Spamdoop: A Privacy-Preserving Big Data Platform for Collaborative Spam Detection

Abdelrahman AlMahmoud, *Member, IEEE*, Ernesto Damiani, *Senior Member, IEEE*, Hadi Otrok, *Senior Member, IEEE*, and Yousof Al-Hammadi, *Member, IEEE*

Abstract—Spam has become the platform of choice used by cyber-criminals to spread malicious payloads such as viruses and trojans. In this paper, we consider the problem of early detection of spam campaigns. Collaborative spam detection techniques can deal with large scale e-mail data contributed by multiple sources; however, they have the well-known problem of requiring disclosure of e-mail content. Distance-preserving hashes are one of the common solutions used for preserving the privacy of e-mail content while enabling message classification for spam detection. However, distance-preserving hashes are not scalable, thus making large-scale collaborative solutions difficult to implement. As a solution, we propose Spamdoop, a Big Data privacy-preserving collaborative spam detection platform built on top of a standard Map Reduce facility. Spamdoop uses a highly parallel encoding technique that enables the detection of spam campaigns in competitive times. We evaluate our system's performance using a huge synthetic spam base and show that our technique performs favorably against the creation and delivery overhead of current spam generation tools.

Index Terms—Spam campaign, privacy-preserving analysis, Map Reduce

1 Introduction

The word spam was originally used to describe unsolicited e-mails sent in bulk. It is hard to define the term spam more accurately. Some argue spam is about the lack of consent on the part of the recipient, while others believe it is about unsolicited e-mail quantity or scale. Other definitions also stressed the commercial nature of spam; for example, the US SPAM act of 2003 established stringent requirements for sending commercial e-mails [1]. Later, spam got closely associated with cyber-crime. Spam e-mails often try to lure the recipient to click on a fake or infected URL that links to a malicious Website (phishing) or downloads a malicious attachment containing a zero-day exploit (spear-phishing).

Spam in all of its forms is still considered as one of the hardest challenges of the connected generation. This in return drove a massive amount of research towards countering it. The effort led to the gradual decline of spamming activities which lead many to believe that spam was no longer a threat. However, recent thorough studies and statistics have concluded otherwise [2]. In fact, about 66.34 percent of all e-mails sent worldwide are considered spam e-mails according to Kaspersky's Spam and Phishing Statistics for the first quarter of 2014 [3] which leads us to conclude that it is still an evolving phenomenon and is still an active cyber threat.

While some spam campaigns advertise products and services, others serve more malicious and sinister purposes such as advertising illegal goods and terrorism which was the main topic of spam in the first quarter of 2016 [4]. Furthermore, evidence has shown that spam serves as a platform for many other cyber-criminal activities. A major case of link between spam and criminal activities goes back to November 11th 2008, when two Internet upstream providers blocked the network access of McColo, a U.S. based web hosting service provider, reporting that the firm's servers were being used for illegal activities. The Washington Post later reported that McColo was used by organized crime as a host for e-mail sales of counterfeit pharmaceuticals, fake security products and child pornography. Following McColo's shutdown, security agencies noticed a decrease of 75 percent percent in unsolicited e-mail sent worldwide.[1] In the last ten years, multiple sources have mentioned spam being used for distributing malicious software, phishing, and delivering other social engineering related attacks [5]

In the 1990s, the average PC user received one or two spam messages a day. Some years ago, the amount of spam grew to an estimated 190 billion messages sent per day [5]. Spammers collect gross worldwide revenues of the order of $200 million per year. Today, the huge quantity of spam generated and distributed on a daily basis makes fighting spam a tall order in terms of processing power and

- *A. AlMahmoud is with Emirates ICT Innovation Center (EBTIC), and Khalifa University of Science and Technology, Abu Dhabi 127788, UAE. E-mail: Abdelrahman.Almahmoud@kustar.ac.ae.*
- *E. Damiani is with Khalifa University of Science and Technology, Abu Dhabi 127788, UAE, and the Università degli Studi di Milano, Milano 20122, Italy. E-mail: Ernesto.Damiani@kustar.ac.ae.*
- *H. Otrok and Y. Al-Hammadi are with Khalifa University of Science and Technology, Abu Dhabi 127788, UAE. E-mail: {Hadi.Otrok, Yousof.Alhammadi}@kustar.ac.ae.*

