Leveraging Social Networks For Effective Spam Filtering

Haiying Shen*, *Member, IEEE*, Ze Li, *Student Member, IEEE*

Abstract—The explosive growth of unsolicited emails has prompted the development of numerous spam filter techniques. Bayesian spam filters are superior to static keyword-based spam filters in that they can continuously evolve to tackle new spam by learning keywords in new spam emails. However, Bayesian spam filters are easily poisoned by clever spammers who avoid spam keywords and add many innocuous words in their emails. Also, Bayesian spam filters 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 in spam detection. In order to develop an accurate and user-friendly spam filter, we propose a SOcial network Aided Personalized and effective spam filter (SOAP) in this paper. In SOAP, each node connects to its social friends; that is, nodes form a distributed overlay by directly using social network links as overlay links. Each node uses SOAP to collect information and check spam autonomously in a distributed manner. Unlike previous spam filters that focus on parsing keywords (e.g, Bayesian filters) or building blacklists, SOAP exploits the social relationships among email correspondents and their (dis)interests to detect spam adaptively and automatically. In each node, SOAP integrates four components into the basic Bayesian filter: social closeness-based spam filtering, social interest-based spam filtering, adaptive trust management, and friend notification. We have evaluated the performance of SOAP using simulation based on trace data from Facebook. We also have implemented a SOAP prototype for real-world experiments. Experimental results show that SOAP can greatly improve the performance of Bayesian spam filters in terms of accuracy, attack-resilience, and efficiency of spam detection. The performance of the Bayesian spam filter is SOAP's lower bound.

Index Terms—Distributed overlays, Spam filtering, Social networks, Bayesian spam filters.

1 Introduction

Internet email is one of the most popular communication methods in our business and personal lives. However, spam is becoming a penultimate problem in email systems. Currently, 120 billion spam emails are sent per day [1], with a projected cost of $338 billion by 2013 [2]. Spam emails interfere with both email service providers and end users. A fundamental way to prevent spam is to make it unprofitable to send spam emails, thereby destroying the spammers' underlying business model [3]. Boykin *et. al.* indicated that there is a strong relationship between the average cost of sending spam, spam detection accuracy, and spam filter deployment by users [3]. Specifically, if a spam filter can stop spam from reaching users' inboxes with probability p and it is deployed by users with probability q, then it can increase the average cost of sending spam by $1/(1-pq)$. This means that in order to increase the cost of sending spam, 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 high accuracy. A more accurate filter generates less *false positives* and *false negatives*. *False positives* are legitimate emails that are mistakenly regarded as spam emails. *False negatives* are spam emails that are not detected. There are two primary types of spam filter attacks: *poison attacks* and *impersonation attacks*. In a poison attack, many legitimate words are added to spam emails, 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, in 2005 [4].

• ** Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*

• *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634.*
E-mail: {shenh, zel}@clemson.edu

By personalized, we mean that an accurate spam filter should consider the social context of a particular individual. First, it considers closeness between correspondents. A closer social relationship between two persons implies higher trust between them [5]. People treat emails sent by strangers and emails sent by acquaintances differently. Emails containing keywords such as "lose weight" are usually regarded as spam. However, such keywords may be in emails between members of a health club. 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, what constitutes a legitimate email differs from person to person. A user-friendly (i.e., easy-to-use) spam filter does not require a large amount of manual spam detection from users.

However, few previous spam filters can meet the requirements of being *user-friendly*, *attack-resilient*, and *personalized*. We list the main approaches in Table 1 with their features. Most approaches do not take into account the closeness relationships and (dis)interests of individuals. Previous spam filtering 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 keywords and patterns that are typical in spam. Machine learning approaches [6]–[14] (including the Bayesian filter [6]) train spam filters with a corpus of both spam and legitimate emails, identify their characteristics, which are used to automatically categorize future emails into two classes. However, these 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 much user effort to manually distinguish spam from legitimate emails for training.

Identity-based spam filters identify spam based on the identities of email senders. In the simplest method, a user maintains a whitelist and a blacklist for email addresses [19]–[22]. Social interaction-based spam filters [23]–[26] exploit friend's friend (FoF) relationships

TABLE 1: Features of previous spam filtering approaches.

Approaches	Person-alized	Attack-resilient		User-friendly
		impersonation	Poison	
Content-based spam filters				
Static keyword [15]	No	Yes	No	No
Machine learning [6]–[14]	No	Yes	No	No
Collaborative [16]–[18]	No	Yes	No	Yes
Identity-based spam filters				
Black/white list [19]–[22]	No	No	Yes	No
Social interaction-based [23]–[26]	No	No	Yes	Yes
Reputation [27]	No	No	Yes	No
Social network aided content and identity based spam filter				
SOAP	Yes	Yes	Yes	Yes

among email correspondents to create whitelists and blacklists. Since these spam filters do not consider email content, they are resilient to poison attacks. Additionally, because these filters automatically identify spammers from normal users according to their communication patterns, they are user-friendly. However, they are not personalized. Also, 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 [28]. Moreover, as indicated in [29], the assumption that person A's FoF is also A's friend is not generally true.

In this paper, we propose a SOcial network Aided Personalized and effective spam filter (SOAP) for spam detection to meet the three requirements. SOAP incorporates an social network (including social relationships and user (dis)interests) into the email network. In SOAP, users register their emails in SOAP client and are encouraged to provide their social information, such as (dis)interests, religions, occupations, affiliations and social relationships, in order to avoid spam. Each node connects to its social friends in the locally stored friendlist; that is, nodes form an overlay network by directly using social network links as overlay links. Each node uses SOAP to collect information and check spam autonomously. Different from social interaction-based spam filters [23]–[26] that only focus on the interaction via emails, SOAP also explores personal information in social networks and infers the relationship closeness and (dis)interests of individuals for more accurate spam detection.

SOAP in each node leverages social networks to combine four components into the basic Bayesian filter [6].

(1) Social closeness-based spam filtering. It calculates the node closeness based on social relationships. Since nodes with higher closeness have a lower probability of sending spam emails to each other, emails from nodes with lower closeness are checked more strictly and vice versa. This component makes SOAP resilient to poison attacks.

(2) Social interest-based spam filtering. It infers nodes' (dis)interests based on social profiles. The inferred information helps the filter enhance the accuracy of spam detection by considering individual preferences. 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) [30] to adjust the trust values of nodes. The trust value is used to tune closeness values in order to block emails from low-trust nodes or normal nodes impersonated by spammers.

(4) Friend notification. In order to strengthen SOAP's capability to combat impersonation attacks, a node quickly notifies its friends and FoF about a detected suspicious compromised node.

SOAP can rapidly determine spam and adapt to new spam keywords. It achieves high spam detection accuracy due to its personalized and attack-resilient features. In addition, it is user-friendly, since 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 basic Bayesian filter. Further, SOAP can collect the information in a decentralized manner, reducing the burden caused by information querying on the centralized server as in [24]. SOAP can greatly reduce the false positive spam detection rate. The lower bound performance of SOAP is the performance of the Bayesian spam filter. Unlike current online social networks that use centralized servers to manage user accounts, in SOAP, a user's social account is independently managed by the user's SOAP email client in the distributed network. We implemented SOAP email client in simulation and real-world prototype experiments. Experimental results show the superior performance of SOAP in comparison with other methods.

2 Related Work

The vast quantity of spam emails distributed blindly in bulk has stimulated many spam filtering approaches. These approaches can be mainly categorized into two classes: content-based and identity-based.

Content-based Approaches. The basic approach of content-based spam filtering is the static keyword list [31], which however makes it easy for a spammer to evade filtering by tweaking the message. The second category of content-based approaches includes machine learning-based approaches such as Bayesian filters [6], decision trees [8], [9], Support Vector Machines [10], [11], Bayes Classifiers [12], [13] and combinations of these techniques [14]. In this approach, a learning algorithm is used to find the characteristics of the spam and of legitimate emails. Then, future messages can be automatically categorized as highly likely to be spam, highly likely to be legitimate emails, or somewhere in between.

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 avoid the spam later on by querying others to see if their received emails are spam or not. SpamNet [18] uses a central server to connect all participants of the collaborative spam filter. SpamWatch [17] is a distributed spam filter based on the Tapestry Distributed Hash Table system. Kong *et al.* [16] proposed a distributed spam filter to increase the scalability of centralized collaborative spam filters.

Identity-based Approaches. The simplest identity-based spam filtering approaches are blacklist and whitelist [19]–[22], which check the email senders for spam detection. Whitelists and blacklists both maintain a list of addresses of people whose emails should not and should be blocked by the spam filter, respectively. One server-side solution [22] records the number and frequency of the same email sent to multiple destinations from specific IP addresses. If the number and frequency exceed thresholds, the node with the specific IP address is blocked.

