# SOAP: A Social Network Aided Personalized and Effective Spam Filter to Clean Your E-mail Box

Ze Li and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University
Clemson, SC 29631
Email: {zel, shenh}@clemson.edu

***Abstract*—The explosive growth of unsolicited emails has prompted the development of numerous spam filtering techniques. A Bayesian spam filter is superior to a static keyword-based spam filter because it can continuously evolve to tackle new spam by learning keywords in new spam emails. However, Bayesian spam filters can be easily poisoned by avoiding spam keywords and adding many innocuous keywords in the emails. In addition, they need a significant amount of time to adapt to a new spam based on user feedback. Moreover, few current spam filters exploit social networks to assist spam detection. In order to develop an accurate and user-friendly spam filter, in this paper, we propose a SOcial network Aided Personalized and effective spam filter (SOAP). Unlike previous filters that focus on parsing keywords (e.g, Bayesian filter) or building blacklists, SOAP exploits the social relationship among email correspondents to detect the spam adaptively and automatically. SOAP integrates three components into the basic Bayesian filter: social closeness-based spam filtering, social interest-based spam filtering, and adaptive trust management. We evaluate performance of SOAP based on the trace data from Facebook. Experimental results show that SOAP can greatly improve the performance of Bayesian spam filters in terms of the accuracy, attack-resilience and efficiency of spam detection. We also find that the performance of Bayesian spam filters is the lower bound of SOAP.**

## I. Introduction

Internet email is one of the most popular communication methods in the realm of our business and daily lives. Internet email has been reliable throughout most of its long history. However, spam is becoming the penultimate problem in email systems, accelerating at an alarming speed. Unsolicited commercial emails have increased from approximately 10% of overall email volume in 1998 to as much as 80% today [1]. Currently, 120 billion spam emails are sent per day [2], with a projected cost of $338 billion by 2013 [3]. Spam emails interfere with both email service providers and end users. It forces users to search legitimate (i.e., non-spam) emails in an inbox filled with spam. Spam emails also take up a huge amount of bandwidth that otherwise could be used to transmit more useful information.

The fundamental way to prevent spam is to make it unprofitable to send spam emails and thereby destroying the spammers' underlying business model [4]. Oscar *et. al.* indicated that there is a strong relationship between the average cost of sending spam, spam detection accuracy and spam filter deployment [4]. Specifically, any measure that stops spam from reaching users' inboxes with probability $p$ and is deployed by users with probability $q$ increases the average cost of sending spam by $1/(1-pq)$. This means that in order to increase the cost, a spam filter should increase detection accuracy $p$ and user deployment $q$. Such a spam filter should be *attack-resilient*, *personalized* and *user-friendly*.

The *attack-resilient* and *personalized* features are important to achieve higher accuracy. A higher accurate filter generates less *false positives* and *false negatives*. *False positives* are the legitimate emails that are mistakenly regarded as spam emails. *False negatives* are the spam emails that are not detected. There are primarily two types of spam filter attacks: *poison attack* and *impersonation attack*. In a poison attack, many legitimate keywords are added to a spam email, thus decreasing its probability of being detected as spam. In an impersonation attack, a spammer impersonates the identities of ordinary users by forging their IDs or compromising their computers. An estimated 50% - 80% of all spam worldwide was sent by compromised computers, also known as zombies [5], in the year of 2005. An attack-resilient spam filter can detect spam emails in these attacks.

By personalized, we mean that an accurate spam filter should consider the social situation of a particular individual in detecting spam. First, it considers social closeness between correspondents. Closer social relationship between two persons implies high trust between them [6]. People treat emails sent by strangers and acquaintances differently. Emails containing keywords such as "lose weight" are usually regarded as spam. However, such keywords may be in the emails sent between members of a health club. Commercial emails are usually considered as spam. However, the email sender may be the receiver's friend and believes the advertisement-like information benefits the receiver. Second, a spam filter considers different (dis)interests of individuals. For example, an email about "football" is not spam to football fans, but is spam to those who are not interested in football. Thus, determining a legitimate mail is different from person to person. A user-friendly (i.e., easy-to-use) spam filter does not require a large amount of manual spam detection activities from users.

However, few previous spam filtering approaches can meet the requirements of being *user-friendly*, *attack-resilient* and *personalized*. We list the main approaches in Table I with their features. Most approaches do not take into account the different closeness relationship and (dis)interests of individu-

als. They usually consider unsolicited commercial emails and investment request emails as spam. This traditional view of spam is not sufficiently accurate.

Previous spam filter approaches can be mainly divided into two categories: content-based and identity-based. In the content-based category, emails are parsed and scored based on the keywords and patterns which are typical in spam. The simplest spam filter is static keyword-based filter [7], in which a spam keyword blacklist is built. However, it cannot adapt to the continuously changing characteristics of spam. Machine learning approaches [8]–[15] can adaptively learn new spam keywords. In these approaches, the spam filters are trained with a corpus of both spam and legitimate emails. Characteristics of the spam and legitimate emails are identified respectively. The characteristics are used to automatically categorize future emails into the two classes. However, machine learning approaches still suffer from a number of problems. First, in order to increase the efficiency and accuracy of training, the spam filters are normally installed in an email server to collect all the training samples. Thus, they are not personalized. Second, the spam filters are vulnerable to poison attacks. Third, the spam filters are not user-friendly. They require a lot of user efforts to manually distinguish the spam from legitimate emails for training.