Manuscript received 25 Oct. 2016; revised 30 May 2017; accepted 10 June 2017. Date of publication 22 June 2017; date of current version 9 Sept. 2019. (Corresponding author: Abdelrahman AlMahmoud.)
Recommended for acceptance by K.-K. R. Choo, M. Conti, and A. Dehghantanha.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TBDATA.2017.2716409

1. See www.spamcop.net/spamgraph.shtml?spamyear for a discussion on how McColo shutdown impacted on the amount of spam worldwide.

bandwidth. Rather than being selective in their campaigns, spammers aim to reach as many users as possible in a short period of time [6]. Many specialized software tools for bulk mail delivery are available, including Shotmail, Batware, Bulk e-mail generator, and others. All these tools support bulk e-mail address collection, the creation of mailing lists and pushing large amounts of e-mails. Recently, spam delivery has become integrated with cyber-crime toolkits such as Blackhole, Whitehole, Cool pack, Zeus and others [7]. Furthermore, spammers have started using botnets to speed up the spread of spam [8]. The suppression of spam involves the need to understand complex patterns of behavior and the capacity to detect emerging types of spam. Consequently, dealing with the massive amount of exchanged e-mails and telling spam messages apart from legitimate e-mails in a reasonable time can be considered as a classic Big Data problem. Many techniques have been proposed to tackle this problem [9], [10], [11]; here, we focus on collaborative spam detection, which tries to detect spam by putting together e-mails collected from several sources. Collaborative detection faces the following challenges:

- Privacy: E-mails can contain private and confidential information about individuals and organizations. Revealing this information to unauthorized parties can have severe consequences.
- Scalability: The flood of exchanged e-mails complicate the detection of spam which makes it more difficult to handle in reasonable time.

Distance-preserving hashing techniques have been put forward for the preservation of the participant's privacy in collaborative spam detection. However, such techniques involve distance computation that limits their scalability [12].

Driven by the need for a truly scalable cooperative spam detection platform we have developed Spamdoop, a platform that can process a huge amount of e-mails in a reasonable time while preserving their content's privacy. More in detail, Spamdoop has the following features:

- Preserves the privacy of e-mails and enables the detection of spam campaigns.
- Applies a two stage obfuscation encoding scheme that utilizes one-way cryptographic hashes.
- Scales easily on distributed Map Reduce platforms.

In order to validate the performance of Spamdoop, we replicated a spammer's behavior by building a spam campaign generator that mimics a commercially available spamming tool. We conducted a number of spam campaigns targeting our developed platform, showing that the encoding used in Spamdoop allows for detecting spam e-mails generated from a single template. The scalability of the Spamdoop platform was evaluated using a huge set of more than 42 million e-mail digests where our solution was able to identify spams within a few minutes. This should be compared with the minimum of 12 hours [13] up to weeks or possibly months [14] that a spam campaign requires from its start until the delivery of its last e-mail. Comparing our detection time to the typical spam campaign duration time, we conclude that Spamdoop can counter large scale spam campaigns in a very competitive time frame.

The remainder of the paper is organized as follows. In Section 2, we present the related work followed by the proposed system which is addressed in Section 3. The experimental results and analysis are considered in Section 4. Finally, we conclude this paper in Section 5.

2 Related Work

Due to the large body of knowledge on spam detection, we will not try to provide a complete survey of the domain. Rather, we focus on work closely related to our problem, i.e., approaches to collaborative privacy aware spam detection relying on distance-preserving hashing to obfuscate e-mail content.

