.5. This is 37.5% less than Rater-Rating where the overall reputation of bad users can go as high as 4. In addition, when the good users are compared against only to their malicious counterpart, the discrepancy of Tulungan and Rater-Rating is even more evident. Tulungan limits the overall reputation of malicious users to 83.88% less than Rater-Rating. In Tulungan, malicious users have a maximum average of overall reputation of less than 1, regardless of their percentage in the total user population. On the other hand, Rater-Rating allows malicious users to get an overall reputation of 6.2.

Tulungan can be used to effectively categorize the URLs of websites with- out the under- and over-blocking issues of the automatic approach as well as the scalability issue of the traditional manual method. More importantly, the system is still effective even if the number of good users involved in the categorization is fewer than their bad counterpart.

Although Tulungan is consensus-independent, it may be possible to manipulate this reputation system if the bad users collude. An improvement in Tulungan can be made in order to address scenarios when bad users create multiple accounts (e.g., phantom accounts). Such situation allows these multiple accounts to have a coordinated attack towards Tulungan, and as a result, these multiple accounts may be given a high reputation value by the reputation system.

REFERENCES

Adler, B. T., & Alfaro, L. de. (2007). A content-driven reputation system for the wikipedia. In *Proceedings of the 16th International Conference on World Wide Web* (pp. 261–270). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1242572.1242608

Adler, B. T., Alfaro, L. de, Pye, I., & Raman, V. (2008). Measuring authorcontributions to the Wikipedia. In *Proceedings of the 4th International Symposium on Wikis* (pp. 15:1–15:10). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1822258.1822279

Adler, B. T., Chatterjee, K., Alfaro, L. de, Faella, M., Pye, I., & Raman, V. (2008). Assigning trust to Wikipedia con- tent. In *Proceedings of the 4th International Symposium on Wikis* (pp. 26:1–26:12). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1822258.1822293

Bansal, S., & Baker, M. (2003). *Observation-based Cooperation Enforcement in Ad hoc Networks.* Available from http://arxiv.org/abs/cs/0307012

Bertino, E., Ferrari, E., & Perego, A. (2010, September). A General Frame- work for Web Content Filtering. *World Wide Web, 13,* 215–249. Available from http://dx.doi.org/10.1007/s11280-009-0073-5

Bertino, E., Ferrari, E., Perego, A., & Zarri, G. P. (2005). An integrated approach to rating and filtering web content. In *Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence* (pp. 749–751). London, UK: Springer-Verlag. Available from http://dx.doi.org/10.1007/11504894 104

Broersma, M. (2008, February). *OpenDNS shows off collaborative Web filter.* Available from http://www.computerworld.com/s/article/9063362/ OpenDNS shows off collaborative Web filter

Buchegger, S., & Le Boudec, J.-Y. (2002a). Nodes Bearing Grudges: To- wards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth EuromicroWorkshop on Parallel, Distributed and Network-based Processing* (p. 403-410).

Buchegger, S., & Le Boudec, J.-Y. (2002b). Performance analysis of the CONFIDANT protocol. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing* (pp. 226–236). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/513800.513828

Chesney, T. (2006, November). An empirical examination of Wikipedia's credibility. *First Monday,* 11 (11). Available from http://www.firstmonday.org/issues/issue11 11/chesney/ index.html

Cross, T. (2006, September). Puppy smoothies: Improving the reliability of open, collaborative wikis. *First Monday*, 11 (9). Available from http://firstmonday.org/issues/issue11 9/cross/index.html

Douceur, J. R. (2002). *The Sybil Attack. In Proceedings for the 1st Inter- national Workshop on Peer-to-Peer Systems (IPTPS)*, 251-260.

Ebner, M., & Zechner, J. (2006). Why is Wikipedia so successful? Experiences in establishing the principles in Higher Education. *Proceedings of I-KNOW 06, 6th International Conference on Knowledge Management, ACM Press* .