TABLE I
FEATURES OF SPAM FILTERS

| Approaches | Person-alized | Attack-resilient | | User-friendly |
|---|---|---|---|---|
| | | Impersonation | Poison | |
| Content-based spam filters | | | | |
| Static keyword [7] | No | Yes | No | No |
| Machine learning [8]–[17] | No | Yes | No | No |
| Collaborative [18]–[20] | No | Yes | No | Yes |
| Identity-based spam filters | | | | |
| Black/white list [21]–[24] | No | No | Yes | No |
| Social interaction-based [25]–[28] | No | No | Yes | Yes |
| Reputation [29] | No | No | Yes | No |
| Social network aided content and identity based spam filter | | | | |
| SOAP | Yes | Yes | Yes | Yes |

Identity-based spam filters focus on the identities of email senders. In the simplest method, a user manually selects addresses of the emails that (s)he would like to accept or reject and put them into whitelist and blacklist, respectively [21]–[24]. Social interaction-based spam filters [25]–[28] exploits friends-of-friends (FoF) relationships among email correspondents to create whitelist and blacklist. Since these spam filters disregard email content, they are resilient to poison attacks. Additionally, because these filters automatically identify spammers from normal users according to their communication pattern, they are user-friendly. However, they are not personalized. In addition, they are vulnerable to impersonation attacks. If a user's email account is compromised or the user's ID is forged by a spammer, the user's friends can be easily attacked by the spammer because of the highly clustered nature of people's interaction network [30]. Moreover, as indicated in [31], the assumption that person $A$'s friend of friend is also $A$'s friend is not generally true. For example, $A$ and $B$ are friends in a math class and $B$ and $C$ are friends in the gym. Then, $A$ and $C$ may not be friends if $A$ does not go to the gym.

On-line social networks have gained significant popularity recently. The users in a social network are linked based on their social relationships. In this paper, we propose a SOcial network Aided Personalized and effective spam filter (SOAP) for spam detection to meet the three requirements. In the SOAP applications, users are encouraged to provide their social information such as (dis)interests, religions, occupation and affiliation in order to avoid spam. Different from social interaction-based spam filters [25]–[28] that focus on the interaction via emails, SOAP explores personal information in the social network and infers the relationship closeness and (dis)interests of individuals for accurate spam detection.

SOAP leverages social networks and combines three components into a basic Bayesian filter:

(1) Social closeness-based spam filtering. SOAP calculates the closeness between nodes based on social relationship. Since nodes with higher closeness have less probability to send spam emails to each other, emails from nodes with lower closeness are checked more strictly and vice versa. This component helps SOAP to be resilient to poison attacks.

(2) Social interest-based spam filtering. SOAP infers nodes' (dis)interests based on social profiles. The inferred information helps the filter to enhance the accuracy of spam detection with the consideration of individual preference. This component contributes to the personalized feature of SOAP.

(3) Adaptive trust management. In order to tackle impersonation attacks, SOAP relies on the additive-increase/multiplicative-decrease algorithm (AIMD) [32] to adjust the trust values of nodes which are used to tune closeness values to block emails from low-trust nodes or normal nodes impersonated by spammers.

SOAP can rapidly determine spam and adapt to new spam keywords. It achieves a high spam detection accuracy due to its personalized and attack-resilient features. In addition, it is user-friendly as it does not need much user effort to identify spam and legitimate emails. Meanwhile, its highly accurate and automatic spam detection reduces the training time of the Bayesian filter. Further, rather than using a centralized server to collect user social information as in RE [26], SOAP can collect the information in a decentralized manner, reducing the burden caused by information querying on the centralized server. Authentication and encryption techniques can be used in SOAP to increase its communication security, and these techniques are orthogonal to the study in this paper.

The remaining part of the paper is organized as follows. Section II gives a brief overview of the existing spam filter approaches. Section III describes the design of SOAP. Section IV evaluates the performance of the SOAP. Finally, Section V concludes the paper.

## II. RELATED WORK

The vast quantities of spam that are distributed blindly in bulk has stimulated many spam filtering approaches. These

approaches can be mainly separated into two classes: content-based and identity-based.

**Content-based Approaches:** The basic approach of content-based spam filter is the static keyword list [33]. If the keywords in the email match the keywords in the spam keyword list, then the email is regarded as spam. However, its static feature makes it easy for a spammer to evade filtering by tweaking the message.

The second category of content-based approaches are machine learning-based approaches including Bayesian filter [16], decision trees [8], [9], Support Vector Machine (SVM) [10], [11], Bayes Classifier [12]–[14] and a combination of these techniques [15]. In this approach, the filter is trained to distinguish the keywords in spam and legitimate emails. A learning algorithm is used to find the characteristics of the spam and legitimate emails. Then, future messages can be automatically categorized as highly likely to be spam, to be legitimate, or somewhere in between. Bayesian filter [16] is one of the widely used machine learning techniques. It parses an email into keywords, assigns a weight to each keyword indicating the probability that the email containing the keyword is spam, and infers the email's posterior probability of spam based on the keyword weights.

The third category of content-based approaches is collaborative spam filtering. Once a spam email is detected by one user, other users in the community can prevent the spam email later on by querying others whether the received email is spam or not. SpamNet [20] uses a central server to connect all participants of the collaborative spam filter. SpamWatch [19] is a distributed spam filter based on the Tapestry Distributed Hash Table (DHT) system. Kong *et al.* [18] proposed a distributed spam filter to increase the scalability of centralized collaborative spam filter. The spam filter leverages email networks for collaborative spam detection. Users query all of their email clients to see if another user has labeled a suspect message as spam.

**Identity-based Approaches:** The simplest identity-based spam filtering approaches are blacklist and whitelist approaches [21]–[24], which check the email senders for spam detection. Whitelist and blacklist maintain a list of addresses of people whose emails should not and should be blocked by the spam filter, respectively. Some server-side solution [24] records the number and frequency of the same email sent to multiples destinations from specific IP addresses. If the number and frequency exceed thresholds, the node with the corresponding IP address is blocked.

Boykin *et al.* [25], [28] constructed a graph in which vertices represent email addresses and direct edges represent email interactions. Emails are identified as spam, valid or unknown based on the local clustering coefficient of the graph subcomponent. This is based on the principle that the social communication network of a normal node has a higher clustering coefficient than a spam node. RE [26] is a whitelist spam filtering system based on social links. It is based on the assumption that all friends and FoF are trustable. Hameed [27] proposed LENS that extend FoF network by adding trusted users from outside of their FoF networks to mitigate spam beyond social circles. Only the emails to a particular recipient have been vouched by the trusted nodes can be sent into the network. However, if a legitimate user's computer is compromised by a spammer, many people will be easily targeted by the spammer through social networks characterized by high clustering and short paths [30]. Also, such social interaction-based method is not sufficiently effective in dealing with legitimate emails from a sender out of the social network of the receiver. Golbeck *et al.* [29] proposed an email scoring mechanism based on an email-network augmented with reputation ratings. An email is considered as spam if the reputation score of the email sender is very low.