2.1 Distance-Preserving Hashing

The idea of using message digesting or hashing for preserving e-mail confidentiality while filtering spam has been put forward since long [15], [16]. However, cryptographic hashing techniques are easily defeated by simply inserting minor changes into the e-mails. In fact, "hash busters", which are random strings that are inserted into the end of an e-mail, are made specifically for altering the hash value and are used by spammers regularly. Distance-preserving hashes look more attractive as they mask the effect of hash busting to some degree. Indeed, a distance-preserving hash can show how similar two e-mails are to each other in terms of character trigram distribution, identical portions of the e-mail or similarity at binary level. Thus, substantial changes to the e-mail are necessary to obtain a completely new/distant hash output. Among hashes most related to spam detection, Spamsum is a Context Triggered Piece-wise Hashing (CTPH) scheme that was proposed in [17]. It computes a rolling hash over a moving window to detect a context which then triggers a hash function over the chunk of data. Variations of CTPH include Multi-Resolution Similarity Hash (MRSH), which introduces bloom filters to the previous process and chooses different standards for the window and chunk hash functions [18]. And MRSH-v2 which follows the same principal as MRSH but aims to improve efficiency using different set window and chunk hash functions [12], [19]. Those techniques, however, have not been directly applied to spam and still require a distance computation step.

Nilsimsa [20] is a Locality Sensitive Hash (LSH) which aims to detect similarity between e-mails by generating a hash that represents the distribution of trigrams in an e-mail using a sliding window and the combination of trigrams obtained from it. Furthermore, the median is used to determine the final encoding value. Trend Micro Locality Sensitive Hash (TLSH) uses a sliding window over bytes of data and uses quartiles instead of the median to adapt this technique to binary data and images [21]. While techniques that are based on finding similar trigram distributions are used for spam detection [18]. They differ from our proposed solution for the following two main reasons:

1) Sensitivity to changes: Minor changes in the input has an influence on the output which effectively forces the need for distance computations (possibly between data points that are scattered across multiple

machines). Our proposal does not require the need for distance computation by ensuring that the output is not easily affected by minor modifications. This effectively allows for efficient grouping of related digests using a Boolean expression rather than computing the distance between hashes.

2) Regulations and users trust: Data owners choose not to trust distance-preserving hash techniques for personal data sharing because these techniques have not been scrutinized as thoroughly as other well-known cryptographic hashes and have not been properly standardized by regulatory bodies [22]. This means that collaborative platforms that rely on distance-preserving hashes are inapplicable because data owners cannot share their data using them. On the other hand, our proposal applies strong cryptographic hashing which complies with data sharing regulations. [22].

A common property of distance-preserving functions is that they all involve costly distance computation. Fast Forensic Similarity Search (F2S2) [23] argues that trying to find similar files using distance-preserving hashes over large amounts of data requires a lot of time. The authors propose indexing digests based on the occurrences of certain trigrams in them. While this technique speeds up the process of finding similarities and looks suitable for forensics, it means that the platform must know which trigrams occur in the digest, potentially giving away some information about the content that makes it unsuitable for certain cases.

2.2 Collaborative Spam Detection

Spam detection has been a subject of much research [24], [25], [26]. Here, we focus on collaborative techniques where participants collaborate in the detection process rather than simply informing others of their detected spam. Furthermore, since collaborating introduces additional data sources, we are interested in platforms that are designed to process large amounts of data in a short time frame.

Classic machine learning approaches such as Naive Bayesian, Linear discriminant analysis and Support Vector Machines proved the effectiveness of this category of spam detectors [27]. These classifiers were studied and compared thoroughly in the literature [28]. Bayesian techniques are especially popular in spam detection domain thanks to the availability of open source statistical packages such as R [29]. However, early implementations did not provide explicit control of parallelization. Later, research has been conducted toward implementing machine learning algorithms for the purpose of spam detection over Map Reduce [30], [31]. Cosdes [32] is a platform for spam detection that relies on the HTML content/tags of e-mails to perform near duplicate matching. Today, libraries are available that provide scalable parallel implementations of popular machine learning algorithms such as Apache Mahout [33]. However, these approaches do not consider privacy issues. The paper [34] presents a system for performing large scale analysis on spam from different sources. Raw e-mails are collected and processed on a Big Data platorm called OrientDB that uses implicit Map Reduce. However the authors report that their distance computation may take a long time (up to several days). Also, the paper does not deal with preserving the privacy of the e-mails.