Boykin *et al.* [23], [26] 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 rationale that the social communication network of a normal node has a higher clustering coefficient than that of a spam node. RE [24] is a whitelist spam filtering system based on social links. It is based on the assumption that all friends and FoF are trustable. Hameed [25] proposed LENS, which extends the FoF network by adding trusted users from outside of the FoF networks to mitigate spam beyond social circles. Only emails to a recipient that have been vouched by the trusted nodes can be sent into the network. DeBarr *et al.* [32] evaluated the use of social network analysis measures to improve the performance of a content filtering model. They tried to detect spam by measuring the degree centrality of message relay agents and the average path length between senders and receivers. They claimed that the messages from a promiscuous mail relay or messages with unusual path lengths that deviate from the average are more likely to be spam. Lam *et al* [33] proposed a learning approach for spam sender detection based on user interaction features (e.g., indegree/outdegree and interaction frequency) extracted from social networks constructed from email exchange logs. Legitimacy scores are assigned to senders based on their likelihood of being a legitimate sender. Tran *et al* [34] implemented an email client called SocialEmail, which provides social context to messages using a social network's underlying social graph. This not only gives each email recipient control over who can message him/her, but also provides the recipient with an understanding of where the message socially originated from. However, if a spammer compromises a legitimate user's computer, the spammer can easily attack the user's friends in the social network, which is characterized by high clustering and short paths [28]. Also, such social interaction-based methods are not sufficiently effective in dealing with legitimate emails from senders outside of the social network of the receiver. Golbeck *et al.* [27] proposed an email scoring mechanism based on an email network augmented with reputation ratings. An email is considered spam if the reputation score of the email sender is very low. Different from these social network based methods, SOAP focuses on personal interests in conjunction with social relationship closeness for spam detection. There are other approaches not belonging to the above two classes [35]–[41]. Due to space limit, we do not present the details of these methods.

3 SOAP: Social Network based Bayesian Spam Filter

3.1 Overview of the Basic Bayesian Spam Filter

A 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, a user is given a pool of emails, and s/he will manually indicate 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 in 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. The probability that an email including a word w is spam is:

$$P(S|w) = \frac{P(S,w)}{P(w)} = \frac{P(w|S)P(S)}{P(w)} \tag{1}$$

$$= \frac{P(w|S)P(S)}{P(w|S)P(S) + P(w|L)P(L)}. \tag{2}$$

Then, the probability that an email including a set of keywords W is spam is:

$$P(S|W) = \frac{P(S,W)}{P(W)} = \frac{\prod_i P(w_i|S)P(S)}{\prod_i P(w_i|S)P(S) + \prod_i P(w_i|L)P(L)}. \tag{3}$$

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

3.2 Overview of SOAP

SOAP is a social network based personalized, attack-resilient, and user-friendly Bayesian spam filter. In the email network, each node uses SOAP to filter spam in a distributed manner. Unlike current social network based filters that focus on email interaction networks [16], [24], [25], [27], SOAP further leverages social information including personal (dis)interests and social relationships. SOAP encourages users to indicate their (dis)interests and social relationships with their email correspondents in order to receive less spam and lose less legitimate emails. Using this information, nodes form a distributed overlay by connecting to their friends; that is, nodes use social network links directly as overlay links.

Fig. 1 shows the structure of SOAP in each node. SOAP integrates four new components into the Bayesian filter: (1) social closeness-based spam filtering, (2) social interest-based spam filtering, (3) adaptive trust management, and (4) friend notification. 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 their corresponding weights showing the probability that the email containing the keyword is spam. 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 therefore adjusted based on the closeness between the receiver and the sender, the receiver's (dis)interests, the receiver's trust of the sender, and the received spammer notification from friends. If the closeness is high, the likelihood that they send spam to each other is low, so the weight is decreased, and vice versa. However, it is possible that close nodes are compromised. This problem is resolved by the trust management and friend notification components. 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 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, which also reduces the training time. In the section below, the following issues will be addressed.

- How is the closeness of individuals calculated in a distributed manner and how is consideration of closeness integrated into the Bayesian filter? (Section 3.3)
- How are the (dis)interests of individuals inferred and integrated the email preference consideration into the Bayesian filter? (Section 3.4)
- How are trust values adjusted to avoid impersonation attacks? (Section 3.5)
- How does a node notify friends when it detects a compromised node? (Section 3.6)
- How do the different components in SOAP cooperate for spam detection? (Section 3.7)

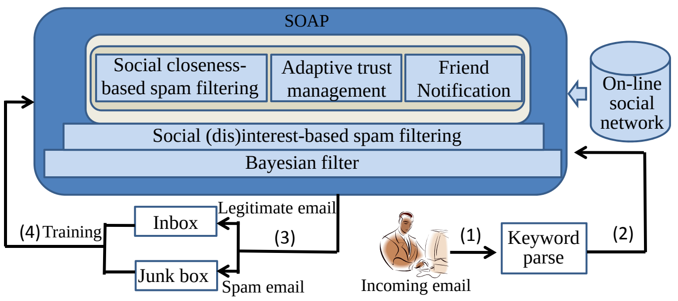


Fig. 1: The structure and workflow of SOAP.

3.3 Social Closeness-based Spam Filtering

When a person receives an email from another socially close person, the email has a low probability of being spam unless the email sender's machine is under an impersonation attack. Thus, the social closeness between individuals can be utilized to improve the accuracy of spam detection. Note that, in a social network, people treat others differently based on their social closeness. People impose different levels of interest, trust, or tolerance to the emails from others with different social closeness. People with close social relationship are willing to receive emails from each other. On the other hand, receivers may have less interest in or tolerance for emails containing spam keywords from senders that are socially far away. We regard spam as emails that receivers are not interested in. Therefore, we need to differentiate emails from persons with different social closeness. 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 that is used in spam detection to numerically measure social closeness between two persons. SOAP relies on nodes' social relationships, such as kinship and friendship, to determine node closeness values. SOAP sets different weights for different social relationships to measure node closeness. For example, the closeness of a kinship relationship usually weights more than a business relationship. We use $c(u,v)$ to denote the weight of a relationship between node u and v. Below, we introduce how to calculate the closeness of adjacent nodes and non-adjacent nodes in a social network.

3.3.1 Node Closeness

In a social network, more relationships between two adjacent nodes make them closer. Thus,

$$C(u,v)=\sum_{i=1}^{n} c_i(u,v), \tag{4}$$

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

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. For example, if node A is D's father and E is D's best friend, A is unlikely to send spam to E. The closeness transitivity should capture three properties in order to correctly reflect the social relationship.

Property 3.1: *Closeness propagation property.* The closeness between node u and node k_i exponentially decreases as their distance increases. As shown in Fig. 2, it can be illustrated by $C(u,k_i)=C(u,k_1)\cdot\varepsilon^{i-1}$, where $\varepsilon<1$ is not necessarily a constant.

Thus, the more hops that exist 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.* [5] discovered that a reduction in social distance between two persons significantly increases the trust between them. Swamynathan *et al.* [42] found that people normally do e-commerce with people within 2-3 hops in their social networks.

Property 3.2: *Weakest link property.* The weakest link in a social path (not necessarily a disjoint path) is the direct link between adjacent nodes that has the minimum closeness, denoted by $\min_{1\le i\le n} C(k_i,k_{i+1})$. 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 between, $C(u,v)<\min_{1\le i\le n} C(k_i,k_{i+1})$, where node k_i is in the path between u and v.

Fig. 3 shows the weakest link property. The intuition behind this property is that the closeness value between any pair of non-adjacent nodes u and v is less than the closeness value between u and k, and k and v, where k is a node in the social path between u and v [43]. Suppose the link between adjacent nodes k_i and k_{i+1} in the path from u to v that has the smallest closeness value $C(k_i,k_{i+1})$. Then, $C(u,v)<C(u,k_i)<C(k_i,k_{i+1})$. That is, $C(u,v)<\min_{1\le i\le n} C(k_i,k_{i+1})$.

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 paths between them, their closeness through p paths denoted by $C(u,v,p)$ is

$$C(u,v,p)=\sum_{j=1}^{p} C_j(u,v). \tag{5}$$

Fig. 4 shows the closeness accumulation property. The closeness value between node u and node v is the sum of the closeness values of individual paths between node u and node v. 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 call this the closeness accumulation property. We then design a closeness calculation formula that can meet the above three properties:

$$C(u,k_{i+1},p)=\sum_{j=1}^{p}\left(C_j(u,k_i)\cdot(C_j(k_i,k_{i+1})/\varphi)^i\right) \tag{6}$$

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

$$\varphi>\max_{1<x<i}\left(C(k_{x-1},k_x)\cup C(u,k_1)\right). \tag{7}$$

Equ. (7) indicates that φ is larger than any closeness value between two adjacent nodes in the path from u to v, which ensures that $C_j(k_i,k_{i+1})/\varphi<1$ in Equ. (6). Therefore, for each social path j from node u, the social closeness value $C(u,k_{i+1})$ decreases exponentially based on i, which meets Property 3.1.

For each social path j, we have:

$$C(u,k_{i+1})=C(u,k_i)\cdot(C(k_i,k_{i+1})/\varphi)^i. \tag{8}$$

Since $C(u,v)=C(u,k_{n-1})\cdot(C(k_{n-1},k_n)/\varphi)^{n-1}$, we can recursively get:

$$C(u,v)=C(u,k_{x+1})\cdot\Big(\prod_{x+1\le i\le n-1}(C(k_i,k_{i+1})/\varphi)^i\Big)$$
$$<C(u,k_{x+1}). \tag{9}$$

Suppose $$C(k_x,k_{x+1})=\min_{1\le i\le n} C(k_i,k_{i+1}). \tag{10}$$

From Equ. (6), we can get:

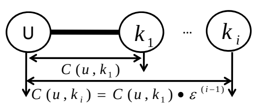


Fig. 2: Closeness propagation property.

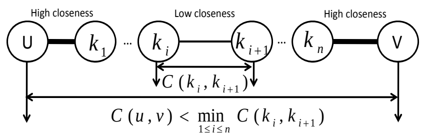


Fig. 3: Weakest link property.

