This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

ALPACAS: A Large-scale Privacy-Aware Collaborative Anti-spam System

Zhenyu Zhong
Secure Computing Corporation
4800 North Point Parkway Suite 300
Alpharetta, GA 30022
ezhong@securecomputing.com

Lakshmish Ramaswamy
Department of Computer Science
The University of Georgia
Athens, GA 30602
laks@cs.uga.edu

Kang Li
Department of Computer Science
The University of Georgia
Athens, GA 30602
kangli@cs.uga.edu

***Abstract*—While the concept of collaboration provides a natural defense against massive spam emails directed at large numbers of recipients, designing effective collaborative anti-spam systems raises several important research challenges. First and foremost, since emails may contain confidential information, any collaborative anti-spam approach has to guarantee strong privacy protection to the participating entities. Second, the continuously evolving nature of spam demands the collaborative techniques to be resilient to various kinds of camouflage attacks. Third, the collaboration has to be lightweight, efficient, and scalable. Towards addressing these challenges, this paper presents ALPACAS - a privacy-aware framework for collaborative spam filtering. In designing the ALPACAS framework, we make two unique contributions. The first is a feature-preserving message transformation technique that is highly resilient against the latest kinds of spam attacks. The second is a privacy-preserving protocol that provides enhanced privacy guarantees to the participating entities. Our experimental results conducted on a real email dataset shows that the proposed framework provides a 10 fold improvement in the false negative rate over the Bayesian-based Bogofilter when faced with one of the recent kinds of spam attacks. Further, the privacy breaches are extremely rare. This demonstrates the strong privacy protection provided by the ALPACAS system.**

I. Introduction

Statistical filtering (especially Bayesian filtering) has long been a popular anti-spam approach, but spam continues to be a serious problem to the Internet society. Recent spam attacks expose strong challenges to the statistical filters, which highlights the need for a new anti-spam approach.

The economics of spam dictates that the spammer has to target several recipients with identical or similar email messages. This makes collaborative spam filtering a natural defense paradigm, wherein a set of email clients share their knowledge about recently received spam emails, provides a highly effective defense against a substantial fraction of spam attacks. Also, knowledge sharing can significantly alleviate the burdens of frequent training stand-alone spam filters.

However, any large-scale collaborative anti-spam approach is faced with a fundamental and important challenge, namely *ensuring the privacy of the emails among untrusted email entities*. Different from the email service providers such as Gmail or Yahoo mail, which utilizes spam/ham classifications from all its users to classify new messages, privacy is a major concern for cross-enterprise collaboration, especially in a large scale. The idea of collaboration implies that the participating users and email servers have to share and exchange information about the emails (including the classification result). But, emails are generally considered as private communication between the senders and the recipients, and they often contain personal and confidential information. Therefore, users and organizations are not comfortable sharing information about their emails until and unless they are assured that no one else (human or machine) would become aware of the actual contents of their emails. This genuine concern for privacy has deterred users and organizations from participating in any large-scale collaborative spam filtering effort.

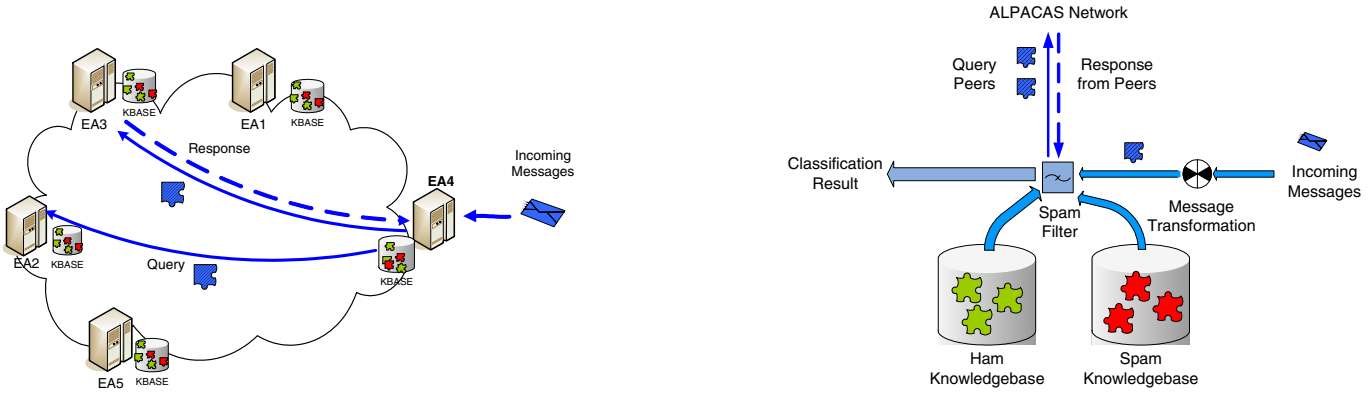
To protect email privacy, digest approach has been proposed in the collaborative anti-spam systems to both provide encryption for the email messages and obtain useful information (*fingerprint*) from spam email. Ideally, the digest calculation has to be a one-way function such that it should be computationally hard to generate the corresponding email message. It should embody the textual features of the email message such that if two emails have similar syntactic structure, then their fingerprints should also be similar. A few distributed spam identification schemes, such as Distributed Checksum Clearinghouse (DCC) [1], Vipul's Razor [2] have different ways to generate fingerprints. However, these systems are not sufficient to handle two security threats: 1) *Privacy breach* as discussed in detail in section II; 2) *Camouflage attacks*, such as character replacement and good-word appendant, make it hard to generate the same email fingerprints for highly similar spam emails.