2.3 Collaborative Privacy Aware Spam Detection

Other previous works and commercial solutions claim to achieve privacy-aware collaborative spam detection. In particular, the notion of Distributed Checksum Clear-housing (DCC) has been around for a long time [35]. A DCC is a central system in which participants share hashes of their e-mails. The platform then counts the number of times similar e-mails appear and tags suspicious ones as spam. The DCC approach relies on distance-preserving hashing techniques that compute the inter-hash distances to check if the e-mails that generated the hashes are similar. Some seminal works involved P2P networks of mail servers which exchange distance-preserving hashes to securely share information about spam [36]. A P2P architecture was also utilized in [37] to allow e-mail users to query each other for similar digests. More recently, a Large-scale Privacy-Aware Collaborative Anti-spam System (ALPACAS) was proposed in [38] and [39] to prevent inference attacks. These papers put forward the notion of shingles, i.e., distance-preserving fingerprints that are used to identify similar e-mails; however, both rely on computing distance between shingles to identify spam. Furthermore, similarly to distance-preserving hashes, the reversibility and security properties of shingles have not been throughly tested and thus are not recommended for private data sharing.

From the above discussion, it is clear that few of any of the proposed solutions can cope with scalability and privacy requirements of spam detection. Our solution, Spamdoop, is described in the next section.

3 The Spamdoop Platform

Spamdoop is a platform that allows multiple entities to collaborate in early detection of bulk spam campaigns. Our platform also satisfies the privacy requirements of participants. An overview of Spamdoop architecture is shown in Fig. 1, highlighting the three key components of the system:

- The Obfuscator: Encodes e-mail content. The encoding allows for parallel spam processing without compromising the privacy of the original e-mail. The details of this component are presented in Section 3.1.
- The Parallel Classifier: Leverages the properties of the encoding in order to parallelise the process of routing digests corresponding to similar messages to the same bucket. Section 3.2 presents the details of this component.
- The Anomaly Detector: Detects spams based on the size of the buckets and their rate of growth. This component is presented in Section 3.3.

3.1 Obfuscator

The Spamdoop obfuscator is deployed on the participant's side to preserve the privacy of e-mails' content. The obfuscator can be customized to fit the organizational needs where administrators can decide their preferred cryptographic

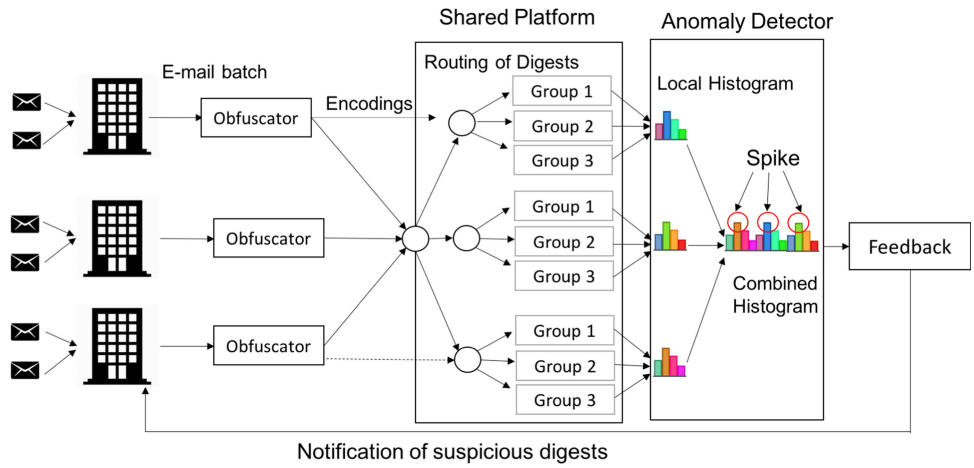

Fig. 1. A high-level illustration of the proposed solution.