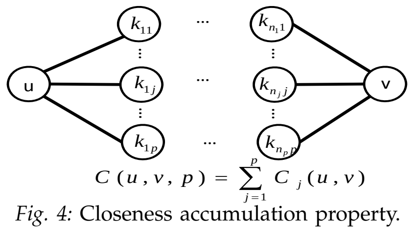
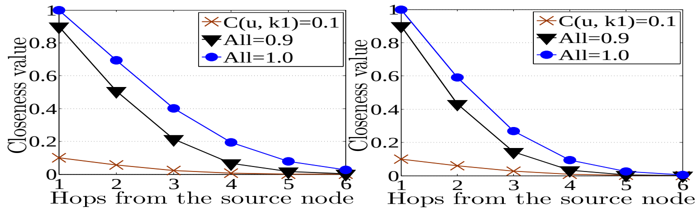
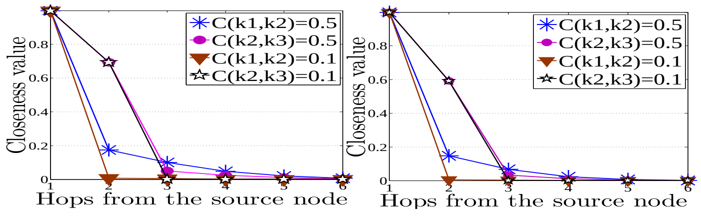


Fig. 4: Closeness accumulation property.



(a) φ=1.2 (b) φ=1.3

Fig. 5: Closeness propagation.



(a) φ=1.2 (b) φ=1.3

Fig. 6: Weakest link.

$$\begin{aligned} C(u,k_{x+1}) &= C(u,k_x)\cdot(C(k_x,k_{x+1})/\varphi)^x \\ &< C(u,k_x)\cdot(C(k_x,k_{x+1})/\varphi) \\ &= (C(u,k_x)/\varphi)\cdot C(k_x,k_{x+1}). \end{aligned} \tag{11}$$

As $C(u,k_x) < C(u,k_1) < \varphi$, $(C(u,k_x)/\varphi) < 1$. Thus, from Formula (11), we get:

$$C(u,k_{x+1}) < C(k_x,k_{x+1}). \tag{12}$$

From Formulas (9) and (12), we can get $C(u,v) < C(k_x,k_{x+1})$. That is, $C(u,v) < \min_{1\le i\le n} C(k_i,k_{i+1})$ for path j based on Formula (10), which meets Property 3.2.

By summing up all $C(u,k_{i+1},p_j)$ for different p different paths, Equ. (6) satisfies Property 3.3. In order to verify that the closeness provided by Equ. (6) meets the expected properties, we calculate the closeness with different parameter values and show the results in Figures 5 and 6. The closeness of the link of each pair of adjacent nodes was set to 1 unless otherwise specified in the figure. "All=0.9" means the closeness of each link of adjacent nodes was set to 0.9. Fig. 5 shows that the closeness between nodes and the source node decreases exponentially as the distance to the source increases, which verifies Property 3.1. "All=1.0" leads to higher closeness values than "All=0.9" because the closeness value of adjacent nodes in "All=1.0" is higher. "C(u,k_1)=0.1" generates much lower closeness values than "All=0.9", which confirms that the closeness of a node to the source highly depends on the closeness of its previous nodes to the source.

Fig. 6 shows how the weakest link affects the closeness between nodes in a path. The figures show that the closeness between a node and the source decreases sharply at the weakest link. For example, in both figures, the closeness in lines "C(k2, k3)=0.1" and "C(k1, k2)=0.5" drops sharply at nodes $k2$ and $k1$. Also, the closeness between the source and a node succeeding the weakest link is always less than the link's closeness value. For example, in line "C(k1, k2)=0.1", the closeness of the node is two hops away from the source, $C(u,k2) < 0.1$. In line "C(k2, k3)=0.5", $C(u,k2) < 0.5$. The results verify Property 3.2.

By comparing Fig. 5(a) with Fig. 5(b) and Fig. 6(a) with Fig. 6(b), we can find that a larger φ leads to a faster decrease in the closeness value. This implies that we can adjust φ to model different kinds of social relationships such as kinship, friendship, familiar stranger, etc.

3.3.2 Distributed Closeness Calculation Algorithm

In a social network, each person has a friend list. Based on the social relationship of his/her friends, the closeness values with adjacent friends can be calculated. Most current social networks have a central server to store all individuals' information 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 as an alternative for the closeness calculation. In the algorithm, a source node sends a query message with a specified TTL along the FoF links. Upon receiving the message, an intermediate node decreases the TTL by 1, inserts its closeness values with its neighbors into the message, then forwards it to all its neighbors. The process continues until the TTL becomes 0. Then, the destination nodes directly sends the message back to the source node. Subsequently, the source node retrieves all closeness values of the nodes in the path to the destination; it can then calculate its closeness with each of node using Equ. (6).

Algorithm 1 Distributed closeness calculation algorithm.

```
 1: Send a query message with TTL
 2: if Receive a response from destinations then
 3:     Calculate its closeness with each node using Equ. (6)
 4: end if
 5:
 6: if Receive a query initiated by node $i$ then
 7:     Insert its closeness with node $i$ to the message
 8:     TTL=TTL-1
 9:     if TTL¿0 then
10:         Forward the message to its neighbors
11:     else
12:         Send the message to node $i$
13:     end if
14: end if
```

It was shown that the average number of hops between any two persons is less than or equal to 6 [44]. Therefore, when a node receives an email, the social distance of the sender to the receiver is less than or equal to 6 on average. A node needs the closeness information of the sender for spam detection. Thus, each node should collect the closeness information of the nodes within a certain distance from itself. Hence, we can set TTL=3 for two reasons: (1) Sending the message along more hops produces high overhead, and (2) Property 3.1 indicates that closeness decreases exponentially. The closeness value is decreased to an extremely small value when the distance exceeds 3 hops [5], [42]. Therefore, the source has very low closeness to the nodes far away from itself, and the emails from these nodes should be strictly checked. Algorithm 1 shows the pseudocode of this distributed closeness calculation algorithm. The number of message transmission hops in information collection from one node is $O(n^3)$, where n

denotes the number of neighbors of a node.

3.3.3 Integration with Bayesian Filter

In the Bayesian filter, each of an email's keywords is weighted to show the probability that an email containing the keyword is spam. SOAP adjusts the keyword weights based on the closeness between the email receiver and 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 strictly checked. The keywords tuning function is:

$$P(S|w) := \begin{cases} P(S|w)e^{-\varphi_f \cdot (C(u,v)-\varphi_t)} & \text{if } C(u,v) \geq \varphi_t; \\ P(S|w)\xi \ (\xi \geq 1) & \text{if } C(u,v) < \varphi_t. \end{cases} \tag{13}$$

where $P(S|w)$ is the weight of a keyword, φ_f is a scale parameter to adjust the decreasing rate of $P(S|w)$, and φ_t is a location parameter to determine the origin for exponential decreasing [45]. If $C(u,v) = \varphi_t$, then the weight is not changed. If $C(u,v) > \varphi_t$, then $P(S|w)$ is decreased by a factor $e^{\varphi_f \cdot (C(u,v)-\varphi_t)}$. If $C(u,v) < \varphi_t$, then $P(S|w)$ is increased by a factor ξ ($\xi \geq 1$). ξ can be adjusted by users with different accuracy requirements. Higher ξ requires the email to have a higher probability to be regarded as spam. ξ normally is set to be 1 in order to reduce false positives.

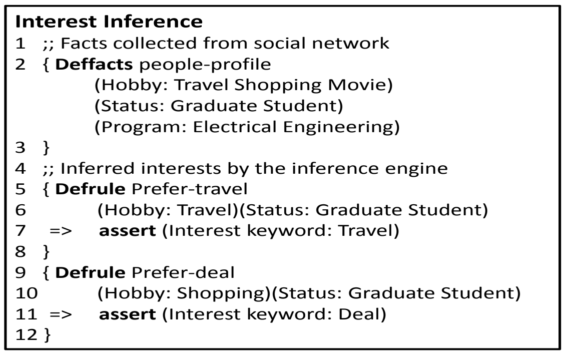
3.4 Social Interest-based Spam Filtering

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 with the email receiver's social interests and disinterests, SOAP increases and decreases the probability of these keywords to be spam, respectively.

3.4.1 Node (Dis)interest Inference

SOAP relies on a rule-based inference system [46] to infer each user's (dis)interests. The inference system has three components: profiles, inference rules, and inference engines. The *profile* component is a database containing all useful facts parsed from the user' profile in the social network including interests, occupations, and affiliations. 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 common sense such as "Most of the birds can fly". 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. Fig. 7 shows an example of the rule-based inference method. All the facts are built into a fact database. Numerous rules are made for the inference engine based on the non-monotonic logic.

Since SOAP derives user (dis)interests from user profiles in the social network, SOAP's spam filtering accuracy based on user interests could be affected if some users do not provide detailed, accurate, or complete profiles. When users register for SOAP, they will be informed that the accuracy of the information in their profiles affects the accuracy of spam detection. Users usually will not enter false information to their profiles since they hope to deter their received spam more accurately. In Section 4, we present how profile completeness affects the false negative rate and false positive rate in spam filtering (Fig. 10 and Fig. 13), respectively. The experimental results show that the more information a user provides, the less false



```
Interest Inference
;; Facts collected from social network
{ Deffacts people-profile
        (Hobby: Travel Shopping Movie)
        (Status: Graduate Student)
        (Program: Electrical Engineering)
}
;; Inferred interests by the inference engine
{ Defrule Prefer-travel
        (Hobby: Travel)(Status: Graduate Student)
 =>     assert (Interest keyword: Travel)
}
{ Defrule Prefer-deal
        (Hobby: Shopping)(Status: Graduate Student)
 =>     assert (Interest keyword: Deal)
}
```

Fig. 7: An example of email preference inference.

positive rate and false negative rate his/her SOAP filter can provide. Therefore, if users want to gain high spam filtering accuracy rate, they would provide more detailed profile information for SOAP. Also, even some users do not provide much profile information, as our experimental results (Fig. 15) show, the spam filtering accuracy will be increased finally after the users use SOAP for a long time (i.e., SOAP is trained with more training samples).