To simultaneously achieve the conflicting goals of ensuring the privacy of the participating entities and effectively and resiliently harnessing the power of collaboration for countering spam, we design a particular framework and name it *"***A** ***L****arge-scale* **P***rivacy-***A***ware* **C***ollaborative* **A***nti-spam* **S***ystem"* (**ALPACAS**)

In designing the ALPACAS framework, this paper makes two unique contributions: 1) We present a resilient fingerprint generation technique called *"feature-preserving transformation"* that effectively captures the similarity information of the emails into their respective encodings, so that it is possible

This work was partially supported by NSF ITR-CyberTrust program (NSF-CNS-0716357) and Georgia Research Alliance.

978-1-4244-2026-1/08/$25.00



(a) ALPACAS Network

(b) Internal mechanism of $EA4$

Fig. 1: ALPACAS System Overview

to perform fast and accurate similarity comparisons without the actual contents of the emails. Further, this technique also ensures that it is computationally infeasible to reverse-engineer the contents of an email from its encoding. 2) For further enforcing the privacy protection, a privacy-preserving protocol is designed to control the amount of information to be shared among the collaborating entities and the manner in which the sharing is done.

We evaluate the proposed mechanisms through series of experiments on a real email corpus. The results demonstrate that the ALPACAS framework has a comparable overall filtering accuracy to the traditional stand-alone statistical filters. Furthermore, ALPACAS resists various kinds of spam attacks effectively. For good-word attack, ALPACAS has 10 times better false negative rates than both DCC and BogoFilter [3], a well known Bayesian-based spam filter. For character replacement attack, ALPACAS shows a 30 times better false negative rate than DCC and 9 times better false negative rate than BogoFilter. ALPACAS also provides strong privacy protection. The probability of a ham message to be guessed correctly by a remote collaborating peer is well controlled below 0.001.

II. Prior Work

Prior efforts on coordinated real-time spam blocking include distributed checksum clearinghouse (DCC) [1], Vipul's Razor [2], SpamNet [4], P2P spam filtering [5], [6] and SpamWatch [7]. We discuss the drawbacks of the existing collaborative anti-spam schemes using DCC as a representative example.

The DCC system attempts to address the privacy issue by using hash functions. Here, the participating servers do not share the actual emails they have received and classified. Rather they share the emails' *digests*, which are computed through hashing functions such as MD5 over the email body. When an email arrives at a mail server, it queries the DCC system with the message digest. The DCC system replies back with the recent statistics about the digest (such as the number of instances of this digest being reported as spam). DCC suffers from two major drawbacks: First, since hashing schemes like MD5 generate completely different hash values even if the message is altered by a single byte, the DCC scheme is successful only if exactly the same email is received at multiple collaborative servers. DCC develops fuzzy checksums to improve the robustness by selecting parts of the messages based on a predefined dictionary. But, spammers can get around this technique by attaching a few different words to each email.

Second, the DCC scheme does not completely address the privacy issue. A closer examination reveals that the confidentiality of the emails can be compromised during the collaboration process of DCC. Thus, it violates the privacy requirement from the email sender for maintaining the confidentiality of the recipients when he wants to deliver emails to multiple recipients by using 'Bcc:'. In particular, one DCC server can possibly infer who else receives the same email by comparing the querying fuzzy checksum. Assuming DCC uses perfect hash function, consider the scenario wherein an email server EA_i received a ham email M_a. Suppose another email server, say EA_j, receives an identical email later, and sends its fuzzy checksum to EA_i. Since EA_i had seen this email before, it immediately discovers that EA_j too has received the same email M_a. We refer to this type of privacy compromise as *inference-based privacy breaches*.

These two drawbacks, namely vulnerability toward camouflage attacks and potential risk of privacy breaches, highlight the need for better collaborative mechanisms that are not only resilient towards minor differences among messages, but are also robust against inference-based privacy compromises.

III. The ALPACAS Anti-Spam Framework

We present ALPACAS framework to address the design challenges of the collaborative anti-spam system.

- **Challenge 1**: To protect email privacy, it is obvious that the messages have to be encrypted. However, in order for the collaboration to be effective, the encryption mechanism has to satisfy two competing requirements: a) The encryption mechanism has to hide the actual contents for

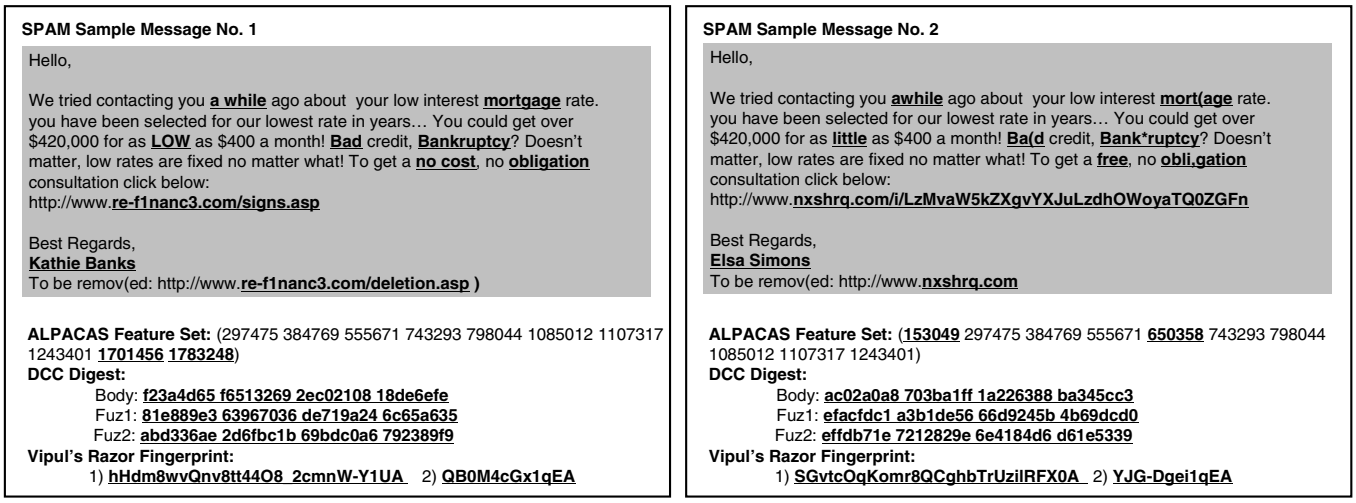


Fig. 2: ALPACAS Feature Sets, DCC and Razor Digests for 2 spam emails (Texts in bold font indicate differences)

privacy protection. b) It should retain important features of the message so that effective similarity comparison can still be performed on the encrypted messages.