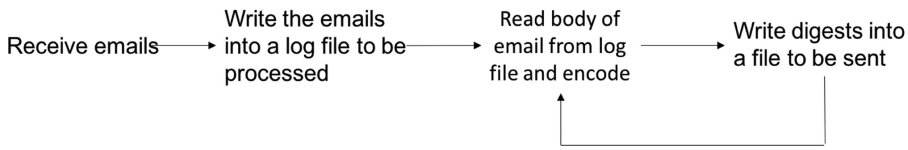

Fig. 2. The obfuscator work flow.

hashing function. Also, the obfuscator source code can be easily verified to make sure it preserves integrity and confidentiality of the data. Fig. 2 shows a possible work-flow. First, e-mails received by a participant are written into an index for batch processing. Second, every t seconds (a period which will be determined and discussed in Section 3.2), the body of the collected e-mails listed in the batch index are extracted and encoded using Spamdoop encoding technique. Third, all digests are written into a file that is sent to our distributed platform. Note that participants can forward their obfuscated e-mails directly to any Spamdoop processing node depending on availability.

3.1.1 The Two Stage Encoder

In this work, we assume that the spammer's strategy is to spread spam as fast as possible by modifying a parent/template to evade detection techniques. Based on the example presented in Fig. 3, we remark that one way of putting together spam e-mails generated from the same template is an encoding technique that leads to identical encoding of different-though similar-e-mails. This rules out standard hash techniques that would lead to close but not identical digests.

We propose a novel many-to-one encoding technique that allows scalable bulk spam detection and classification

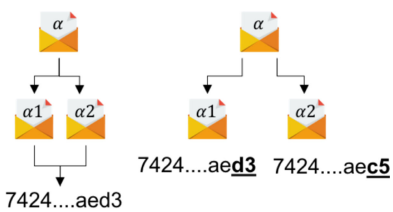

Fig. 3. A spam template would generate two distinct but similar outputs when using traditional fuzzy digest techniques. Our proposal gives the exact same output for two similar yet distinct inputs.

of e-mails over Map Reduce platforms. Our encoding is designed to meet the following requirements:

1) Resilience w.r.t. to modification techniques: Our encoding must be robust against e-mail modification techniques employed by spammers, such as hash busting and customization of content. To address this requirement, we introduce a process which produces a normalized representation of e-mails that is robust w.r.t. to changes. The details of the process are given in stage 1.
2) Privacy-Preservation: Our encoding technique must not be reversible to retrieve the original e-mail. To achieve this, we add a second stage of obfuscation based on one-way hashing.
3) Scalability: Since traditional distance-preserving hashing techniques can require in the order of days to compute the distances between the hashes, we aim to eliminate the need for distance measurement by introducing a preprocessing stage that represents an e-mail as a language model which contains all the trigrams and their corresponding occurrence value. Then substitutes the occurrence value with a channel which is a string that represents a fixed range of numerical values. This allows for two slightly different e-mails to be grouped together using a Boolean expression as will be further explained in Section 3.1.2.

The operation of a two-stage encoding system which meets all of the above requirements is shown in Algorithm 1 which is divided into two main stages. The notation used in this paper is shown in Table 1. The first stage generates a set of trigram and value pairs of an e-mail such that similar e-mails (from the same parent template) have an identical representation. The second stage applies a cryptographic hash function to ensure the privacy and non-reversibility of the e-mail. The details the two stages are as follows:

TABLE 1
Notation Table

K	$\triangleq$	Language model of a character set.
S	$\triangleq$	A Sliding window.
C	$\triangleq$	Channel size.
c	$\triangleq$	Trigram occurance count.
N	$\triangleq$	Set of all channels.
T[K,$count$]	$\triangleq$	An array containing the trigram as key and the number of occurrences as the value.
B	$\triangleq$	Group ID.
E	$\triangleq$	The encoded e-mail.
G	$\triangleq$	The number of groups.
q	$\triangleq$	A score of a digest.
$\alpha\ \beta\ \gamma$	$\triangleq$	Weight parameters that control if the anomaly detector prioritizes historical data or emerging spams.
$\triangle\ \triangle^2$	$\triangleq$	Score
$\mu\ \ \mu'\ \ \mu''$	$\triangleq$	A threshold parameter denoting the top % of spam messages.
L	$\triangleq$	Top occurring digests.
h	$\triangleq$	An encoding's occurrence count.
b	$\triangleq$	A value that determines the closest neighbour.
δ	$\triangleq$	A threshold value to determine close hashes.