3.4.2 Integration with Bayesian Filter

SOAP is a personalized spam filter since it considers individual (dis)interests in spam detection. The basic Bayesian filter calculates the weight (i.e., probability) that an email containing a keyword is spam. Recall that SOAP adjusts the weights of keywords in an email according to social closeness between the email receiver and sender. After that, SOAP further adjusts the weights according to the email receiver's social (dis)interests. 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 regarded as 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 regarded as spam. Then, SOAP relies on the basic Bayesian filter (Section 3.1) for spam detection.

For a spam keyword within the email receiver's interests, its weight is tuned by:

$$P(S|w_{interest}) := P(S|w_{interest}) \cdot e^{-\rho_I} \tag{14}$$

where $w_{interest}$ is the spam keyword in interests and ρ_I is a scale parameter. As ρ_I increases, $P(w_{interest})$ decreases. Therefore, the probability that the email is considered to be spam decreases. As a result, emails within a receiver's interests 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 the email receiver, the weight of the keyword is adjusted by

$$P(S|w_{disinterest}) := P(S|w_{disinterest}) \cdot e^{\rho_D} \tag{15}$$

where $w_{disinterest}$ is the spam keyword in the email receiver's disinterests. As the scale parameter ρ_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 the email receiver's disinterests, it has a high probability of being regarded as spam. The more disinterest keywords an email has, the higher the probability that 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 the disinterests of a person. In this way, SOAP resists spam poison attacks. Note that the inference results may not match the actual interests of a person. This problem can be resolved in the training step (step (4) in Fig. 1). SOAP not only can learn the spam keywords, it also learns the interests and disinterests of the users. If a legitimate email is falsely regarded as spam, the disinterest keywords in the legitimate email are deleted from the disinterest keyword list. Similarly, if a spam is falsely regarded as legitimate email, the interest keywords are also deleted from the interest keyword list.

3.5 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 or report fake relationships with neighbors to the source node during the social closeness calculation process. Due to the power-law and small-world characteristics of social networks, in which nodes are highly clustered, impersonation can spread spam extremely fast. In order to combat 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) [30] to adjust node trust. Node trust is then used to update node closeness for the detection of false negatives due to impersonation attacks.

AIMD is a feedback control algorithm used in TCP congestion avoidance. It combines linear growth of the congestion window with an exponential reduction during congestion. 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 i initially assumes node j with high closeness is trustworthy until it receives a spam email (i.e., false negative) from node j.

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 i changes the trust value of j by

$$t_{(i,j)} := a \cdot t_{(i,j)} \ (0 < a < 1). \tag{16}$$

When node i receives a legitimate email from sender j, then

$$t_{(i,j)} := t_{(i,j)} + b \ (0 < b < 1). \tag{17}$$

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. A smaller a (i.e., a faster trust decrease) and b (i.e., a slower trust increase) lead to fewer false negatives, but more false positives. On the other hand, a larger a and b result in fewer 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

$$t_{(i,j)} := t_{(i,j)} + \alpha \cdot b \ (\alpha > 1). \tag{18}$$

In order to reach the optimal point between the false negatives and false positives, parameters a and b are 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. The two functions are formally presented in math below, when $n_{fn1} < n_{fn2}$, then $\mathrm{F}(n_{fn1}) > \mathrm{F}(n_{fn2})$ (i.e., $a_1 > a_2$); if $n_{fp1} < n_{fp2}$, then $\mathrm{F}(n_{fp1}) < \mathrm{F}(n_{fp2})$ (i.e., $b_1 < b_2$) [47]. Briefly, a larger n_{fn} leads to a smaller a, and a larger n_{fp} leads to a 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 the 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:

$$C(i,j) := t_{(i,j)} \cdot C(i,j). \tag{19}$$

The closeness value $C(i,j)$ is then used to adjust the weights of keywords in the social closeness-based spam filtering component.

3.6 Friend Notification Scheme

Due to high clustering in social networks, once a spammer compromises a node in the social network in an impersonation attack, the spammer can quickly send spam to the close friends of the compromised node in its social network. As the spam receivers are socially close to the compromised node, their filters will loosely check the spam emails from the compromised node and receive the spam.

Since a compromising spammer always sends spam to the compromised node's socially close friends, SOAP takes advantage of the power of a collective of socially close nodes to combat spam in impersonation attacks, in addition to letting nodes individually fight against spam with the adaptive trust adjustment. That is, a node notifies its friends when it identifies a compromised node. This can help its friends to avoid spam more quickly. Thus, we propose a friend notification scheme to enhance the effectiveness of SOAP in avoiding spam in impersonation attacks.

Recall that in the adaptive trust management, when a node receives a spam email, the trust value of the email receiver on the email sender, $t_{(i,j)}$, is reduced. Thus, when node i finds that it keeps on receiving spam from a certain node j, $t_{(i,j)}$ is continuously reduced. When the trust value is less than a threshold value T_t, node i regards node j as a spammer and broadcasts a warning message containing the ID of suspicious node j to all its friends in its social network. As the nodes in a social network are highly clustered [48], the friends of node j are very likely to be the friends of node i. According to Equ. (6), the social closeness value between two nodes with a distance beyond 2 hops is very small. Therefore, there is no need to notify friends 2 hops away in the social network as they already strictly check the emails from the senders who are two hops away in the social network according to Equ. (13). Therefore, in order to reduce the notification overhead, we confine the number of broadcasting hops to 2. The nodes that receive more than m notification messages about node j from different reporters add the ID of node j into their blacklists. The value m is called blacklist threshold. Otherwise, the notification receiver ignores the notification. A large m can prevent compromised users conduct sybil attack to mis-accuse a legitimate user as a spammer. Also, since the notifications are sent between friends in the social network, node j is very also likely to receive the notification messages accusing itself. If node j is not a spammer, it checks if itself has been compromised. In addition, as user j and the reporters are socially close, user j contacts the reporters offline (e.g., through telephone) to inform them to check if their computers are compromised. The user j report to the administrator about the accuse. If a node receives an email from node j in its blacklist, rather than adjusting the weights of keywords in the email based on closeness, the node increases the weights of these keywords:

$$P(S|w) = P(S|w)\xi \ (\xi \geq 1). \tag{20}$$

A spammer may use a Sybil attack to mislabel a legitimate user as a spammer. The Sybil attack problem has been addressed in many works [49] and it is out of the scope of this paper. Also, we can use the methods in the previous identity-based spam filtering approaches [19]–[22] to decide when the nodes in the blacklist get out of the blacklist. This problem is also out of the scope of this paper.

3.7 The Integration of Components in SOAP

SOAP is a user-oriented spam filter. Each user will run SOAP independently to detect spam. Fig. 1 shows how the different components in SOAP cooperate with each other for spam examination of an incoming email. Algorithm 2 shows the pseudocode of the workflow process. Using the social closeness-based spam filtering algorithm, a node calculates the closeness between other nodes and itself, and keeps a list of the closeness values. In phase (1) in Fig. 1, when a node receives an email, SOAP parses out the keywords in the email (Line 2).

For each keyword, the basic Bayesian filter keeps the weight (i.e., probability) that an email containing the keyword is spam. In phase (2), the different components adjust the probability in order to increase the accuracy of spam detection. First, if the sender is not in the blacklist, based on the closeness between the email sender and receiver, the probability of each keyword is adjusted based on Equ. (13). Otherwise, the probability of each keyword is adjusted based on Equ. (20) (Lines 4-12). Then, 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 Equ. (14) (Lines 15-17). On the other hand, if the keywords are in the disinterest list of the receiver, it means the email is useless to the receiver. Then, the probabilities of these keywords are increased based on Equ. (15) (Lines 18-20). 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* or the *Junk box* correspondingly. The results are used for spam detection training in phase (4) (Lines 21-22). 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 adaptive trust management in SOAP adjusts node trust and node closeness accordingly (Equ. (19)) (Lines 23-24). If the trust value is lower than a threshold T_t, the friend notification mechanism is used (Lines 25-28). The goal of trust management and friend notification is to counter impersonation attacks, i.e., reducing false negatives while restricting false positives.

The social information about an individual is dynamic. For instance, the friends and dis(interest) of a user may change overtime. This is a problem for all social network based spam filters [24], [25]. SOAP can periodically update the social information for spam detection. Untimely update would affect the accuracy of spam detection. Therefore, the time period for the update should be appropriately determined by the dynamic degree. The performance of the Bayesian spam filter is still SOAP's lower bound.