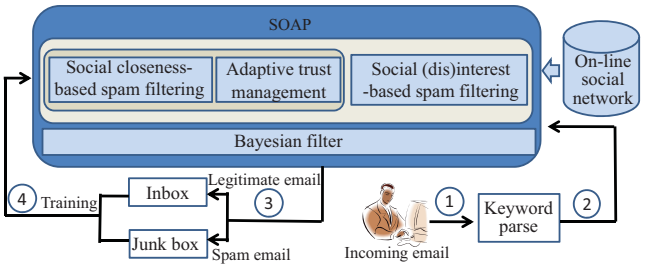


Fig. 1. The structure of the SOAP.

## III. SOAP: Social Network based Bayesian Spam Filter

### A. System Overview

SOAP is a personalized, attack-resilient and user-friendly social network based Bayesian spam filter. Unlike current social network based filters that focus on email interaction networks [18], [26], [27], [29], SOAP also leverages social information including (dis)interests and people relationships. SOAP encourages users to indicate their (dis)interests and their social relationship with their email correspondents in order to receive less spam and lose less legitimate emails. Using this information, SOAP can build a social network connecting the users. SOAP can also be used as a plugin in current on-line social online networks, such as Facebook, MySpace, to filter out the spam-like comments posted on walls or pictures.

Figure 1 shows the structure of SOAP. SOAP integrates three new components to the Bayesian filter: (1) social closeness-based spam filtering, (2) social interest-based spam filtering, and (3) adaptive trust management. Based on the collected social information, SOAP infers node closeness and email preference for individuals. The Bayesian filter keeps a list of spam keywords and corresponding weights showing the probability that the email containing the keyword is spam. For simplicity, we call the weight as the keyword's weight. Based on the three social-based components, after parsing the keywords of an email, SOAP adjusts the weights of the keywords. Then, SOAP resorts to the Bayesian filter for spam evaluation. The weights are adjusted based on the closeness between the receiver and the sender, the receiver's (dis)interests, and the receiver's trust of the sender. If the closeness is high, the likelihood that the emails sent between them are spam is low, and then the weight is decreased

and vice versa. However, it is possible that close nodes will be compromised. This problem is handled by the trust management component. For those nodes with low closeness, the emails are evaluated based on the user's (dis)interests.

As mentioned, content-based spam filters focus on email content and can prevent impersonation attacks. Identity-based spam filters focus on the communication relationship between correspondents and hence are resilient to the poison attacks. SOAP combines the advantages of both types of spam filters. The accurate results from SOAP become training data to automatically train the Bayesian filer, thus making the filter user-friendly and personalized and reducing the training time. In the section below, the following issues will be addressed.

- How does the basic Baysian filter work? (Section III-B)
- How to calculate the closeness of individuals in a distributed manner and how to integrate the closeness consideration into the Baysian filter? (Section III-C)
- How to infer the (dis)interests of individuals and integrate the email preference consideration into the Baysian filter? (Section III-D)
- How to adjust trust value to avoid impersonation attacks? (Section III-E)
- How do the different components in SOAP cooperate for spam detection? (Section III-F)

### B. Overview of the Basic Bayesian Spam Filter

Bayesian filter has a list of keywords along with their probabilities to identify an email as a spam email or a legitimate email. The list is built by training the filter. During training, given a pool of emails, a user manually indicates whether each email is spam or not. We use $P(S)$ and $P(L)$ to denote the probability that an email is a spam email and a legitimate email, respectively. The filter parses each email for spam keywords. It calculates the probabilities that a word $w$ appears in a spam email and a legitimate email, denoted by $P(w|S)$ and $P(w|L)$ respectively. After training, the calculated probabilities are used to compute the probability that an email with a particular set of keywords in it belongs to either category. Then, the probability that an email including a set of keywords $W$ is spam is:

$$P(S|W) = \frac{P(S,W)}{P(W)} = \frac{\prod_i P(w_i|S)P(S)}{\prod_i P(w_i|S)P(S) + \prod_i P(w_i|L)P(L)}. \quad (1)$$

The Bayesian filter sets a threshold, denoted by $T$. If an email's parsed keywords is $W_1$ and $P(S|W_1) \geq T$, then the email is spam. Otherwise, it is legitimate.

### C. Social Closeness-based Spam Filtering

The probability that two persons with high closeness send spam to each other is low unless their machines are under impersonation attacks. Thus, the closeness between individuals can be utilized to improve the accuracy of spam detection. SOAP loosely checks emails between individuals with high closeness and strictly checks emails between individuals with low closeness. In this section, we propose an algorithm to calculate social closeness values.

SOAP relies on node social relationship, such as kinship, friendship and colleague, to determine node closeness values. SOAP sets different weights for different social relationships for node closeness measurement. For example, the closeness of kinship relationship usually weights more than business relationship. We use $w(u,v)$ to denote the weight of a relationship between node $u$ and $v$.

*1) Closeness of Adjacent Nodes:* In a social network, more relationships between two adjacent nodes make them closer. Thus,

$$C(u,v) = \sum_{i=1}^{n} w_i(u,v) \quad (2)$$

where $n$ is the number of relationships between $u$ and $v$, $w_i(u,v)$ is the relationship weight of the $i^{th}$ relationship.

*2) Closeness of Non-adjacent Nodes:* Based on the closeness value between any two adjacent nodes, the closeness of non-adjacent nodes can be calculated with the aid of relationship transitivity, in which relationship closeness can be passed along the nodes. The closeness transitivity should capture three properties in order to correctly reflect the social relationship.

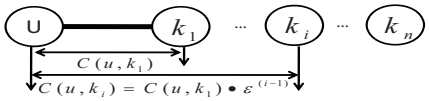


Fig. 2. Closeness propagation property.

*Property 3.1: Closeness propagation property.* The closeness between node $A$ and other nodes exponentially decreases with their distances. As shown in Figure 2, it can be illustrated by $C(u,k_i) = C(u,k_1) \cdot \varepsilon^{i-1} (\varepsilon < 1)$.