- **Challenge 2**: To avoid inference-based privacy breaches, it is necessary to minimize the information revealed during the collaboration process. However, the lesser the information conveyed, the harder it is to perform meaningful similarity comparisons.

Accordingly, the ALPACAS framework includes two unique components, namely *feature-preserving fingerprint* and *privacy-preserving protocol* to address the above challenges respectively. In addition, in the interests of scalability, we design a *DHT-based architecture* for distributing ham/spam information among the collaborating entities.

The ALPACAS framework essentially consists of a set of collaborative anti-spam agents. An email agent can either be an entity that participates in the ALPACAS framework on behalf of an individual end-user, or it may represent an email server having multiple end-users. Without loss of generality, in this paper, we assume that the email agents represent individual end-users. Each email agent of the ALPACAS framework maintains a spam knowledgebase and a ham knowledgebase, containing information about the known spam and ham emails. Figure 1(a) shows the email agent EA_4 querying two other collaborative agents with *partial* information of an incoming message for the purpose of classification. Figure 1(b) illustrates the internal mechanism of each email agent: Upon receiving an email, the respective email agent transforms the message into a feature digest. It then uses *part* of the feature digest to query a few other email agents to check whether they have any information that could be used for classifying the email. Based on the responses from these agents and its local knowledgebase, a simple method to classify email is presented in section III-B.

A. Feature-Preserving Fingerprint

In our approach, the fingerprint of an email is a set of digests that characterize the message content. The set of digests is referred to as the *transformed feature set (TFSet)* of the email. The individual digests are called the *feature elements*. The transformed feature set of a message M_a is represented as $TFSet(M_a)$. In the following sections, we will discuss how to generate $TFSet$ and how to further enforce the privacy preservation.

1) Shingle-based Message Transformation: Our feature-preserving fingerprint technique is based upon the concept of Shingles [8], which has been used in a wide variety of web and Internet data management problems, such as redundancy elimination in web caches and search engines, and template and fragment detection in web pages [9], [10].

Shingles are essentially a set of numbers that act as a fingerprint of a document. Shingles have the unique property that *if two documents vary by a small amount their shingle sets also differ by a small amount*.

Figure 2 presents an example to illustrate the strength of this feature-preserving fingerprint technique. The figure shows two real spam emails that are very similar to each other. The spammers have deliberately mutated one of the emails through word and letter substitutions to obtain the other. The figure shows the *TFsets* of the two emails. For comparison purposes, we also indicate the results of the MD-5 , Vipul's Razor and the DCC transformations on the two emails. For MD-5, Vipul's Razor and DCC, the hash digests of the two emails are totally different from each other whereas the shingle sets of the two emails retain a high degree of similarity that 80% of the *TFsets* of both spam emails are the same.

To generate a *TFset* of a message M, we use a sliding window algorithm, in which a window of some pre-determined length (W) slides through the message. At each step the algorithm computes a Rabin fingerprint [11] of W consecutive tokens (a token could be either a single word or character,

and we use character-based token throughout this paper) that fall within the window. Each fingerprint is in the range (0, $2^K - 1$), where K is a configurable parameter. For a message with X tokens, we obtain a set of $X - W + 1$ fingerprints. Of these, the smallest Y are retained as the (W,Y) *TFset* of M, because using a subset of the fingerprints that represent partial information of M provides more privacy protection than using the entire set of fingerprints. We represent (W, Y) *TFset* of a message M as $TFSet_{(W,Y)}(M)$. The similarity between two messages M_a and M_b can be calculated as $\frac{|TFSet_{(W,Y)}(M_a) \cap TFSet_{(W,Y)}(M_b)|}{|TFSet_{(W,Y)}(M_a) \cup TFSet_{(W,Y)}(M_b)|}$.

In consideration of the privacy preservation, the message transformation uses a Rabin fingerprint algorithm, which is a one-way hash function such that it is computationally infeasible to generate the original email from its $TFset$. However, it is possible to infer a word or a group of words from an individual feature value. The privacy protection requires multiple levels of defenses. In the next subsection, we present our privacy enhancement.

2) Term-level Privacy Preservation: Term-level privacy breach is defined as a feature element uniquely identifies a word or a group of words, and an email agent could infer a phrase or a sentence out from a feature with a reasonable probability if the agent had come across a previous message whose *TFset* contained the same feature value. For example, a term "$99,999" corresponds to a shingle value 16067109. If a recipient of message M_a knows that the encryption of message M_b contains a common shingle value 16067109, he can immediately infer that M_b also contains the term "$99,999".

One approach to mitigate the possibility of inferring a word or a group of words is to shuffle the tokens of the original email and compute *TFset* on the shuffled email. Though this is expected to accomplish term-level privacy compromise, arbitrary and large-scale shuffling can destroy the email features thereby affecting the spam filtering accuracy.