Algorithm 1. Encoding Algorithm

Input: E-mail's Body
Output: Digest
Parameter: C, N, T[K, count], S

```
1  while K has Next do
2     T[K, count] = T[K, 0]; \\Initialize to memory the language
      model and set occurrence values to 0
3  end
4  S = first five characters of the e-mail;
5  while S has Next do
6     T[S_t, count++]; \\Increment the value of an occurring
      trigram;
7     S++;
8  end
9  for All K do
10    c = T[t_k]; \\Places count of t_k into c ;
11    i = c mod C;
12    if i > C/2 then
13       b = 1;
14       else b = −1
15    end
16    T[t_k, count ↦ N_i ∪ N_{i+b}];
17 end
18 Hash_algo(T);
19 END
```

Stage 1: Our process of encoding an e-mail starts by initializing a base *language model* including all possible trigrams obtainable using the chosen set of characters (K) and setting their occurrence count to 0 [40]. Moreover, a sliding window (S) is initialized and set to the beginning of the e-mail content. All possible trigram combinations from S are captured and their corresponding values in K are incremented, then S is moved ahead by a character. Trigrams and a sliding window (S) size of five characters were chosen because they proved to be the most suitable for spam detection applications (see [20] and [21]). This process is repeated until S reaches the end of the e-mail where K contains a count of all the occurrences of trigram combinations in the e-mail. The next step involves substituting the values of all the trigram occurrences count in K with two channels that represent a range of occurrences. A simple example of value-to-channel substitution is shown in Fig. 6, where a trigram becomes a member of two channels based on the number of times the trigram occurred in the e-mail. For example, the trigram "aaa" shown in Fig. 6 occurred 4 times, thus it falls in channel 2. Furthermore, channel 1 is the nearest neighbor to the occurrences value 4, thus the trigram belongs to both channels. Similarly, "aab" has a value of 13 placing it in channel 5 and its next closest neighbor channel is 4.[2]

The final output is the entire set of possible trigrams (the language model of the message set [40]) and the channels that correspond to the trigram counts in the processed e-mail.

Stage 2: The process of obtaining the final representation includes adding an appropriate hash function to provide an adequate level of non-reversibility. Since language models are big and unpractical to store and manipulate in their raw form, some compressing is necessary. Here, we assume that the spammer's strategy is to spread e-mails as fast as possible by modifying a parent/template to evade detection techniques. Moreover, even though cryptographic hashes are sensitive to minor changes, the output of the first stage ensures that similar e-mails are identical. SHA and CRC were used in our implementation, but the choices are not restricted to those two. The full two stage process is described in Fig. 5 showing a reduced example of the outputs.

Finally, a file that contains the outputs of our encoder is fed into the Parallel Classifier. Our current implementation feeds the classifier with entire batch of e-mail encodings. However, the data can also be streamed in smaller updates as micro-batches. This design choice affects the way the anomaly detection works and will be discussed in detail in the next section.

3.1.2 Scalability

In a centralized system that uses traditional distance-preserving hashing for spam detection, the distance between each data point and all the other points is computed to find pairs with a lower distance than a determined threshold as $d(X_i, X_j) < \delta$. This process is repeated for each hash value which requires significant computational time for large data sets. However, in distributed systems, there is no efficient method to route related data (close in distance) that are obtained from hashing to the same physical machine, this results in data being scattered across multiple nodes regardless of how related or close the data points are to each other. The issue with this approach to data partitioning is that in order to perform analysis such as spam detection, heavy cross-machine data exchange must occur. An example of this is shown in Fig. 4 Case 1 where a data point from node A is moved to node B so that the distance between it and all the points on node B can be computed, the process of moving

2. The channel sizes were determined according to the top trigram count where the value was rounded up and divided over the channel number. Different dynamic and static channel sizes can be defined but that is outside the scope of this work.