As shown in Figure 2, the more hops between node $u$ and node $k_i$, the less closeness between them. The closeness value is decreased to an extremely small value when the distance exceeds 3 hops. This relationship has been confirmed by other studies. Binzel *et al.* [6] discovered that a reduction in social distance between two persons significantly increases the trust between them. Swamynathan *et al.* [34] found that people normally do e-commerce business with people within 2-3 hops in their social networks.

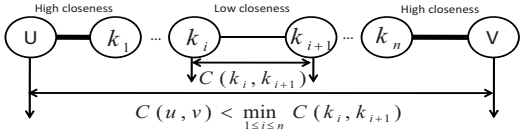


Fig. 3. Weakest link property.

*Property 3.2: Weakest link property*. The weakest link in a social path (not necessary disjoint path) is the direct link between adjacent nodes that has the minimum closeness, denoted by $min_{1 \leq i \leq n} C(k_i, k_{i+1})$. The closeness between two non-adjacent nodes $u$ and $v$ is upper bounded by the closeness of the weakest link between $u$ and $v$. That is, for a social network path from node $u$ to node $v$ with $n$ nodes in the middle, $C(u,v) < min_{1 \leq i \leq n} C(k_i, k_{i+1})$, where node $k_i$ is in the path between $u$ and $v$.

According to Property 3.1, the closeness value decreases as the social distance between two nodes increases. In Figure 3,

if person $u$ knows $v$ though person $k_1$, then the closeness between $u$ and $v$ must be less than that between $k_1$ and $v$, i.e., $C(u,v) < C(v,k_i)$.

*Property 3.3: Closeness accumulation property*. The more social paths that exist between node $u$ and node $v$, the higher closeness they have. Specifically, if node $u$ and node $v$ have $p$ social pathes between them, their closeness through $p$ paths denoted by $C(u,v,p)$ is

$$C(u,v,p) = \sum_{m=1}^{p} (C_m(u,v)). \tag{3}$$

The underlying idea is that if person $u$ has more ways to get in touch with person $v$, $u$ has higher closeness with $v$.

We design a closeness calculation formula that merges the above three properties:

$$C(u,k_{i+1},p) = \sum_{m=1}^{p} \Big( C_m(u,k_i) \cdot (C_m(k_i,k_{i+1})/\varphi)^i \Big) \tag{4}$$

where $\varphi$ is a scale parameter to control the closeness scale rate in each hop in closeness propagation, and

$$\varphi > max_{1<x<i}\big(C(k_{x-1},k_x) \cup C(u,k_1)\big).$$

We can see that this formula meets Properties 3.1, 3.2 and 3.3.

*3) Distributed Closeness Calculation Algorithm:* In social networks, each person has a friend list. Based on the personal information of his/her friends, the closeness values with adjacent friends can be calculated. Most current social network have a cental server to store all information of individuals in the social network. However, such a centralized method may generate a single point of failure, and hence is not scalable. We propose a distributed algorithm for the closeness calculation. Specifically, a source node broadcasts a query message with a specified TTL along the FOF links. Upon receiving the message, an intermediate node decreases TTL by 1 and inserts its closeness values with its neighbors into the message and then forwards it to its neighbors. The process continues until TTL becomes 0. Then, the destination node directly sends the message back to the source node. As a result, the source node retrieves all closeness values of the nodes in the path to the destination. It calculates its closeness with each of the node depending on Equation (4).

It was shown that the average hops between any two persons in the world are within 6 hops [35]. Hence, we set TTL=6/2=3 for two reasons. (1) Broadcasting along more hops produces high overhead, and (2) Property 3.1 indicates the closeness decreases exponentially. Therefore, the source has very low closeness to the nodes far away from itself, and the emails from these nodes should be strictly checked.

*4) Integration with Bayesian Filter:* In the Bayesian filter, email keywords are weighted to show the likelihood that an email is spam. SOAP adjusts the keyword weights based on the closeness between the email receiver and the sender in determining spam. Specifically, high closeness reduces weights and low closeness increases weights. Thus, emails from people with high closeness are regarded as legitimate emails with high probability, while emails from strangers or people with low closeness are further checked strictly. The keyword tuning function is:

$$W(key) := \begin{cases} W(key)e^{-k_f \cdot (C(u,v)-k_t)} & \text{if } C(u,v) \geq k_t; \\ W(key)\xi \ (\xi \geq 1) & \text{if } C(u,v) < k_t. \end{cases} \tag{5}$$

where $W(key)$ is the weight of a keyword, $k_f$ is a scale parameter to adjust the decreasing rate of $W(key)$, and $k_t$ is a location parameter to determine where the origin of exponential decreasing is located [36]. If $C(u,v) = k_t$, then the weight is not changed. If $C(u,v) > k_t$, then $W(key)$ is decreased by a factor of $e^{k_f \cdot (C(u,v)-k_t)}$. If $C(u,v) < k_t$, then $W(key)$ is increased by a factor of $\xi$ ($\xi \geq 1$). $\xi$ can be adjusted by users with different accuracy requirement. Higher $\xi$ makes an email to have a higher probability to be regarded as spam. $\xi$ normally is set to be 1 in order to reduce false positives.

### *D. Social Interest-based Spam Filtering*

The social interest-based spam filtering component aims to make SOAP personalized in order to increase the spam detection accuracy. It is actually a content-based spam detection method. By matching the keywords in an email and the email receiver's social (dis)interests, SOAP increases and decreases the probability of these keywords to be spam respectively. Then the Bayesian filter calculates the probability that the email is spam based on Formula (1).

SOAP relies on a rule-based inference system [37] to infer user preference. The inference system has three components. The *profile* component is a database containing all useful information parsed from users' profiles in the social network. Such information includes interests, occupation, affiliates and so on. The *inference rules* component contains all the rules that are used for the inference of (dis)interests. Such rules can be rational reasoning based on non-monotonic logic or people's common sense. The *inference engine* component determines the applicability of the rules in the context of the current profile, and selects the most appropriate rules for the inference.