To shuffle the email content in an acceptable manner, our feature-preserving fingerprint scheme adopts a *controlled shuffling* strategy wherein the tokens are shuffled in a predetermined format. Further, the position of a token after shuffling is always within a fixed range of its original position.

Specifically, the controlled shuffling scheme works as follows. The email text is divided into consecutive *chunks* of tokens. Each chunk consists of z consecutive tokens of the email text, where z is a configurable parameter. The tokens in each chunk are shuffled in a pre-determined manner, whereas the ordering of the chunks within the email text remains unaltered. Concretely, each chunk is further divided into y *sub-chunks* (we assume that y is a factor of z). The tokens within an arbitrary chunk CK_h are shuffled such that the token at r^{th} position in the s^{th} sub-chunk (this is the token at the index $(s \times \frac{z}{y}) + r$) in the chunk CK_h) is moved to $(r \times y + s)^{th}$ position within CK_h.

Suppose two messages contain an identical term, by shuffling the term, the rendered text could be different. Thus, it could make the feature element generated from the shuffled text different. We expect this controlled shuffling scheme to reduce the term-level privacy breach. A comprehensive study on this subject will be done in our future work.

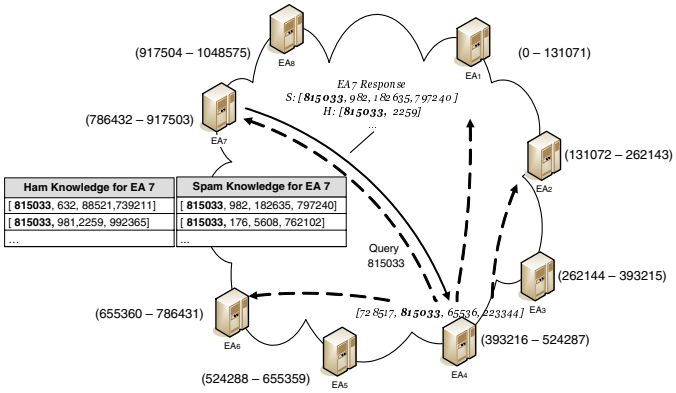


Fig. 3: ALPACAS Protocol: Query and Response

B. Privacy-preserving Collaboration Protocol

Feature-preserving fingerprint is just one level of privacy protection, the amount of information exchanged during collaboration can be further controlled for stronger privacy protection. In particular, we design the collaborative anti-spam system equipped with privacy-aware message exchange protocol based on the following spam/ham dichotomy that *revealing the contents of a spam email does not affect the privacy or confidentiality of the participants, whereas revealing information about a ham email constitutes a privacy breach.*

Our protocol works as follows: When an agent EA_j receives a message M_a, EA_j computes its $TFSet$: $TFSet(M_a)$. It then sends a query message to other email agents in the system to check whether they can provide any information related to M_a. However, instead of sending the entire $TFSet(M_a)$ as a part of the query message to all agents, EA_j sends very small subsets of $TFSet(M_a)$ to a few other email agents (the email agents to which the query is sent is determined on the basis of the underlying structure (please see Section III-C)). The subsets of $TFSet(M_a)$ included in the queries sent to various other email agents need not be the same (our architecture optimizes the communication costs by sending non-overlapping subsets to carefully chosen email agents).

An email agent that receives the query, say EA_k, checks its spam and ham knowledgebases looking for entries that include the feature subset that it has received. A feature set is said to *match* a query message if the set contains all the feature elements included in the query. Observe that there could be any number of entries in both spam and ham knowledgebases matching the partial feature set. For each matching entry in the *spam knowledgebase*, EA_k includes the *complete* transformed feature set of the entry in its response to EA_j. However, for any matching *ham entries*, EA_k sends back a small, *randomly selected part* of the transformed feature set. Figure 3 illustrates our privacy preserving collaboration protocol. In this figure, the agent EA_4 sends a query with the feature element 815033

to EA_7, which responds with a complete feature set of a matching spam email and a partial feature set of a matching ham email.

At the end of the collaboration protocol, EA_j would have received information about any matching ham and spam emails (containing the feature set of the query) that have been received by other members in the collaborative group. For each matching spam email, EA_j receives its complete TFSet. For each matching ham email, EA_j receives a subset of its transformed feature set. EA_j now computes the ratio of $MaxSpamOvlp(M_a)$ to $MaxHamOvlp(M_a)$ and decides whether the M_a is spam or ham. $MaxSpamOvlp$ is the maximum overlaps between the $TFSet$ of the query message and the $TFSets$ of all the matching spam emails, and $MaxHamOvlp$ is similarly defined. In this paper, we use a simple classification strategy that is described in equation 1.

$$Score = \frac{1 + MaxSpamOvlp(M_a) - MaxHamOvlp(M_a)}{2} \quad (1)$$

If the score is greater than a configurable threshold λ, M_a is classified as spam. Otherwise it is classified as ham.

C. System Structure

We design an efficient and scalable structure for the ALPACAS prototype which also minimizes the chances of inference-based privacy breaches. Our prototype structure is based upon the following design principle: *A query should be sent to an email agent only if it has a reasonable chance of containing information about the email that is being verified. Contacting any other email agent not only introduces inefficiencies but also leads to unnecessary exposure of data.*