Following the crowd [Online]. (2008, September). Available from http://www.economist.com/node/11999251

Gray, C. (2010, December). *Launching a Collaboration or Content Management System: 8 Tricks for Adoption [Online]*. Available from http://www.forumone.com/blogs/post/launching-collaboration-or-content-management-system-8-tricks-adoption

Hoffman, K., Zage, D., & Nita-Rotaru, C. (2009, December). A Survey of Attack and Defense Techniques for Reputation Systems. *ACM Comput. Surv.*, 42 , 1:1–1:3. Available from http://doi.acm.org/10.1145/1592451.1592452

Howe, J. (2006, June). *Crowdsourcing: A Definition [Online]*. Available from http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_ a.html

Hu, J., & Burmester, M. (2006). LARS: a locally aware reputation system for mobile ad hoc networks. In *Proceedings of the 44th annual South- east regional conference* (pp. 119–123). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1185448.1185475

IETF MANET Working Group. Mobile Ad Hoc Networks (MANET). Working Group charter [Online]. (n.d.). Available from http://www.ietf.org/html.charters/manet-charter.html

Johnson, D. B., Maltz, D. A., & Broch, J. (2001). DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In *Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5* (pp. 139–172). Addison-Wesley.

Kennes, J., & Schiff, A. (2003, January). *The Value of a Reputation System* (Industrial Organization No. 0301011). EconWPA. Available from http://ideas.repec.org/p/wpa/wuwpio/0301011.html

Marti, S., Giuli, T. J., Lai, K., & Baker, M. (2000). Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking* (pp. 255–265). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/345910.345955

Michiardi, P., & Molva, R. (2002). CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks. In *Proceedings of the IFIP Communication and Multimedia Security Conference.*

Netcraft - Internet Research, Anti-Phishing and PCI Security Services [On- line]. (n.d.). Available from http://www.netcraft.com

Noll, M. G., & Meinel, C. (2005). Web Page Classification: An exploratory study of internet content rating systems. In *Proceedings of HACK 2005 conference.*

Noll, M. G., & Meinel, C. (2006). Design and Anatomy of a Social Web Filtering Service. In *Proceedings of 4th International Conference on Cooperative Internet Computing* (pp. 35–44).

Noll, M. G., & Meinel, C. (2008). Building a Scalable Collab-orative Web Filter with Free and Open Source Software. In *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems* (pp. 563–571). Washington, DC, USA: IEEE Computer Society. Available from http://dx.doi.org/10.1109/SITIS.2008.27

OpenDNS - Solutions - Business/Enterprise - Web Content Filtering [Online]. (2008). Available from http://www.opendns.com/solutions/business/filtering/

Pantola, A. V., Pancho-Festin, S., & Salvador, F. (2010). Rating the Raters: A Reputation System for Wiki-Like Domains. In *Proceedings of the 3rd international conference on Security of information and networks* (pp. 71–80). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1854099.1854116

Perkins, C. E., & Royer, E. M. (1999, February). Ad-hoc On Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile ComputingSystems and Applications* (p. 99-100).

A Radically Simpler Approach to Web Content Filtering and Security [White paper]. (n.d.). Available from http://www.opendns.com/support/whitepapers/?

Ravikumar, S., McAFee, R. P., & Tomkins, A. (2009, April). *Community-Based Web Filtering.* Available from http://www.freepatentsonline.com/20090112974.pdf

reCAPTCHA - Stop Spam, Read Books[Online]. (n.d.). Available from http://www.google.com/recaptcha/

Reddit.com: Help [Online]. (n.d.). Available from http://www.reddit.com/help/faq

Some eBay Users Abuse Auction Site's Feedback System, Professor Finds [Online]. (2007, January). Available from http://www.physorg.com/news87832472.html (Haas School of Business, UC Berkeley)

Steiner, D. (2003, January). Survey: How do Users Feel About eBay's Feedback System? *Auction Bytes. The Independent Trade Publication for Online Merchants* . Available from http://www.auctionbytes.com/cab/abu/y203/m01/abu0087/s02

Trivedi, A. K., Arora, R., Kapoor, R., Sanyal, S., Abraham, A., & Sanyal, S. (2009). Mobile Ad Hoc Network Security Vulnerabilities. *IGI Global.*

Untangle - Multi-functional firewall Software - Open Source Content Filter and Spam Filter [Online]. (n.d.). Available from http://www.untangle.com

Untangle - Web Filter submission tool [Online]. (n.d.). Available from http://forums.untangle.com/web-filter/2143-web-filter- submission-tool.html

Untangle - WebFilter Technical Specifications [Online]. (n.d.). Available from http://www.untangle.com/Web-Filter/Tech-Specs

Wikipedia, the free Encyclopedia [Online]. (2001). Available from http://www.wikipedia.org

Yao, E., Fang, R., Dineen, B. R., & Yao, X. (2009, December). Effects of customer feedback level and (in)consistency on new product acceptance in the click-and-mortar context. *Journal of Business Research* , 62 (12), 1281-1288.