Algorithm 2 The process of the spam detection in SOAP.

```
1: for each incoming email $e$ do
2:   $\mathcal{K}$= parsed keywords in the email
3:   Retrieve the weights of these keywords in Bayesian filter
4:   //closeness-based weight adjustment when the email sender is not in
     the blacklist of the email receiver
5:   if e.sender is not in the blacklist of e.receiver then
6:     //calculate the social closeness between sender and receiver and
       update the weight of the keywords
7:     Calculate C(e.sender,e.receiver) based on Equ. (6)
8:     Adjust every keyword's weight based on Equ. (13)
9:   else
10:    //weight adjustment when the sender is in the blacklist of the email
       receiver
11:    Increase every keyword's weight based on Equ. (20)
12:  end if
13:  //execution of interest-based weight adjustment
14:  for each keyword $k$ in $\mathcal{K}$ do
15:    if $k$ matches the interests of e.receiver then
16:      Decrease $k$'s weight according to Equ. (14)
17:    end if
18:    if $k$ matches the disinterests of e.receiver then
19:      Increase $k$'s weight according to Equ. (15)
20:    end if
21:    Calculate the weight of $e$ according to Equ. (3)
22:    Email classification
23:    //update trust value of the email receiver to the email sender based
       on the classification result
24:    Update trust value on e.sender according to Equ. (19).
25:    if trust value of e.receiver on e.sender is less than $T_t$ then
26:      //send a notification message to inform friends within 2 hops in
         e.receiver's social network
27:      Send friend notification message with TTL=2
28:    end if
29:  end for
30: end for
```

4 Performance Evaluation

We evaluated the performance of SOAP compared to the Bayesian filter [6] (Bayesian in short) and the RE interaction-based spam filter [24] through simulations. The false negative (FN) rate is the number of false negatives divided by the total number of emails, and the false positive (FP) rate is the number of false positives divided by the total number of emails.

We 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. 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 as classical music and the book *Gone with the Wind* is classified as 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. The average completeness of a personal profile is about 50%.

We collected 9500 emails including 2000 spam emails and 7500 legitimate emails from the spam-assassin project [50], and 500 commercial emails from the email

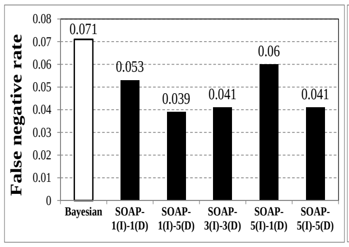


Fig. 8: FN rate.

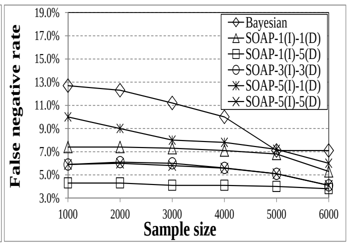


Fig. 9: FN rate vs. # of emails.

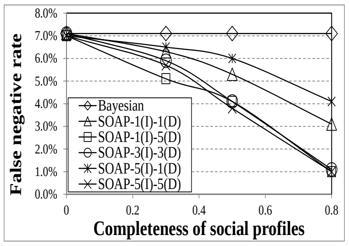


Fig. 10: FN rate vs. profile completeness.

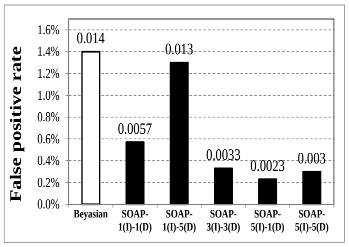


Fig. 11: FP rate.

TABLE 2: A list of SOAP parameters.

Parameter	Value
Closeness value	Kinship=InRelationship=2; Colleague=Classmate=1.5; Familiar=1;
Closeness scale rate	φ= 1.3
Scale parameter of closeness	$\varphi_f = 1$
Location parameter of closeness	φ_t=1
Scale parameter ρ_I and ρ_D	$\rho_I = 3$, $\rho_D = 3$
AIMD parameters	a=0.5, b=0.1
Notification parameter	m=1

boxes of five members in our lab. The size of the training sample for Bayesian and SOAP was 6000 emails randomly chosen from the total emails. The parameters used in the simulation are shown in Table 2. We determined the closeness values of different social relationships based on the intuition on the roles and strength of social relationships. A higher closeness value of a relationship indicates that this relationship is more trustable. Based on Equ. 13, we set $\varphi_t = 1$ in order to set the "familiar" relationship with $C(u, v) = 1$ as the boundary relationship to decrease or increase the keyword weights $P(S|w)$. Based on our experiments, we found that $\varphi_f = 1$, $\varphi = 1.3$, $\rho_I = 3$ and $\rho_D = 3$ can provide a moderate sensitivity to the increase or decrease of the (dis)interest keyword weights, and a=1 and b=0.5 can provide a moderate sensitivity to adjust the social closeness between two users. Thus, we set the parameters to these values. As we focus on spam filtering rather than malicious attack prevention, the parameter m in the friend notification scheme is set to 1 by default.

We continuously chose 4 random pairs of nodes at a time and let them send an email to each other until 40000 pairs were chosen. The email sent out 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. We manually judged whether an email is a spam email based on the social relationship between the sender and the receiver and on the (dis)interests of the receiver. The accuracy of a spam filter is determined by comparing the spam filter's results with the manually judged results. In the experiments, unless otherwise indicated, we excluded the friend notification component from SOAP in order to test its performance with other components.

4.1 False Negative Rate

Fig. 8 shows the average FN rate in Bayesian and SOAP with different ρ_I and ρ_D. We use SOAP-x(I)-y(D) to denote SOAP with interest scale parameter ρ_I=x and ρ_D=y. The figure shows that the FN rate of Bayesian is higher than that of SOAP with different ρ_I and ρ_D. The improved performance of SOAP is attributed to its interest-based spam filtering component. The reduced FN emails are the emails that match the receivers' disinterests, which are generally regarded as legitimate emails by others. We see that SOAP-1(I)-5(D) has the lowest FN. This is because by giving higher weight to disinterest keywords than interest keywords, SOAP strictly checks emails by disinterests. In contrast, SOAP-5(I)-1(D) gives higher weight to interest keywords, so if an email matches the user's interests SOAP regards the email as legitimate, even if it contains more disinterest keywords than interest keywords. Because SOAP-3(I)-3(D) has the same weights for disinterests and interests, and SOAP-5(I)-5(D) also has the same weights for disinterests and interests, they generate much lower FN rates than SOAP-5(I)-1(D). Recall that a higher keyword weight means a higher probability that an email is considered spam. Since the keyword weight in SOAP-1(I)-1(D) is smaller than SOAP-3(I)-3(D) and SOAP-5(I)-5(D), its FN rate is higher than theirs according to Equations (14) and (15).

The personalization of SOAP is demonstrated in Fig. 9, which plots the FN rate versus the training sample size. We still see that SOAP outperforms Bayesian. The FN rate of Bayesian decreases as the sample size grows since more samples enable Bayesian to learn more about the users' personal preferences. In SOAP, some emails that disinterest a receiver can be determined directly from the personal profile without training; thus, its FN rate does not greatly vary as the sample size increases. The figure also shows that as the sample size increases, the FN rate of SOAP-5(I)-1(D) approaches that of SOAP-1(I)-1(D), and the FN rates of SOAP-3(I)-3(D) and SOAP-5(I)-5(D) approach that of SOAP-1(I)-5(D). This is because a well trained Bayesian component can learn those (dis)interest keywords, which enable SOAP to detect the spam that cannot be identified by the (dis)interest keywords checked by the poorly trained Bayesian component.

Fig. 10 shows the FN rate versus the completeness of social information. We can observe that the FN rates of SOAP and Bayesian are the same when the completeness equals to 0, and the rate of SOAP drops sharply as the completeness increases. The results imply that more social information helps the interest-based spam filtering component to infer more personal preferences for spam detection. Bayesian has a higher FN rate than SOAP when the completeness is greater than 0, and the completeness does not have any effect on Bayesian, since it does not consider social factors in spam detection. We can also observe that the relationship between the FN rates of SOAP with different ρ_I and ρ_D remain the same as the completeness increases. This is because the social information completeness does not affect the weights of interest and disinterest keywords in detecting spam.

4.2 False Positive Rate

Fig. 11 shows the average FP rate in Bayesian and SOAP. The figure shows that the FP rate of Bayesian is higher than SOAP with various ρ_I and ρ_D. The improved performance of SOAP results from the social closeness-based spam filtering component and the social (dis)interest-based filtering component. These FP emails are legitimate emails that are from close friends or that match the receivers' personal interests. Because

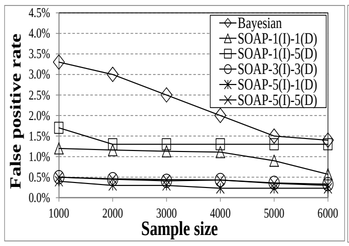


Fig. 12: FP rate vs. # of emails.

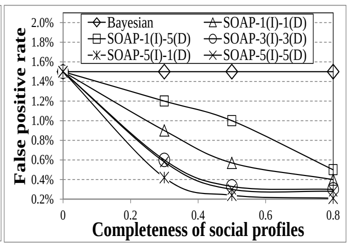


Fig. 13: FP rate vs. profile completeness

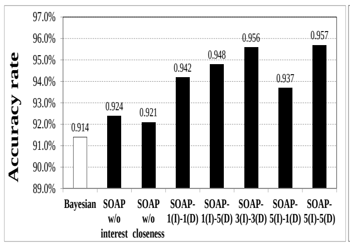


Fig. 14: Accuracy.