The proposed prototype structure is based on the distributed hash table (DHT) paradigm [12], [13]. In this DHT-based structure, each email agent is allocated a range of feature element values. An email agent EA_j is responsible for maintaining information about all the emails (received by any email agent in the system) whose TFSet has at least one feature element in the range allocated to it. Specifically, if there are N email agents in the collaborative group, the range $(0, 2^K - 1)$ (recall that the all feature elements lie within this range) is divided into N non-overlapping consecutive regions represented as $\{(MinF_0, MaxF_0), (MinF_1, MaxF_1), \ldots, (MinF_{N-1}, 2^K - 1)\}$, where $(MinF_j, MaxF_j)$ denotes the sub-range allocated to the email agent EA_j. EA_j maintains information about every spam and ham email that has at least one feature element between $MinF_j$ and $MaxF_j$ (inclusive of both end-points). For each such spam email, EA_j stores the entire TFSet in its spam knowledgebase. For ham emails, EA_j stores a subset of the email's $TFSet$. If the feature element value Ft falls within the sub-range allocated to EA_j (i.e., $MinF_j \leq Ft \leq MaxF_j$), then EA_j is called the *rendezvous agent* of Ft. The set of rendezvous agents of all the feature elements of M_a is called M_a's rendezvous agent set. The spam and ham knowledgebases at a rendezvous agent is indexed by the feature element that falls within the agent's sub-range. Figure 3 illustrates a ALPACAS prototype with eight agents and feature elements in the range of (0,1048575).

The presented DHT structure is only for proof of concept. This paper focuses on the feasibility of collaboration with transformed messages and we expect that a more sophisticated and robust P2P structure is applied in a real deployment.

IV. Experiments and Results

In this section, we compare ALPACAS with two popular spam filtering approaches, namely Bayesian filtering and simple hash-based collaborative filtering. We use BogoFilter [3] and DCC as the representatives of these two approaches respectively. As most other Bayesian filters, BogoFilter calculates a score (spamminess) for each message. The message is classified as a spam if its spamminess is greater than or equal to a preset threshold (μ), and vice-versa. On the other hand, the DCC bases its decision on the number of times the email corresponding to a particular hash value have been reported as spam. If this *spam count* of the hash value corresponding to in-coming email exceeds a threshold, the email is classified as spam, and otherwise it is classified as ham.

We conduct a comprehensive study on the accuracy comparison between ALPACAS and BogoFilter for the entire range of the threshold. For other performance measurements, the default threshold for both is set to 0.5. Since DCC is strongly bias to a low false positive rate, we set the DCC threshold to 1, which gives the best false negative rate as shown in Figure 5.

A. Experimental Setup

The datasets used in our experiments are derived from two publicly available email corpus, namely TREC email corpus [14] and the SpamAssassin email corpus [15]. To simulate the collaboration among recipients, we categorize the emails in the TREC corpus, which are the real emails from Enron Corporation according to their target addresses ('To:' and 'cc:' fields) to obtain 67 email sets, each corresponding to the emails received by one individual. Half of each email set including ham and spam are used for training, and the remainder is used for testing. In the experiment, we also assume that each individual can have a pre-classified email corpus (spamAssassin corpus) as the initial knowledgebase. Each individual incrementally feeds the knowledgebase with a fraction of his email set (TREC) categorized for the training purpose. We apply BogoFilter, DCC and ALPACAS on each individual's email set and measure the overall accuracy results.

B. Performance Metrics

We use the standard metrics to measure the spam filtering accuracy. A ham email that is classified as spam by the filtering scheme is termed as a *false positive*. The false positive percentage is defined as the ratio of the number of false positive emails to the total number of actual ham emails in the dataset used during the testing phase. The false negative percentage is analogously defined.

Currently there are no available metrics to measure the privacy of collaborative anti-spam systems. In this paper,

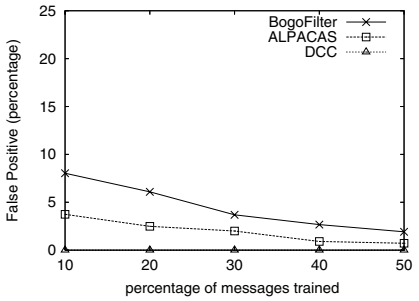


Fig. 4: False Positive Percentages of ALPACAS, BogoFilter and DCC

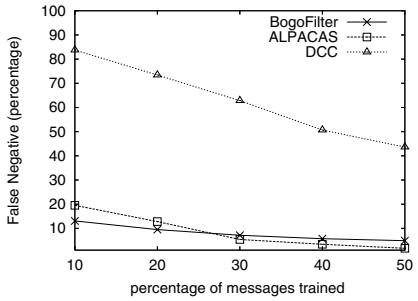


Fig. 5: False Negative Percentages of ALPACAS, BogoFilter and DCC

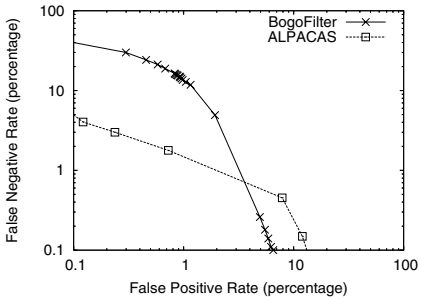


Fig. 6: System Overall Accuracy (DCC is not displayed because its FP is 0)

we first define the *message-level privacy breach percentage* as follows. A ham email M_a is said to have suffered a privacy compromise if an email agent that is not a recipient of M_a discovers its contents. *Message-level privacy breach percentage is defined as the ratio number of test ham messages suffering privacy compromises to the total number of test ham messages.*

The communication overhead of the system is quantified through the *per-test communication cost* metric, which is defined as the total number of messages circulated in the system during the entire experiment.

C. SPAM Filtering Effectiveness

The first set of experiments we study the effectiveness of ALPACAS approach in filtering traditional spam messages (as captured by the testing datasets). Figure 4 shows the false positive percentages of the BogoFilter, the ALPACAS and the DCC schemes when the size of the training set employed by each agent increases from 10% to 50% of the total messages in its email set. Figure 5 indicates the false negative rates for the same experiment.