Zittrain, J. (2008). *The Future of the Internet–And How to Stop It* (No. 36). Yale University Press. Available from http://www.amazon.com/Future-Internet-How-Stop/dp/0300151241

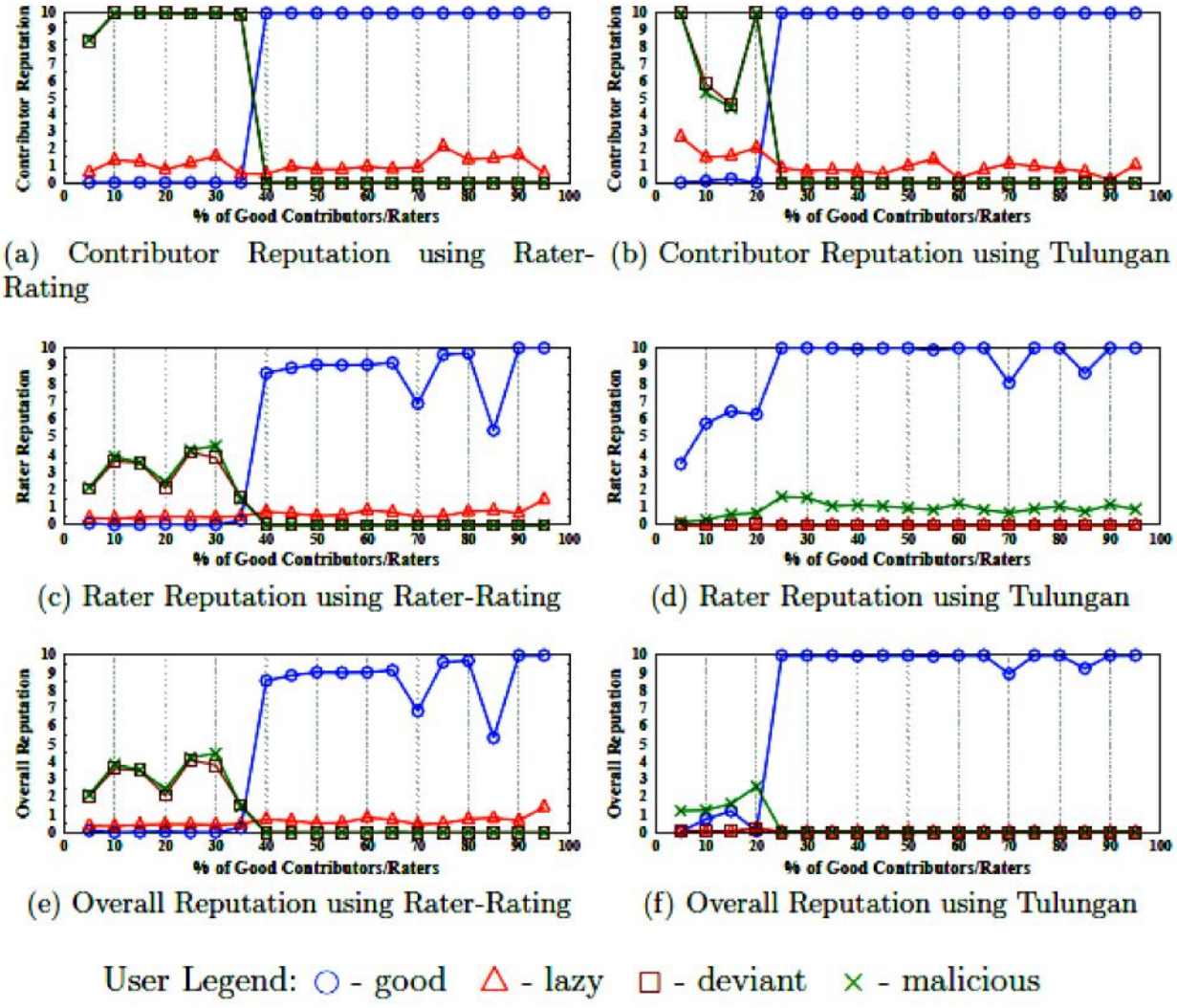

Figure 1: Good versus Bad Users: Rater-Rating Reputation System Results (left figures: a, c, e) and Tulungan Reputation System Results (right figures: b, d, f)