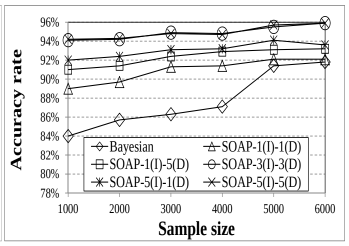


Fig. 15: Accuracy vs. # of emails.

these emails contain general spam keywords, they are regarded as spam by Bayesian, but are correctly classified by SOAP's personalized spam filter. Note that SOAP also mistakenly regards some legitimate emails as spam. This is due to insufficient social information. The figure also shows that SOAP-1(I)-5(D) generates the highest FP rate and SOAP-5(I)-1(D) generates the lowest FP rate among SOAP methods. Although SOAP-1(I)-5(D) has a low FN rate as shown in Fig. 8, since it highly biases on the disinterest keywords, it may mistakenly consider legitimate emails as spam, which leads to high FP rate. Similarly, though SOAP-5(I)-1(D) produces a high FN rate as shown in Fig. 8 by biasing on interest keywords, since it considers an email as legitimate if the email's keywords match a user's interests, it reduces the probability that a legitimate email is detected as spam.

The personalization of SOAP is also illustrated in Fig. 12, which demonstrates the FP rates of SOAP and Bayesian with different sample sizes. The figure shows that SOAP with different ρ_I and ρ_D generates lower FP rates than Bayesian. Also, as the sample size increases, the rate of Bayesian decreases while the rate of SOAP does not greatly change due to the same reason as observed in Fig. 9.

Fig. 13 shows the relationship between the FP rate and the completeness of social information. The figure shows that higher completeness of a person's profile leads to a lower FP rate in SOAP with different ρ_I and ρ_D. This result is consistent with that in Fig. 10. It confirms that social information can help to increase spam detection accuracy but does not affect the relative performance relationship between the FN rates of SOAP with different ρ_I and ρ_D. Thus, SOAP's social interest-based filtering component is effective in reducing the FP rate, while Bayesian still generates a higher FP rate because of the lack of social network consideration.

4.3 Detection Accuracy

We define the accuracy rate as the ratio between the number of successfully classified emails and the number of all received emails. Fig. 14 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 ρ_I and ρ_D. The experimental results show that Bayesian has 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. We can also see that SOAP-3(I)-3(D) and SOAP-5(I)-5(D) produce the highest accuracy. When $\rho_I{=}\rho_D$, SOAP does not bias either interest or disinterest keywords, leading to low FP and FN rates. Since the keyword weights generated by $\rho_I{=}\rho_D{=}\ 3$ and $\rho_I{=}\rho_D{=}5$ based on Equations (14) and (15) are very close, SOAP-3(I)-3(D) and SOAP-5(I)-5(D) have close accuracy rates. Although ρ_I and ρ_D are also equal in SOAP-1(I)-1(D), the keyword weights generated are extremely small, so the (dis)interest filtering in SOAP is not demonstrated. Therefore, its accuracy is much lower than SOAP-3(I)-3(D) and SOAP-5(I)-5(D). As SOAP-1(I)-5(D) and SOAP-5(I)-1(D) have either a high FP rate or a high FN rate, their spam detection accuracies are low.

The results imply that the different components in SOAP play important roles in spam detection. Their synergistic 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 stricter checking, while a larger closeness value leads to looser checking. Since an email from a person who is socially close social is less likely to be spam, both FP and FN emails can be reduced. The interest-based spam filtering component increases the spam detection accuracy by filtering out the emails in the receiver's disinterests and accepting the emails that match the receiver's interests. The experimental results imply that the values of ρ_I and ρ_D in the interest-based spam filtering component need to be equal and large.

Fig. 15 shows 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 that 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 data and time to learn a new spam keyword. Large training samples help to enhance its accuracy.

4.4 Training Time

In this section, 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. Since Fig. 9, Fig. 12 and Fig. 15 can already show the relationship between FN rate, FP rate and Accuracy rate versus sample training, in this section, we focus on accuracy rate which is the most direct measurement of the performance of spam filters. Fig. 16 shows the accuracy rate versus the number of training cycles. We observed that the accuracy of both SOAP and Bayesian increases as the training cycle increases. Unlike SOAP, which exhibits only a 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 social profiles. Instead of requiring much user effort to manually train the Bayesian filter, SOAP uses

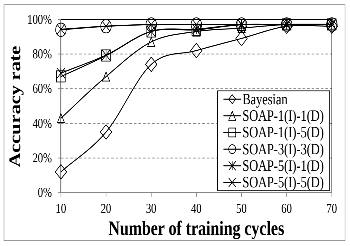


Fig. 16: Training time.

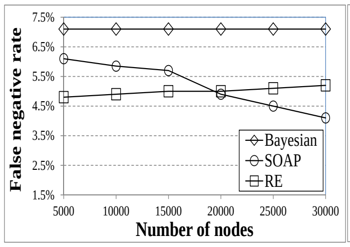


Fig. 17: FN rate vs. network size.

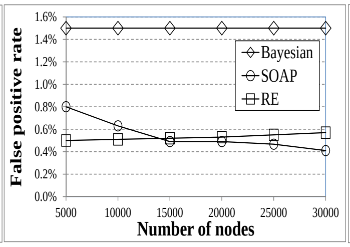


Fig. 18: FP rate vs. network size.

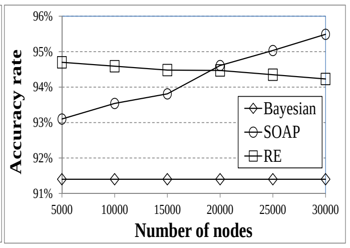


Fig. 19: Accuracy vs. network size.

social information to assist spam detection with less learning. Thus, SOAP need less training time to achieve a high spam detection accuracy rate.

4.5 Resilience to Incompleteness of Social Network

In order to test the scalability of SOAP, we gradually increased the network size from 5000 to 30000 with a 5000 increase in each step. The nodes were randomly selected from the social network node pool in the trace. In the experiment, each of the 10 randomly selected nodes in the network sends an email to another randomly selected node until 5000 emails were sent out. Note that a smaller number of nodes from the Facebook social network less completely reflect the real Facebook social network and make it more likely to generate a disjointed simulated social network, leading to less accurate closeness measurement. Network size here can be considered as the completeness of the constructed social network.

Fig. 17 shows the FN rate versus the network size. As the number of nodes in the network increases, the FN rate of Bayesian stays the same and remains the highest. This is because Bayesian focuses on the spam keywords received by each individual user, so the increased number of total users in the network does not affect its FN rate. We also see that the FN rate of SOAP is initially larger than RE but decreases as the number of users in the network increases. Social closeness between nodes is an important factor for SOAP to calculate the weight of spam keywords. As we directly use the nodes in the Facebook social network in the simulation, the more nodes we put into the simulated social network, the more complete the social network of a certain user is. Then, the calculated closeness is more accurate, resulting in lower FN rate. The reason that the FN rate of RE increases slightly as the network size grows is that as more nodes come into the system, the links of nodes in the real social network are more completely reflected. Therefore, a node can send a spam message to more friends in its social network as the nodes in RE within the same social network are likely to trust each other when the spammer sends the spam into the social network.

Fig. 18 shows the FP rate versus the network size. As the figure illustrates, Bayesian still produces the highest FP rate, which remains constant due to the same reason as in Fig. 17. This is because Bayesian fails to consider the social closeness between persons in spam filtering. We also see that the FP rate of SOAP is initially larger than RE and deceases as the network size increases. This is because more nodes in the system leads to a social network with more complete social relationships. Therefore, social closeness between two nodes can be more accurately captured, leading to a reduced FP rate in SOAP. Although RE also tries to consider social relationships between users in spam filtering, it cannot efficiently detect spam when the spam is sent between socially close nodes, as RE trusts all emails sent from the email receiver's friends and FoF. As there are more nodes in the system, the real social network is more completely reflected. As the real social network has a high clustering coefficient, a node can send spam to more nodes in its connected social network.

Fig. 19 shows the accuracy rate versus the network size. The figure shows that when the number of nodes in the system is small, the accuracy rate of SOAP is less than RE. This is because when the number of nodes in the system is small, the simulated social network cannot completely reflect a node's connections in the real social network. As SOAP filters the spam based on the calculated social closeness, the inaccurate social closeness between nodes may mislead the spam filtering, resulting in a low accuracy rate. In RE, the emails from the senders can only travel through the connected social network. Fewer nodes in the network increases the probability of a disconnected network. Therefore, the spam is less likely to be spread in the social network. Therefore, the accuracy rate of SOAP is comparatively smaller than that of RE. As the number of nodes in the system increases, the accuracy rate of SOAP increases while the accuracy rate of RE decreases. This is because SOAP can retrieve more accurate social information and hence more accurate closeness for combating spam. In RE, the spam can easily be spread to more nodes in the social network. When the number of nodes in the system is more than 20000, the accuracy rate of SOAP is larger than RE for the same reason as in Fig. 17 and Fig. 18.

4.6 Resilience to Impersonation Attacks

In this experiment, we tested the effectiveness of the adaptive trust management component of SOAP in combating impersonation attacks. Recall that in this component, a node decreases another node's trust value if it receives spam from that node. To mimic the behavior of impersonation attacks, we randomly selected 4 nodes to continuously send 10 spam emails to 4 nodes whose closeness values are > 2. The simulation finished after 250 pairs of nodes were selected. We compared SOAP to a completely trained Bayesian filter with a sample size of 6000 to test how fast SOAP can adjust trust values to filter spam from an impersonated node.