In general, as we expect, ALPACAS has a strong feature preserving capabilities and demonstrates a better accuracy than BogoFilter when there are enough email resources shared in the network. Figure 4 shows that ALPACAS always performs a better false positive percentage than the BogoFilter. For the false negative percentage shown in Figure 5, ALPACAS is better than BogoFilter after around 27% of the messages in the email sets are employed during the training phase. And ALPACAS shows about 60% lower false negative percentage than that of the BogoFilter when 50% of the messages in the email sets are used for training.

The results also indicates that the essence of the collaboration is knowledge sharing. When the size of the training sets employed at the individual agents is small, ALPACAS doesn't demonstrate a better false negative rate than the BogoFilter. It is also natural that transformed message is less effective than the original message. Furthermore, DCC performs much worse for the false negative percentage than the other two schemes. Note that the false negative percentages of DCC is an order of magnitude higher than our approach.

All the ALPACAS, DCC and Bayesian schemes are threshold-based approaches, so finding the appropriate threshold to achieve both low false positive and false negative rates is the key to the success of these approaches. We obtain results from previous experiment when 50% of the emails in its email set are used during the training phase. We vary the threshold parameters of the two schemes and collect the false positive and false negative percentages. In Figure 6 we plot the results of the experiment with false positive percentages on the X-axis and the false negatives on the Y-axis.

The results show that neither of the approaches outperforms the other at all false positive percentage values. However, ALPACAS approach yield significantly better false negative results than the BogoFilter for the normally preferred false positive range. Generally, users have a much lower tolerance of false positives than false negatives, and anything more than 1% percent false positives is usually considered unacceptable.

In summary, ALPACAS has an overall comparable accuracy to the current approaches such as BogoFilter. It has advantages over BogoFilter when low false positive is preferred. Notice that, even with the same accuracy results, a collaborative filter is often preferred because of its resistance to the camouflage attacks, which is presented in the next subsection.

D. Robustness Against Attacks

In this section we evaluate the robustness of the ALPACAS approach against two common kinds of camouflage attacks, one is *good-word attack* and the other is *character replacement attack*. We compare the results with those of Bayesian and DCC approaches.

In the first experiment of this series, we emulate the good-word attack by appending words that generally appear in ham messages in the test set. The good words are selected randomly from a good word database created from the labeled ham data. We vary the amount of appended words in the range of 0% to 100% of the original emails' word count and we call it *degree of attack*. The experimental setup consists of 67 agents with each agent employing 50% of the messages in its email set during the training phase.

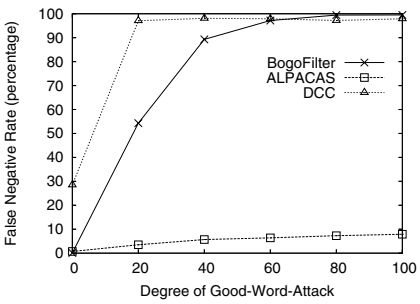


Fig. 7: System Robustness Against Good-Word Attacks

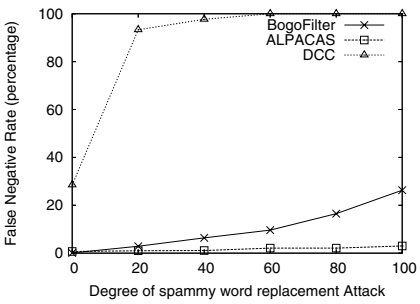


Fig. 8: System Robustness against Character Replacement Attacks

Figure 7 shows the false negative rate of BogoFilter, DCC and the ALPACAS approach at various degrees of attack. False positive results are not presented because they are not affected by the attacks. The false negative percentages of the ALPACAS and BogoFilter are very low when the degree of attack is less than 5%. However, the performance of the BogoFilter degrades drastically as the degree of attack increases, whereas the false positive percentage of the ALPACAS approach increases by very small amounts. For example, when the amount of good words introduced is around 80%, the false negative rate of BogoFilter is close 100%, whereas it is around 7% for the ALPACAS scheme. The performance of DCC is very bad for all its different forms of checksums even at very low degrees of attack. This is because of the nature of its hashing mechanism which maps similar (but not identical) messages into two totally different hash values.

In the second experiment of this series, we study the resilience of the ALPACAS, BogoFilter, and DCC schemes towards another common type of attack, which we call *character replacement attack*. In this attack the spammer replaces a few characters of certain fraction of words that are highly likely to be present in spam emails (henceforth, we refer to these words as "spammy words"). The spammer attempts to reduce the spam weight (weight indicating the probability that the email is a spam) assigned by filters to the email. Emails containing "Vi@gra" instead of "Viagra" are examples of character replacement attacks. In order to emulate this attack, we first create a spam dictionary. For each email in the corpus, we extract the words that appear in the spam dictionary. We then replace a few characters of a certain randomly selected fraction of the words in the spam list. The ratio of the number of changed words to the total number of words in the email that appear in the spam dictionary is called the *degree of attack*.

We then measure the filtering effectiveness of the three anti-spam schemes. The setting is similar to that of the previous experiment. Figure 8 shows the false negative percentage of the three schemes when the percentage of spam words that modified in each email varies from 0% and 100%. As the degree of attack increases, the effectiveness of BogoFilter deteriorates. When 100% of spammy words are modified, the false negative percentage is as high as 27%. In contrast, the false negative percentage of the ALPACAS system is 3% even when 100% of spammy words are modified. The DCC again performs very poorly even at low degrees of attack.