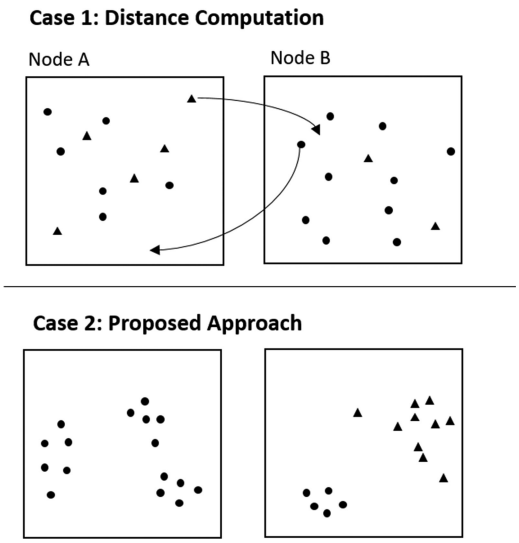

Fig. 4. A comparison between traditional approaches and spamdoop.

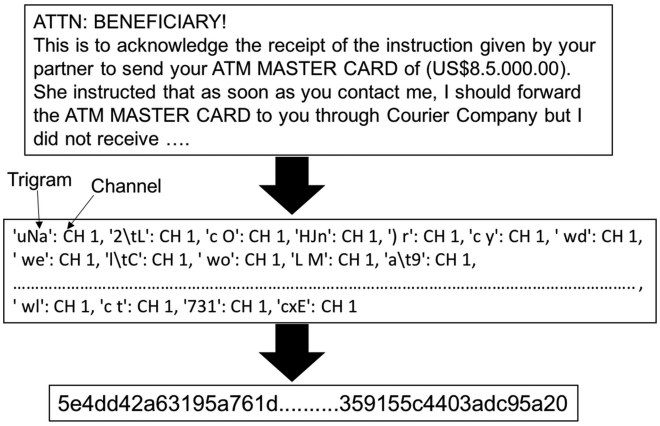

Fig. 5. A spam template would generate two distinct but similar outputs when using traditional fuzzy digest techniques. Our proposal gives the exact same output for two similar yet distinct inputs.

data between nodes is very costly in terms of network communication specially when dealing with huge data sets and a high number of nodes. On the other hand, the design of our approach allows for related data (where the hashes as identical) to be routed and stored on the same nodes in the collection stage as shown in Fig. 4 Case 2. This means that the computations can be performed locally without the need for data exchange between nodes, which reduces network communication latency as well as distance-computation steps.

3.2 Parallel Classifier

This component has two purposes: it routes related digests to the same bucket and groups data so that it is stored on the same node.

The properties of our encoding make this goal simple and inexpensive. The routing of related digests is performed using a simple and efficient method that is applied by [41] which uses hashes of IP addresses to route Netflow information to the same Hadoop nodes. It is shown in Equation (1) where B is the group id, E is the encoded e-mail, and G is the number of groups

$$B = E \ mod \ G. \tag{1}$$

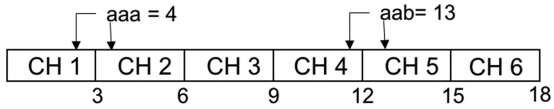

Fig. 6. Multiset membership: In this case, aaa is a member of group CH 1 $\cup$ Ch 2 while aab is a member of Ch 4 $\cup$ CH5.

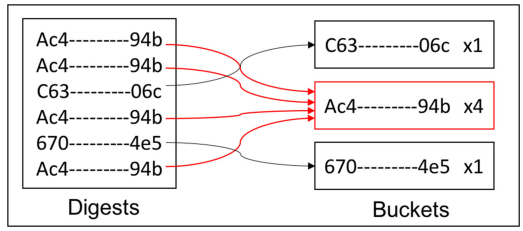

Fig. 7. Grouping of encoded e-mails.

As a result, similar e-mails are grouped together as shown in Fig. 7.