Fig. 20 shows the different spam filtering rate versus nodes with different social closeness. The figure illustrate that the accuracy rate of nodes with different social closeness are almost the same. This is because the nodes can automatically adjust their trust value to reach a high spam detection accuracy. The figure also shows that the FN rate increases and the FP rate decreases as the social closeness between nodes increases. This is because the nodes with high closeness loosely check the emails sent between them, which lead to a higher FN rate. In contrast, nodes with low closeness will strictly check the email sent between them, leading to a higher FP rate. Since the relationship between the FN rate, the FP rate and the accuracy rate versus closeness have been shown in Fig. 20,

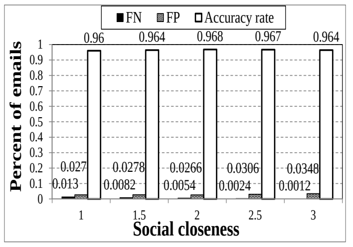


Fig. 20: Percents of emails vs. social closeness.

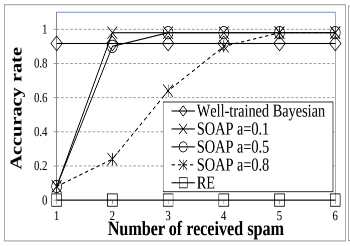


Fig. 21: Impersonation attacks.

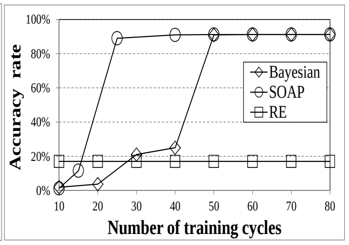


Fig. 22: Trust adjustment.

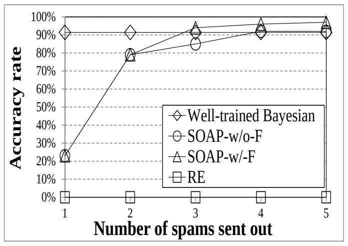


Fig. 23: Effectiveness of friend notification.

in the experiments below, we focus on accuracy rate and shed the lights on how other system parameters affect the system performance.

Fig. 21 shows the average accuracy rate versus the average number of spam emails sent between two pairs of nodes. The figure shows that initially when a receiver receives a spam email from a highly-trustworthy sender, the spam detection accuracy is low. This is because SOAP trusts emails from close people 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 hence the closeness value. Subsequently, it checks the emails from the sender more strictly. This process is demonstrated in Fig. 21. As more spam emails are received, 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 because RE always regards email senders in the whitelist as trustable. Therefore, if spammers impersonate the identities of persons in the whitelist, their spam will be accepted by all nodes in the impersonated persons' FoF social network, leading to a low accuracy rate.

Next, we let nodes send both spam emails and legitimate emails to their randomly selected nodes in the social network within two hops. Bayesian is only trained with partial spam keywords. Fig. 22 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 Fig. 16 that Bayesian needs more training time without using a social network, while SOAP needs less cycles by relying on a social network. The result verifies the effectiveness of the adaptive trust management component in SOAP. RE exhibits poor performance with an accuracy rate of less than 20% for the same reasons as in Fig. 21.

4.7 Effectiveness of Friend Notification

In this experiment, we include the friend notification scheme in SOAP to see how this scheme increases the effectiveness of SOAP in combating impersonation attacks. We use SOAP-w/o-F to denote SOAP without the scheme and use SOAP-w/-F to denote SOAP with the scheme. This experiment was run 50 times, and the average value of the accuracy rates of all nodes is shown in the figures. We randomly selected a node to be a compromised node, which randomly selects 10% of nodes in the social network to send spam.

Fig. 23 shows the average accuracy rate versus the total number of spam messages sent out from the compromised node. We see that the average accuracy rate of Bayesian is constantly high. This is because Bayesian is resilient to the impersonation attack. We also see that the average accuracy rate of RE constantly remains the lowest. This is because when a node in the social network is compromised, the spam can quickly be sent to the node's friends and FoF in the social network, who will always regard those emails as legitimate in RE. The figure also shows that the accuracy rate is the same for SOAP-w/o-F and SOAP-w/-F before the compromised node sends more than two spam emails into the system. After the number of received spam emails become greater than 2, some of the spam receivers' trust values for the spam sender becomes lower than the threshold $T_t = 0.15$. Then, these receivers send out friend notification messages to their friends within two hops in the social network. Later on, as the notified receivers strictly check the emails from the compromised node, the average accuracy rate of SOAP-w/-F is increased and is higher than SOAP-w/o-F.

Fig. 24 shows the accuracy rate of SOAP-w/-F with different number of spam nodes versus different blacklist threshold. We use SOAP-w/F (SN=x) to represent SOAP-w/F system with x spammers. The figure shows that with different blacklist threshold and number of spam nodes in the system, the accuracy rate of SOAP-w/F remains the same. The reason is that since a spammer sends a spam to a large number of users in the system, by using notification messages, the spammer report message about the spammer in each node will be increased quickly to exceed the blacklist threshold. Therefore, SOAP-w/-F can quickly detect the spammer regardless of the blacklist size.

We also test how SOAP prevents malicious users from taking advantage of the notification messages to mis-accuse legitimate users. In the evaluation, a malicious user mis-accuses 10% of randomly chosen users in the system. Fig. 25 shows the percent of nodes being blocked because of the mis-accuse versus blacklist threshold. We use SOAP-w/-F (MN=x) to represent SOAP-w/-F system with x malicious nodes. We can see that as the blacklist threshold increases, the percent of nodes being blocked decreases. This is because to block a node with a large blacklist threshold, it need a large number of nodes to report it as a spammer. Since a spammer normally broadcasts a number of emails to a large number of users in the system, the spammer is very easily to be detected based on the notification mechanism. However, for a legitimate user, it is unlikely that a large number of users report it as spammer if it does not send spam to others. We can see that for the system with 128 malicious nodes, when the blacklist threshold is set to 20, the percent of nodes being blocked is reduced to 0. As the user being accused can also contact email administrators and accuser to release the accuse, the blocking rate can be further reduced in

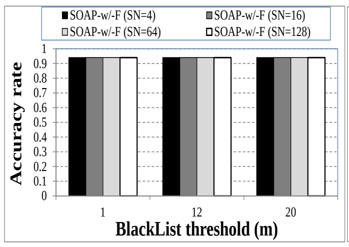


Fig. 24: Accuracy rate vs. blacklist threshold.

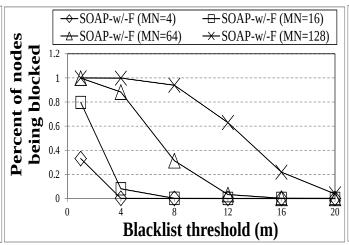


Fig. 25: Percent of nodes being blocked vs. blacklist threshold.

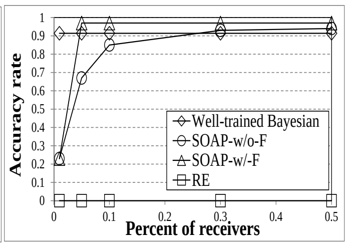


Fig. 26: Accuracy rate vs. percent of receivers.

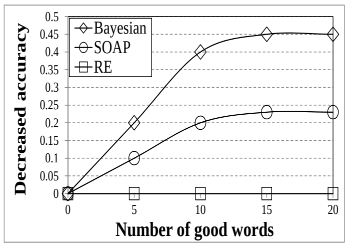


Fig. 27: Resilience to poison attack.

the real environment.

Next, we varied the percentage of nodes in the social network to which the compromised node sends spam. The compromised node sends three spam emails into the networks. Fig. 26 shows the average accuracy rate versus the percent of receivers in the compromised node's social network. We see that as the percent of receivers increases, the average accuracy rates of both SOAP-w/o-F and SOAP-w/-F increase. This is because after a node receives a spam email, the adaptive trust management component adjusts the trust value of the compromised node and strictly checks any received emails from the compromised node. As the percentage of spam receivers increases, the probability of a node receiving more than 2 spam emails increases. Therefore, the average accuracy of the spam filtering increases. We also see that SOAP-w/-F has a much higher increase rate than SOAP-w/o-F. This is because the first node that notices the compromised node will notify its friends about the compromised node. The notification receivers then strictly check the emails from the compromised node even though they have never previously received spam from the compromised node. We further observe that the average accuracy rate of Bayesian is constantly high and the average accuracy rate of RE is constantly small. The reason remains the same as that for Fig. 23.

4.8 Resilience to Poison Attacks

In this experiment, we tested the performance of Bayesian and SOAP under poison attacks. In this attack, extra legitimate words are added into spam to avoid being detected. Fig. 27 shows the decreased accuracy versus the number of legitimate keywords added to each email. 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 words in the emails regardless of other legitimate words. Because spam is always sent out by broadcasting, it is unlikely that a spammer would search for the interests of an individual receiver to poison his/her individual spam filter. SOAP's personalization enables it to avoid poison attacks to a certain degree. In contrast, Bayesian is not personalized and is vulnerable to poison attacks. For emails that do not contain the (dis)interest keywords of the receiver, 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 the email. Thus, its performance does not change as more legitimate keywords are added into spam emails.

5 Prototype Experiments

To further investigate the performance of SOAP in a real-world application, we have implemented prototypes for the Bayesian filter and SOAP in the email client using C#. We invited nine students in the ECE department at Clemson University to use the prototype by connecting their prototype clients to their Gmail accounts and connected them into a social network. The email client periodically fetches new emails from their own Gmail accounts, which are checked by the Bayesian filter and SOAP in the email client. The Bayesian filter and SOAP are trained by the 10000 emails as we used in the simulation in the client installation step.