*1) Integration with Bayesian Filter:* Bayesian filter detects spam by directly matching the keywords in an email with the keywords in the spam filter. SOAP integrates a receiver's (dis)interests social information into the matching process. If an email keyword is within the receiver's interests, SOAP decreases the spam weight of the keyword in order to increase the probability that the email is legitimate. On the other hand, if an email keyword is within the receiver's disinterests, SOAP increases the spam weight of the keyword in order to increase the probability that the email is spam. Then, SOAP relies on the basic Bayesian filter (Section III-B) for spam detection. SOAP is a personalized spam filter since it considers individual (dis)interests in spam detection.

For a spam keyword within an individual's interests, its weight is tuned by:

$$W(key_{interest}) = W(key_{interest}) \cdot e^{-\rho_I} \tag{6}$$

where $key_{interest}$ is the spam keyword in interests and $\rho_I$ is a scale parameter. As $\rho_I$ increases, $W(key_{interest})$ decreases. Therefore, the probability that the email is a spam decreases.

As a result, emails interested by a receiver usually will not be regarded as spam. Therefore, SOAP can reduce false positives in traditional spam filters that lack the personalized feature.

If a spam keyword matches the disinterests of an email receiver, the weight of the keyword is adjusted by

$$W(key_{disinterest}) = W(key_{disinterest}) \cdot e^{\rho_D} \quad (7)$$

where $key_{disinterest}$ is the spam keyword in disinterests. As $\rho_D$ increases, the weight of the spam keyword is increased. Thus, even if a spammer has added many legitimate words into an email in order to disguise spam keywords, as long as the email has keywords in disinterests, it has high probability to be regarded as spam. The more disinterest keywords an email has, the higher probability the email will be rejected. Meanwhile, since the (dis)interests of different persons are different, it is very difficult for a spammer to modify the keywords in a spam email to match the interests and avoid disinterests of a person. In this way, SOAP resists spam poison attacks.

### *E. Adaptive Trust Management*

Recall that in impersonation attacks, a spammer impersonates the identities of benign computers by forging their IDs or compromising them to send spam. Due to the power-law and small-world characters of social networks, impersonation can spread spam extremely fast. In order to avoid impersonation attacks, SOAP integrates an adaptive trust management component. Specifically, a node tracks rapid behavior changes of close-relationship nodes. It uses the additive-increase/multiplicative-decrease algorithm (AIMD) [32] to adjust node trust. Node trust is used to update node closeness for the detection of false negatives due to compromised attacks.

AIMD is a feedback control algorithm used in TCP Congestion Avoidance. It combines linear growth of the transmission rate with an exponential reduction when congestion takes place. AIMD aims for a balance between responsiveness to congestion and utilization of available capacity. Similarly, SOAP aims for a balance between responsiveness to false negatives and acceptance of trustable emails. In SOAP, node $A$ initially assumes node $B$ with high closeness is trustworthy until it receives a spam email (i.e., false negative) from $B$.

We use $t_{(i,j)}$ to denote the trust value of node $j$ regarded by node $i$. The maximum trust value $t_{max} = 1$ and $t \leq t_{max}$. $t$ is initially set to $t_{max}$. When node $i$ receives a spam email from sender $j$, node $j$ changes the trust value of $i$ by

$$t_{(i,j)} := a \cdot t_{(i,j)} \ (0 < a < 1). \quad (8)$$

When a node receives a legitimate mail from a sender, then

$$t_{(i,j)} := t_{(i,j)} + b \ (0 < b < 1). \quad (9)$$

In this way, SOAP can sensitively adjust node trust value to quickly react to zombies, thus reducing false negatives.

It is important to determine appropriate values for $a$ and $b$. Smaller $a$ (i.e., faster trust decrease) and $b$ (i.e., slower trust increase) lead to less false negatives, but more false positives. On the other hand, larger $a$ and $b$ result in less false positives, but more false negatives. In order to maximally reduce false negatives without concurrently generating more false positives, SOAP complements AIMD with a new strategy. That is, when a user notices a legitimate email in the junk box (i.e., false positive), it increases the node trust by

$$t_{(i,j)} := t_{(i,j)} + \alpha \cdot b \ (\alpha > 1). \quad (10)$$

In order to reach the optimal point between the false negatives and false positives, parameters $a$ and $b$ are the functions of the number of false negatives and false positives respectively, denoted by $n_{fn}$ and $n_{fp}$. Specifically, $a = \mathrm{F}(n_{fn})$ is a decreasing function and $b = \mathrm{F}(n_{fp})$ is an increasing function. In other words, if $n_{fn1} < n_{fn2}$, then $a_1 > a_2$. If $n_{fp1} < n_{fp2}$, then $b_1 < b_2$. Briefly, larger $n_{fn}$ leads to smaller $a$, and larger $n_{fp}$ leads to larger $b$. The functions are designed in this way so that when there are a large number of false negatives, $a$ decreases in order to quickly reduce node trust value to reduce false negatives. On the other hand, when there are a large number of false positives, $b$ increases in order to quickly increase the trust value to reduce false positives.

After the trust value is updated, the closeness value between node $i$ and node $j$ is updated by

$$C(i,j) := t_{(i,j)} \cdot C(i,j). \quad (11)$$

### *F. The Integration of Components in SOAP*

Figure 1 shows how the different components in SOAP cooperate with each other for accurate spam examination of an incoming email. Using the social closeness-based spam filtering algorithm, a node calculates the closeness of other nodes with itself and keeps a list for the closeness values. In phase $(1)$, when a node receives an email, SOAP parses out the keywords in the email.

For each keyword, the basic Bayesian filter calculates the probability that it is spam keyword. In phase $(2)$, the different components adjust the probability in order to increase the accuracy of spam detection. First, based on the closeness between the email sender and receiver, the probability of each keyword is adjusted (Equation (5)). Second, if the closeness is above a pre-determined threshold, this email is legitimate. Otherwise, the social interest-based spam filtering algorithm is used. If the keywords match the interests of the receiver, it means the email is useful to the receiver. Subsequently, the probabilities of these keywords are decreased based on Equation (6). On the other hand, if the keywords are within the disinterested list of the receiver, it means the email is useless to the receiver. Then, the probabilities of these keywords are increased based on Equation (7). Finally, according to the probabilities of the keywords, the Bayesian filter determines whether the email is spam. In phase $(3)$, the email is forwarded to the *Inbox* and the *Junk box* correspondingly. The results are used for spam detection training in phase $(4)$. The Bayesian training enables SOAP to automatically determine whether an email is spam later on. A user usually recovers legitimate emails in the junk box by using the "not spam" function, and deletes spam emails directly without reading them. Based on these false negatives and false positives, the

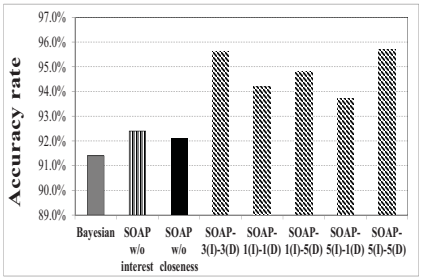


Fig. 4. Accuracy.