E. Privacy Awareness of ALPACAS Approach

One major design consideration of the ALPACAS approach is preserving the privacy of the emails and their recipients. To measure the privacy breaches, we emulate the following model for privacy compromises. When a rendezvous agent EA_i gets a part of the transformed feature set of an email M_a (either for querying or for publishing), EA_i collects all the ham emails received by it that match the part of the feature set that has been sent to it. In the absence of any further information EA_i selects one of these matching ham emails, say M_b as its guess. In other words, EA_i guesses the contents of the email M_a to be similar to that of M_b. If the guess is correct (the contents of M_a are indeed similar to those M_b) then we conclude that a privacy breach has occurred. We count such privacy breaches to calculate the message-level privacy breach percentage.

The privacy breach also relates to how much information is conveyed during the collaboration. We consider three different query policies in our experiment: 1) query with minimal feature set, 2) query with full feature set, 3) query with partial feature set. To further reduce the content breach possibility, we only share spam knowledge across the collaborative network.

Figure 9 shows the message-level privacy breach percentages of the ALPACAS approach as the number of collaborating agents vary from 100 to 600 for the three query policies. Since the TREC dataset only contains emails received by 67 individuals, we split the email set corresponding to each user into 10 equi-sized trace files. Each of these trace-files drives an email agent. The number of feature elements in the TFSet of each email is 50, and 50% of the emails in each trace is used during the training phase.

The results show that the privacy breaches are very rare for all three modes of the ALPACAS approach. We only show result for the query with 4% partial $TFSet$, because the results for the query with other percentages of $TFSet$ are very close to each other. Further, the privacy breach percentages go down

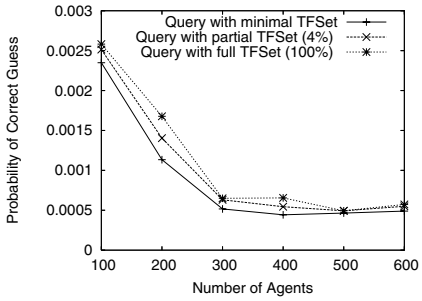


Fig. 9: Privacy Breach in ALPACAS (Varying Number of Agents)

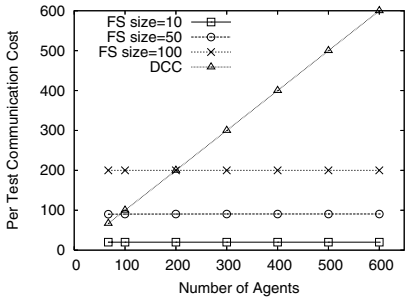


Fig. 10: Communication Overheads of the ALPACAS and the DCC systems

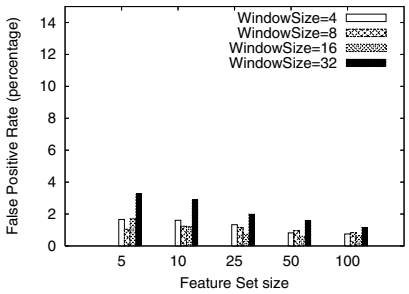


Fig. 11: False Positive of ALPACAS for Various Parameter Setup

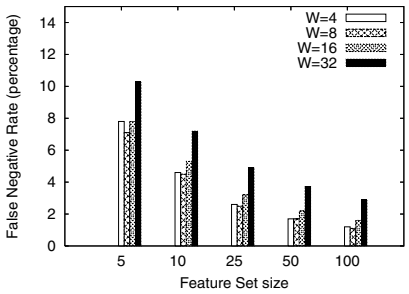


Fig. 12: False Negative of ALPACAS for Various Parameter Setup

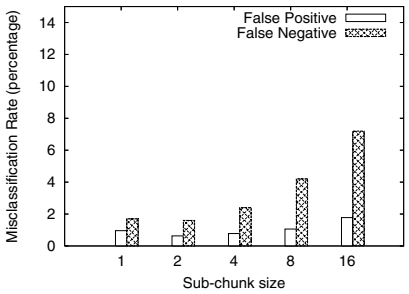


Fig. 13: Effectiveness of Controlled Shuffling Strategy

as the number of agents in the system increases. This can be explained as follows. When the number of email agents in the system increases, the range of DHT values allocated to each email agent decreases. Thus, the probability of a rendezvous agent that has received a similar email in the recent past decreases.

Although with an overall low privacy breach for all three policies, the reduction of privacy breach by using smaller sets is not as significant as we expected. We ascribe this behavior to the small number of email instances in our testing set when compared to the large feature set space. We plan to further study this topic by two means: one is to experiment with various sizes of datasets and feature set spaces; the other is to use feature range in the query rather than the exact feature value, with the hope to further hide the real feature value for the purpose of privacy protection.

F. Communication Overheads of the ALPACAS approach

Communication overhead is a major factor affects the performance of collaborative anti-spam systems. We compare the ALPACAS approach with the replicated DCC approach. Figure 10 indicates the per-test communication cost of both schemes when the number of agents in the system increases from 67 to 600. We conducted experiments with the size of $TFSet$ being set to 10, 50, and 100. The training phase employed 50% of the emails in the trace files.

The graph indicates that the per-test communication costs of the DCC approach increases rapidly with increasing number of email agents, whereas the per-test communication costs of the ALPACAS approach essentially remains constant. This result can be explained as follows. In the DCC system, the spam digest database is replicated at each participating agent. Hence, any update to this database has to be reflected at all replicas, which results in high communication overheads. In the ALPACAS approach, the query and publish messages are sent to only the rendezvous nodes of the corresponding emails. The number of rendezvous nodes is directly dependent upon the cardinality of the transformed feature set being employed. Thus, in this scheme the per-test communication costs depend on the number of feature elements in $TFSet$s and not upon the number of participating agents. The results also show that the ALPACAS approach is highly scalable with respect to number of participating agents.

G. Message Transformation Algorithm Analysis

In this set of experiments, we study the effects of various configuration parameters on the effectiveness of the ALPACAS approach. We first study the effects of feature set size and window size on the accuracy of ALPACAS approach.