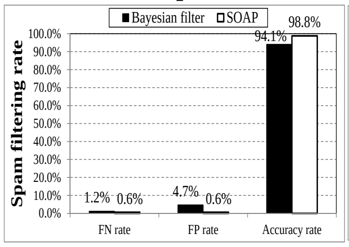


Fig. 28: Spam filtering about the legitimate email.

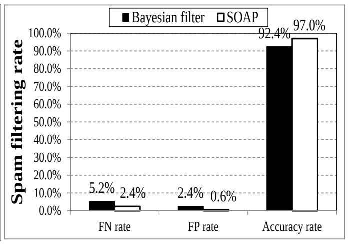


Fig. 29: Spam filtering about spam.

At first, each user sent 20 emails that (s)he thought were legitimate to any other 8 users. Then, all users reported the FN rate, FP rate and accuracy rate in their spam filter based on their own judgement. Fig. 28 shows the FN rate, FP rate and accuracy rate over all users. We see that SOAP has higher accuracy rate, lower FN rate and lower FP rate than the Bayesian filter. The FP emails in our test are emails about the Black Friday Deals. These emails and the recommendation deals from a sender who believes that the receiver would be interested in the deals are regarded as spam by the Bayesian filter at the receiver. Based on the social relationship between users, SOAP loosely checks the emails between friends. Therefore, such FP emails are avoided. On the other hand, some emails that are interested by the senders but are not interested by the receivers are regarded as FN emails by the receivers. SOAP outperforms Bayesian by checking the email containing disinterest keywords strictly.

We then let each user send 20 emails they thought are spam to any other 8 users. Then, all users report the FN rate, FP rate and accuracy rate in their spam filters based on their own judgement. Fig. 29 shows the FN rate, FP rate and accuracy rate over all users. We see that SOAP generates higher accuracy rate, lower FN and lower FP rate than the Bayesian filter in spam detection. Some spam emails were not regarded as spam (e.g., Deals) by the receivers as these emails match their interests. SOAP outperforms Bayesian filter by loosely checking the emails that match the interests of the receivers, resulting in lower FP rate. SOAP detects spam not only by spam keyword as the Bayasian filter, but also based on disinterests of users, leading to lower FP than the Bayesian filter.

6 Conclusion

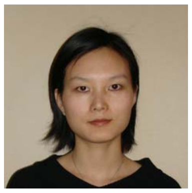
A personalized, attack-resilient, and user-friendly spam filter is needed to effectively combat spammers. However, most current spam filters cannot meet these requirements. In this paper, a SOcial network Aided Personalized and effective spam filter (SOAP) is proposed to meet these requirements. In SOAP, nodes form a distributed overlay by connecting to their social friends. Each node uses SOAP to prevent spam autonomously. SOAP integrates four new components into the Bayesian filter. The social closeness-based spam component prevents spam poison attacks, the social interest-based spam component helps to realize personalized spam filtering, the adaptive trust management component prevents impersonation attacks, and the friend notification scheme leverages the power of a collective of socially close nodes to reinforce SOAP's ability to counter impersonation attacks. Accurate spam filtering results function as input for Bayesian automatic training with reduced user effort to distinguish spam emails. We have also implemented a prototype for SOAP. The results of trace driven and prototype based experiments show that SOAP improves on the performance of the basic Bayesian filter in term of spam detection accuracy and training time. In the future, we will analyze the complexity of the process of the spam detection in SOAP.

Acknowledgment

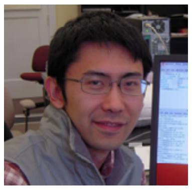
This research was supported in part by U.S. NSF grants CNS-1254006, CNS-1249603, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, Microsoft Research Faculty Fellowship 8300751. An early version of this work was presented in the Proceedings of Infocom'11 [51]

References

[1] Happy spamiversary! http://www.newscientist.com/.
[2] Tracking the high cost of spam. http://www.redcondor.com/.
[3] P. O. Boykin and V. Roychowdhury. Personal Email Networks: An Effective Anti-Spam Tool. *IEEE Computer*, 2004.
[4] Zombie. http://en.wikipedia.org/wiki/Zombie_computer.
[5] C. Binzel and D. Fehr. How Social Distance Affects Trust and Cooperation: Experimental Evidence from A Slum. In *Proc. of ERF*, 2009.
[6] M. Uemura and T. Tabata. Design and Evaluation of a Bayesian-filter-based Image Spam Filtering Method. In *Proc. of ISA*, 2008.
[7] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-mail. In *Proc. of AAAI-98 Workshop on Learning for Text Categorization*, 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] S. J. Delany and P. Cunningham. An Assessment of Case-based Reasoning for Spam Filtering. *Artifical intelligent review*, 2005.
[13] F. Fdez-Riverola, E. Iglesias, F. Diaz, J. R. Mendez, and J. M. Corchado. SpamHunting: An Instance-based Reasoning System for Spam Labeling and Filtering. *Decision Support System*, 2007.
[14] W. Zhao and Z. Zhang. Email Classification Model based on Rough Set Theory. In *Proc. of AMT*, 2005.
[15] L. Cranor and B. LaMacchia. Spams. *ACM Communications*, 1998.
[16] 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.
[17] 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.
[18] SPAMNET. http://www.cloudmark.com.
[19] DNS Real-time Black List. http://www.dnsrbl.com/index.html.
[20] Spamhaus, http://www.spamhaus.org/sbl/index.lasso.
[21] blars.org, http://www.blars.org/.
[22] SpamCop Blocking List, http://spamcop.net/bl.shtml.
[23] O. Boykin and V. Roychowdhury. Personal Email Networks: An Effective Anti-spam Tool. *IEEE Computer*, 2004.
[24] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. Re: Reliable Email. In *Proc. of NSDL*, 2006.
[25] S. Hameed, X. Fu, P. Hui, and N. Sastry. LENS: Leveraging social networking and trust to prevent spam transmission. In *Proc. of FIST*, 2011.
[26] P. Oscar Boykin and Vwani P. Roychowdhury. Leveraging Social Networks to Fight Spam. *IEEE Computer*, 2005.
[27] J. James and J. Hendler. Reputation Network Analysis for Email Filtering. In *Proc. of CEAS*, 2004.
[28] C. Wilson, B. Boe, A. Sala, K. Puttasway, and B. Zhao. User Interactions in Social Networks and Implications. In *Proc. of EuroSys*, 2009.
[29] F. Heider. Attitudes and Cognitive Organization. *J. of Psychology*, 1946.
[30] J. F. Kurose and K. W. Ross. Computer Networking: A Top-Down Approach. *ISBN-10: 0136079679*, 2009.
[31] GFI Software. Why Bayesian Filtering is the Most Effective Antispam Technology. http://www.gfi.com/whitepapers/why-bayesian-filtering.pdf.
[32] D. DeBarr and H. Wechsler. Using Social Network Analysis for Spam Detection. In *Proc. of SBP*, 2010.
[33] H. Lam and D. Yeung. A Learning Approach to Spam Detection based on Social Networks. In *Proc. of CEAS*, 2007.
[34] T. Tran, J. Rowe, and S. F. Wu. Social Email: A Framework and Application for More Socially-Aware Communications. In *Proc. of SocInfo*, 2010.
[35] Spam Arrest, http://www.spamarrest.com/.
[36] Mailblocks, http://www.mail-block.com/.
[37] T. Loder, M. V. Alstyne, and R. Wash. An Economic Answer to Unsolicited Communication. In *Proc. of EC*, 2004.
[38] M. Walfish, J. D. Zamfirescu, H. Balakrishnan, D. Karger, and S. Shenker. Distributed Quota Enforcement for Spam Control. In *Proc. of NSDI*, 2006.
[39] G. Brown, T. Howe, M. Ihbe, A. Prakash, and Kevin Borders. Social Networks and Context-Aware Spam. In *Proc. of CSCW*, 2008.
[40] Y. Niu, Y. M. Wang, H. Chen, M. Ma, and F. Hsu. A Quantitative Study of Forum Spamming Using Context-based Analysis. In *Proc. NDSS*, 2007.
[41] G. Stringhini, C. Kruegel, and G. Vigna. Detecting Spammers on Social Networks. In *Proc. of ACSAC*, 2010.
[42] 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.
[43] M. O. Jackson and J. Wolinsky. A Strategic Model of Social and Economic Networks. *Journal of Economic Theory*, 1996.
[44] S. MILGRAM. The Small World Problem. *Psychology today*, 1967.
[45] L. B. Koralov and Y. G. Sinai. Theory of Probability and Random Processes. *Berlin New York Springer*, 2007.
[46] R. Brachman and H. Levesque. Knowledge Representation and Reasoning. *Morgan Kaufmann*, 2004.
[47] Increasing and decreasing functions. http://www.mathsisfun.com/sets/functions-increasing.html.
[48] O. Boykin and V. Roychowdhury. Personal Email Networks: An Effective Anti-Spam Tool. *Arxiv Archive*, 2004.
[49] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against sybil attacks via social networks. In *Proc. of SIGCOMM*, 2006.
[50] Spam Assassin, http://spamassassin.apache.org/.
[51] Z. Li and H. Shen. SOAP: A Social Network Aided Personalized and Effective Spam Filter to Clean Your E-mail Box. In *Proc. of Infocom*, 2011.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and cloud computing. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.



Ze Li Ze Li received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China in 2007, and the Ph.D. degree in Computer Engineering from Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks. He is currently a data scientist in the MicroStrategy Incorporation.