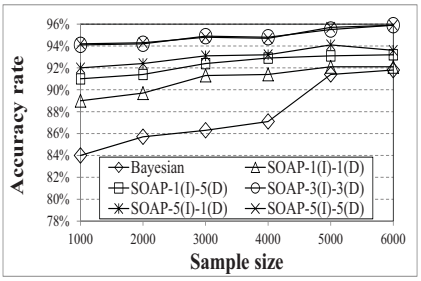


Fig. 5. Accuracy vs. # of emails.

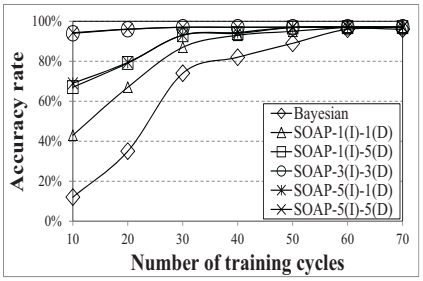


Fig. 6. Training time.

adaptive trust management in SOAP adjusts node trust and tune node closeness accordingly (Equation (11)). The goal of trust management is to counter impersonation attacks, i.e., reducing false negatives while restricting false positives.

## IV. Performance Evaluation

### A. Experiment Methodology

We evaluated the performance of SOAP compared to the Bayesian filter [16] (Bayesian in short) and the RE [26] interaction-based spam filter through simulations.

TABLE II
List of SOAP parameters

| Parameter | Value |
|---|---|
| Closeness value | Kinship=In relationship=2; Colleague = Classmate = 1.5; Familiar=1; |
| Closeness scale rate | $\varphi$= 1.3 |
| Scale parameter of closeness | $k_f = 1$ |
| Location parameter of closeness | $k_t$=1 |
| Scale parameter | $\rho_I = 3$, $\rho_D = 3$ |
| AIMD parameters | a=0.5, b=0.1 |

We have built a social network based on data crawled from Facebook. We selected two users with no social relationship in Clemson University as seed nodes and built a friend graph using the breadth first search through each node's friend list. We skipped the users whose personal information cannot be accessed. Finally, a connected social network with 32344 users was established for SOAP. The average number of friends per node is $32.51$ and the average path length of the graph is $3.78$. The personal information such as religion and interests of each node was parsed and stored in the node. We merged sub-category interests into a higher level category. For example, *Mozart* is classified into classic music, and the book *Gone with the Wind* is classified into literature. The average number of interests per person after merging is 6.64. The links between persons are weighted based on their relationship indicated in their profiles in Facebook.

We collected $9500$ emails including 2000 spam and 7500 legitimate mails from the spam-assassin project [38], and 500 commercial emails from the email boxes of five members in our lab. Among the emails, we randomly chose 6000 emails as training sample for Bayesian and SOAP. The parameters used in the simulation are shown in Table II. We continuously chose random $4$ pairs of nodes at a time and let them send an email to each other until $40000$ pairs were chosen. The email sent was randomly selected from the repository of the collected emails. We assume persons with closeness $> 2$ do not send spam to each other except in the case of impersonation attacks.

### B. Detection Accuracy

We define the metric of accuracy rate as the ratio between the number of successfully classified emails and the number of all received emails. Also, we use SOAP-x(I)-y(D) to denote SOAP with interest scale parameter $\rho_I$=$x$ and $\rho_D$=$y$. Figure 4 shows the average accuracy rate of Bayesian, SOAP without the interest-based spam filtering component, SOAP without the social closeness-based spam filtering component, and SOAP with all components with different $\rho_I$ and $\rho_D$. The experimental results show that Bayesian produces lower accuracy than all other methods. Without the interest-based component or the closeness-based component, SOAP still achieves higher accuracy than Bayesian. SOAP with all components achieves the highest accuracy. The improved performance of SOAP without closeness-based component is because the interest-based component can identify the emails that match the receiver's disinterests but are generally regarded as legitimate, and the emails that match the receiver's interests but are generally regarded as spam. The improved performance of SOAP without interest-based component is because the closeness-based component can identify the legitimate mails from close nodes. Legitimate emails containing general spam keywords are regarded as spam by Bayesian, but are correctly classified by SOAP considering node social closeness.

We can also see from the figure that SOAP-3(I)-3(D) and SOAP-5(I)-5(D) produce the highest accuracy rates. When $\rho_I$=$\rho_D$, SOAP does not bias either interest or disinterest keywords. Recall that a higher keyword weight means higher probability that an email is considered as spam. Giving the same weight to interest and disinterest keywords leads to a fair judgement on email keywords, thus generating high detection accuracy. Since the weights generated by $\rho_I$=$\rho_D$= 3 and $\rho_I$=$\rho_D$= 5 based on Equations (6) and (7) are very close, SOAP-3(I)-3(D) and SOAP-5(I)-5(D) have close accuracy rates. Although $\rho_I$ and $\rho_D$ in SOAP-1(I)-1(D) are also the same, as the keyword weight it generates is extremely small, the disinterest/interest filtering feature of SOAP is not demonstrated. Therefore, its accuracy is much less than SOAP-3(I)-3(D) and SOAP-5(I)-5(D). SOAP-1(I)-5(D) and SOAP-5(I)-1(D) produce lower detection accuracy as they give higher weight to either disinterest keywords or interest keywords. Since SOAP strictly checks emails with keywords of high weight, their results are biased.