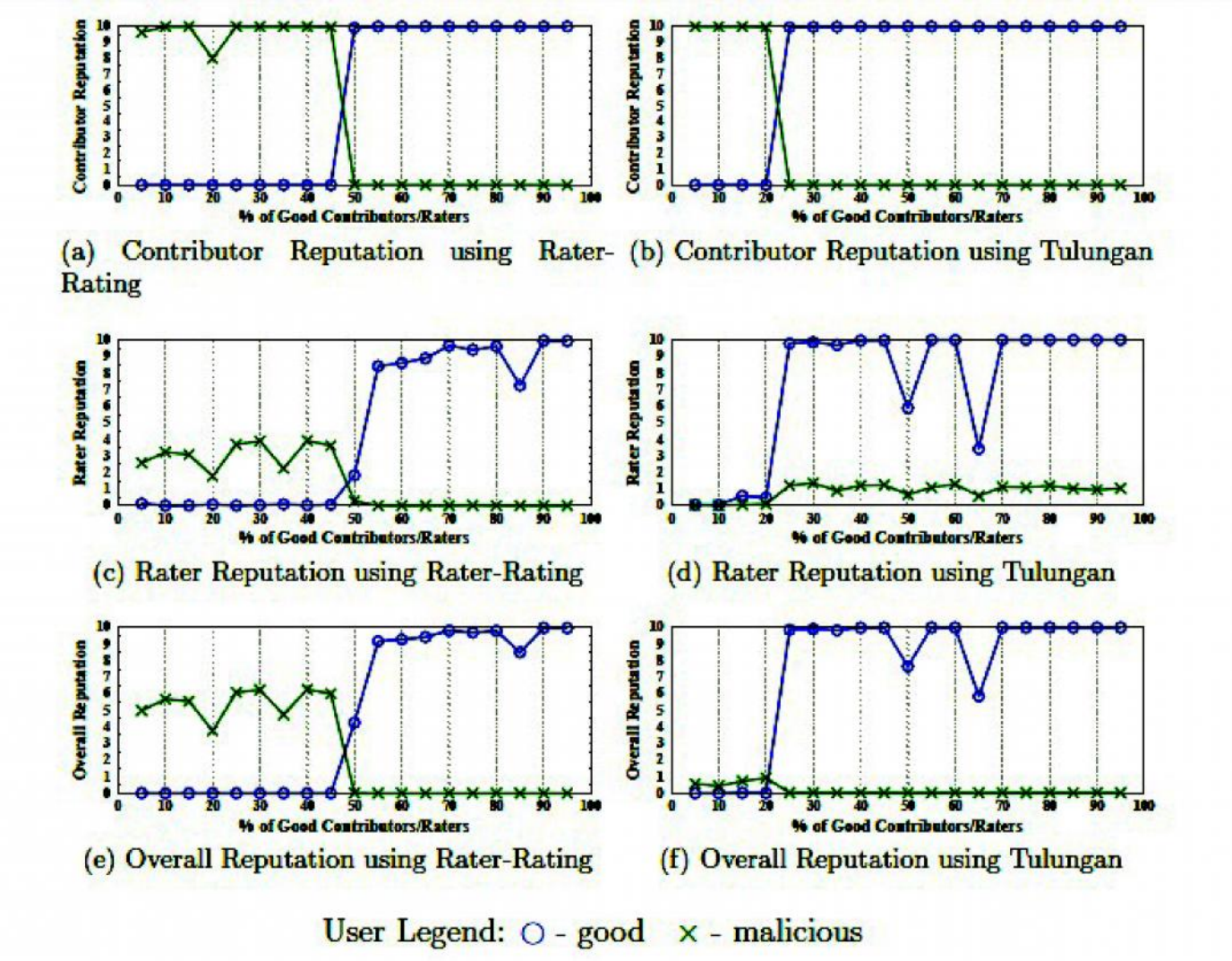

Figure 2: Good versus Malicious Users: Rater-Rating Reputation System Results (left figures: a, c, e) and Tulungan Reputation System Results (right figures: b, d, f)