Figure 11 and 12 respectively show the false positive and

the false negative percentages of the ALPACAS approach at various settings of the feature set size and the window size parameters. The results show that employing larger number of feature elements yields better classification accuracies. This is because, larger feature sets capture more information about the characteristics of individual emails. We also observe that ALPACAS approach performs best with medium sized windows (windows containing 8-10 characters). This observation can be explained as follows. When the window size is very small, the feature elements correspond to small, commonly occurring sequences of characters. For example, 'agr' can come from either 'vi**agr**a' or '**agr**ee'. Hence, the feature set of an individual email is likely to exhibit high similarities to both ham and spam emails in the knowledgebases, which affects the classification accuracy. On the contrary, when the window size is set to high values, even *similar* emails are likely to have very different feature sets. This is because, when the windows are bigger, each character of the email text appears in several windows. In this scenario, even a few differing characters between two emails can affect the similarity of their feature sets to a considerable extent. Thus, when window sizes are very large, feature set of an individual email is likely to have very little similarity to either the spam or the ham emails in the knowledgebase. This again affects the classification accuracy.

To protect term-level privacy, we propose shuffle method. We assume the entire email is a chunk divided into sub-chunks by a factor to increase the shuffling degree. Figure 13 shows the false positive and false negative rates for different sub-chunk sizes. The results show that when the shuffling degree increases, the accuracy drops. It is because increasing the shuffling degree would break the similarity among emails. However, we believe that with a small degree of shuffle, the ALPACAS approach can still achieve a high classification accuracy, and the attackers would spend much more effort to infer the content from a single shuffled feature element.

V. Discussion

In the current design, we use a simple mechanism for the actual message classification. Approaches like statistical filtering [16] can be utilized in conjunction with the feature preservation transformation scheme. One such strategy would be to apply Bayesian filtering on the feature elements. We believe that sophisticated classification techniques would further improve the filtering accuracy of the ALPACAS approach. Further, our design of the ALPACAS approach assumes that the email agents are stable (i.e., they have low failure rates). Techniques such as replication and finger-table based routing [12] can improve the resilience of the ALPACAS approach towards entries and exits of agents.

The current design of the ALPACAS approach assumes that no participating email agent maliciously uploads erroneous information into the knowledgebases. Further, it is also assumed that no email agent in the ALPACAS approach mounts collaborative inference attacks. For example, if the rendezvous agents of an email exchange the feature elements they have received as a part of the query message, then they have a better chance of correctly guessing the contents of the email. Preventing these types of malicious behaviors by participating agents is a part of our ongoing work.

VI. Conclusion

In this paper, we presented the design and evaluation of ALPACAS, a privacy-aware collaborative spam filtering framework that provides strong privacy guarantees to the participating email recipients. Our system has two novel features: 1) a feature preserving transformation technique encodes the important characteristics of the email into a set hash values such that it is computationally impossible to reverse engineer the original email. 2) a privacy-preserving protocol enables the participating entities to share information about spam/ham messages while protecting them from inference-based privacy breaches. Our initial experiments show that ALPACAS approach is very effective in filtering spam, has high resilience towards various attacks, and it provides strong privacy protection to the participating entities.

References

[1] V. Schryver, "Distributed checksum clearinghouse," http://www.rhyolite.com/anti-spam/dcc/ Last accessed Nov 2, 2005.

[2] Vipul Ved Prakash, "Vipul's Razor Anti Spam System," http://razor.sourceforge.net/.

[3] E. S. Raymond, "Bogofilter: A fast open source bayesian spam filters," http://bogofilter.sourceforge.net/ Last accessed Nov 2, 2005.

[4] Coludmark Corp., "Spamnet anti-spam system," http://www.cloudmark.com/desktop.

[5] A. Gray and M. Haahr, "Personalised, Collaborative Spam Filtering," in *Proceedings of the Second Email and SPAM conference (CEAS)*, 2005.

[6] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "P2p-based collaborative spam detection and filtering," in *The Fourth International Conference on Peer-to-Peer Computing*, August 2004. [Online]. Available: citeseer.ist.psu.edu/721025.html

[7] Feng Zhou, Li Zhuang, "SpamWatch A Peer-to-peer Spam Filtering System," 2003, Available at http://www.cs.berkeley.edu/~zf/spamwatch.

[8] A. Broder, "Some applications of rabins fingerprinting method," in *Sequences II: Methods in Communications, Security, and Computer Science, Springer-Verlag*, 1993, pp. 143–152.

[9] Z. Bar-Yossef and S. Rajagopalan, "Template Detection via Data Mining and its Applications," in *Proceedings of the* 11^{th} *International World Wide Web Conference*, May 2002.

[10] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, "Automatic Detection of Fragments in Dynamically Generated Web Pages," in *Proceedings of the* 13^{th} *World Wide Web Conference*, May 2004.

[11] M. O. Rabin, "Fingerprinting by Random Polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep., 1981.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2001.

[13] L. Ramaswamy, L. Liu, and A. Iyengar, "Cache Clouds: Cooperative Caching of Dynamic Documents in Edge Networks," in *Proceedings of the* 25^{th} *International Conference on Distributed Computing Systems(ICDCS-2005)*, June 2005.

[14] G. V. Cormark and T. Lynam, "Spam Corpus Creation for TREC," in *Proceedings of the Second Email and SPAM conference (CEAS)*, 2005.

[15] M. Sergeant, "Internet level spam detection and spamassassin," in *Proceedings of the 2003 Spam Conference*, January 2003.

[16] K. Li and Z. Zhong, "Fast statistical spam filter by approximate classifications." in *Proceedings of ACM SIGMETRICS 2006/IFIP Performance*, 2006.