The results imply that the different components in SOAP play important roles in spam detection. Their synergetic

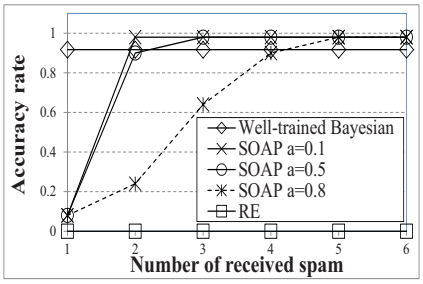


Fig. 7. Impersonation attacks.

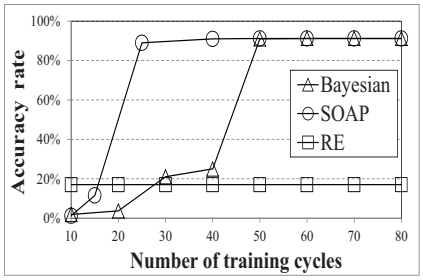


Fig. 8. Trust adjustment.

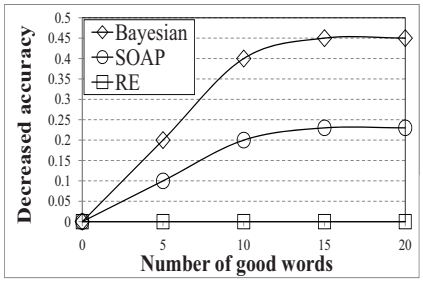


Fig. 9. Poison attack.

efforts contribute to the high accuracy of SOAP. The social closeness-based spam filtering component checks emails based on the closeness between the receiver and the sender. A smaller closeness value leads to more strict checking, while a larger closeness value leads to loose checking. Since the emails sent between people with close relationship normally has low probability to be spam, the spam detection accuracy rate can be increased. The interest-based spam filtering component increases the spam detection accuracy by filtering out the emails in the receiver's disinterests and accepts the emails that match the receiver's interests. The experimental results imply that the values of $\rho_I$ and $\rho_D$ in the interest-based spam filtering component need to be equal and large.

Figure 5 demonstrates the accuracy rate of SOAP and Bayesian with different training sample sizes. It shows that SOAP always produces a higher accuracy rate than Bayesian, and the accuracy rate of Bayesian increases as the sample size grows. SOAP has less dependency on data training because the social profile provides personal information for accurate spam detection. In contrast, Bayesian completely relies on data training and needs a significant amount of time to learn a new spam keyword. More training samples help to enhance its accuracy.

### C. Training Time

We further compare the training time needed for SOAP and Bayesian to achieve the same accuracy. We define a training cycle as the time period of training that enables a node to learn a new type of spam. Figure 6 shows the accuracy rate versus the number of training cycles. We observe that the accuracy of both SOAP and Bayesian increases as the training cycle increases. Unlike SOAP which exhibits slight increase because its accuracy is already very high, the accuracy of Bayesian grows rapidly. Moreover, Bayesian needs about 60 cycles to learn a certain category of keywords to reach the same accuracy rate as SOAP, while SOAP already achieves high accuracy with one training cycle. This is because the social interest-based spam filtering component infers the (dis)interests of users from their on-line social profiles. Instead of letting user manually train the Bayesian filter to detect the spam, SOAP can detect spam without learning based on social networks. Thus, SOAP does not need as much training time to achieve high spam detection accuracy rate.

### D. Resilience to Impersonation Attacks

In this experiment, we consider the impersonation attack in which a spammer impersonates the identities of the benign senders by forging their IDs or compromising their computers. We used a completely trained Bayesian with 6000 sample size to test how fast SOAP can adjust trust values to filter the spam from an impersonated node. To mimic the behavior of impersonation attacks, we randomly selected 4 random nodes to continuously send 10 spam emails to 4 nodes whose closeness values are $> 2$. The simulation finishes after 250 pair of nodes are selected. Recall that in SOAP, a node decreases another node's trust value if it receives spam from that node.

Figure 7 shows the average accuracy rate versus the average number of spam emails between the selected pairs. The figure shows that initially when a receiver receives spam from a highly-trustworthy sender, the spam detection accuracy is low. This is because SOAP trusts emails from close persons initially, and its closeness-based filtering component reduces the weight of spam keywords in emails sent from close people. Once the receiver detects the spam, it immediately reduces the trust value of the sender and checks the emails from the sender more strictly. This process is demonstrated in the figure. As more spam emails are received by the receivers, SOAP's accuracy rate grows rapidly. For SOAP with $a < 0.5$, its spam filtering accuracy rate is higher than Bayesian after receiving only one spam email. This is because using AIMD for trust adjustment, the accuracy rate of SOAP increases exponentially. A small $a$ leads to a high trust value decreasing rate. Because Bayesian has been already well-trained, its accuracy rate remains high. We notice that the accuracy rate of RE remains 0. RE always regards the email senders in the whitelist trustable. Therefore, if spammers impersonate the identities of the persons in the whitelist, their spam will be accepted by all in the impersonated persons' FOF network. As RE has no trust adjustment mechanism, the accuracy rate of RE is low.

In this experiment, in addition to legitimate emails, nodes randomly send spam emails to their close nodes. Bayesian is only trained with partial spam keywords. Figure 8 shows the accuracy rate versus the training cycle of Bayesian, SOAP and RE. The figure shows that it takes $20-30$ training cycles for SOAP to adjust the trust to an appropriate value that reaches the maximum spam detection performance. It also shows that Bayesian takes $50-60$ cycles to learn the new spam keywords. This result matches the observation in Figure 6 that Bayesian needs more training time without using a social network, while SOAP needs less cycles relying on a social network. The result verifies the effectiveness of the adaptive trust management

component in SOAP. RE exhibits poor performance with less than 20% accuracy rate. The reason is the same as in Figure 7.

### E. Resilience to Poison Attacks

In this experiment, we test the performance of Bayesian and SOAP under poison attacks. In this attack, extra legitimate keywords are added into spam to avoid to be detected. Figure 9 shows the decreased accuracy versus the number of legitimate keywords. As expected, the accuracy of both SOAP and Bayesian decreases. The detection accuracy of Bayesian decreases much faster than SOAP. This is because SOAP's interest-based spam filtering component only focuses on the receivers' (dis)interest keywords in the emails regardless of other legitimate keywords. Because a spam is always sent out by flooding, it is unlikely for a spammer to search the interests of an individual receiver to poison his/her spam filter. SOAP's personalized feature enables it to avoid poison attacks to a certain degree. In contrast, Bayesian is not personalized and is vulnerable to poison attacks. Some emails do not contain the (dis)interest keywords of the receiver. In this case, SOAP resorts to its basic Bayesian function for spam detection. Thus, its detection accuracy also decreases. Since RE is an identity-based spam filter, its accuracy is not affected by the content of an email. Thus, its performance does not change as more legitimate keywords are added into spam emails.

## V. Conclusion

A personalized, attack-resilient and user-friendly spam filter is needed to effectively combat spammers in order to reduce spam emails. However, most of current spam filters cannot meet these requirements. Some filters are vulnerable to spam filter attacks and some are not user-friendly. Few spam filters rely on social network to achieve personalized spam filtering to increase spam detection accuracy. In this paper, a SOcial network Aided Personalized and effective spam filter (SOAP) is proposed to meet the requirements. SOAP offers three new components to be integrated into the Bayesian filter. The social closeness-based spam filtering component prevents spam poison attacks, the social interest-based spam filtering component helps to realize personalized spam filtering, and the adaptive trust management component prevents impersonation attacks. More accurate spam filtering results function as input for Bayesian automatic training, leading to less need of user's effort to distinguish spam emails. The results of trace driven experiments show that SOAP improves the performance of Bayesian networks in term of spam detection accuracy and training time. In the future, we plan to build a real testbed for testing the performance of SOAP with real interaction samples.

## Acknowledgements

This research was supported in part by U.S. NSF grants NSF-OCI 1064230, CNS-1049947, CNS-1025652, CNS-1025649, and CNS-0917056, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

## References

[1] Messaging Anti-Abuse Working Group. MAAWG email metrics program. Technical report, First Quarter 2006 Report, 2006.
[2] Happy spamiversary! http://www.newscientist.com/.
[3] Tracking the high cost of spam. http://www.redcondor.com/company/.
[4] P. O. Boykin and V. Roychowdhury. Personal email networks: an effective anti-spam tool. *IEEE COMPUTER*, 2004.
[5] Cisco. Spammers continue innovation. http://ironport.com/.
[6] C. Binzel and D. Fehr. How social distance affects trust and cooperation: experimental evidence from A slum. In *Proc. of ERF*, 2009.
[7] L. F. Cranor and B. A. LaMacchia. Spams. *ACM Communications*, 1998.
[8] X. Carreras, L. Mrquez, and J. Salgado. Boosting trees for anti-spam email filtering. In *Proc. of RANLP*, 2001.
[9] P. Haider, U. Brefeld, and T. Scheffer. Supervised clustering of streaming data for email batch detection. In *Proc. of ICML*, 2007.
[10] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 1999.
[11] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Proc. of NIPS*, 2007.
[12] I. Kononenko. Semi-naive Bayesian classifier. In *Machine Learning*, 1991.
[13] S. J. Delany and P. Cunningham. An assessment of case-based reasoning for spam filtering. *Artifical intelligent review*, 2005.
[14] F. Fdez-Riverola, E. Iglesias, F. Diaz, J. R. Mendez, and J. M. Corchado. Spamhunting: sn instance-based reasoning system for spam labeling and filtering. *Decision Support System*, 2007.
[15] W. Zhao and Z. Zhang. Email classification model based on rough set theory. In *Proc. of AMT*, 2005.
[16] M. Uemura and T. Tabata. Design and evaluation of a bayesian-filter-based image spam filtering method. In *Proc. of ISA*, 2008.
[17] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Proc. of AAAI Workshop on Learning for Text Categorization*, 1998.
[18] J. S. Kong, P. O. Boykin, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury. Let your cyberalter ego share information and manage spam. *IEEE COMPUTER*, 2006.
[19] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. D. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proc. of Middleware*, 2003.
[20] SPAMNET. http://www.cloudmark.com.
[21] DNS Real-time Black List. http://www.dnsrbl.com/index.html.
[22] Spamhaus, http://www.spamhaus.org/sbl/index.lasso.
[23] blars.org, http://www.blars.org/.
[24] SpamCop Blocking List, http://spamcop.net/bl.shtml.
[25] O. Boykin and V. Roychowdhury. Personal email networks: an effective anti-spam tool. *IEEE COMPUTER*, 2004.
[26] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. RE: reliable email. In *Proc. of NSDL*, 2006.
[27] S. Hameed and P. Hui. Lens: Leveraging anti-social network against spam. Technical Report Technical Report No. IFI-TB-2010-02, Institute of Computer Science, University of Gttingen, Germany.
[28] P. Oscar Boykin and Vwani P. Roychowdhury. Leveraging social networks to fight spam. *Computer*, (4):61–68, 2005.
[29] J. James and J. Hendler. Reputation network analysis for email filtering. In *Proc. of CEAS*, 2004.
[30] C. Wilson, B. Boe, A. Sala, K. Puttasway, and B. Zhao. User interactions in social networks and implications. In *Proc. of EuroSys*, 2009.
[31] F. Heider. Attitudes and cognitive organization. *J. of Psychology*, 1946.
[32] J. F. Kurose and K. W. Ross. Computer networking: A top-down approach (5th edition). 2009.
[33] GFI Software. Why bayesian filtering is the most effective antispam technology. http://www.gfi.com/whitepapers/why-bayesian-filtering.pdf.
[34] G. Swamynathan, C. Wilson, B. Boe, K. C. Almeroth, and B. Y. Zhao. Can social networks improve e-Commerce: a study on social marketplaces. In *Proc. of WOSN*, 2008.
[35] S. MILGRAM. The small world problem. In *Psychology*, 1967.
[36] L. B. Koralov and Y. G. Sinai. Theory of probability and random processes. *Berlin New York Springer*, 2007.
[37] R. Brachman and H. Levesque. Knowledge representation and reasoning. *Morgan Kaufmann*, 2004.
[38] Spam Assassin, http://spamassassin.apache.org/.