A major challenge when performing parallel analytics is the way input data is distributed across processing nodes. If related data is scattered across multiple nodes, it has to be shuffled around, causing large overheads due to network latency. Our technique groups digests together, and then stores groups on suitable slave nodes. As we shall see, this makes anomaly detection faster, as scores can be computed without shuffling data between nodes.

3.3 Anomaly Detector

Our approach is based on the notion that occurrences and the rate of arrival of spam messages to be very different than the ones of ordinary mail. This is due to spam being generated by automatic tools that try to make the most out of the available bandwidth sending as many messages as possible in the shortest possible time [42]. To take advantage of this difference, we adopted a histogram-based anomaly detection technique that has been used successfully for many other applications, including finding outlying instances in network traffic, or system calls in computers indicating compromised systems. The literature also shows histogram based anomaly detectors to be fast and simple [43].

In our implementation of histogram anomaly detection, we start by constructing the e-mail occurrence *density function*; for each number of occurrences, we compute the number of messages that are present in the batch that many number of times. Fig. 8 shows an example of such a density function. We establish a threshold corresponding to the number of occurrences which is higher than the one in the x percent of the messages. Parameter x is tunable in our system.[3]

At each time t, we compute our score q as shown in Equation (2) where a digest's score is not only determined by its occurrences count in the current batch L_t, but also affected by the number of times it appears in the previous batches by capturing the first and second order differences of digest occurrences $\triangle$ and $\triangle^2$ between time t and $t-1$. Determining the score based on all of the previous parameters allows

3. In our experiments we considered x as 99.9 percent. However, this parameter can be learned, as it depends on the number of participants and many other factors.

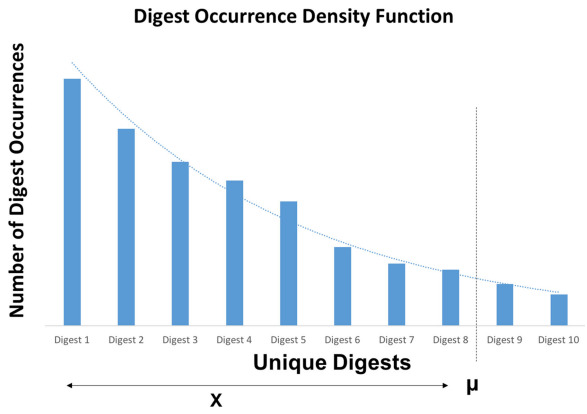

Fig. 8. An example of a density function.

for the anomaly detector to be applicable for both historical data processing and the detection of emerging spams. This is achieved by tuning the values of the weight parameters α, β and γ. For example, a higher value of α might be preferable if historical data is being processed for statistical purposes. On the other hand, if the target is to detect an emerging spam campaign quickly or when using smaller but more frequent batches of digests then a higher value of β is advisable. Top scorers are earmarked as potential spams. The values of L_t for an encoding is computed as shown in Equation (3) where a value is only computed if the digest count surpasses the defined threshold μ shown in Fig. 8. A value is computed by dividing the digest's occurrences count by the count of the highest occurring digest in that batch. Similarly, $\triangle_{Lt}$ is computed as shown in Equation (4) using μ', which is a specific threshold value for the first order difference of digest occurrences which determines if a value is to be calculated for the encoding. This value is calculated using the difference between digest occurrence values in consecutive batches $\triangle_h$ divided by the maximum obtained value $\triangle_{hmax}$. Finally, $\triangle^2_{Lt}$ is computed using Equation (5) taking into account that μ'' is a threshold for the second order difference of digest occurrences.

A well-known trick for the spammer to avoid detection is to introduce variations in the messages in order to decrease the occurrence and rate of any individual version. However, our implementation computes the score based on the density function of our proposed encoding of messages, which brings multiple versions of the message under the same digest (Fig. 7).

$$q = \alpha L_t \ + \ \beta \triangle_{Lt} + \gamma \triangle^2_{Lt} \tag{2}$$

$$L_t = \left\{ \begin{array}{cc} \frac{h}{h_{max}} & if\ h > \mu \\ 0 & otherwise \end{array} \right\} \tag{3}$$

$$\triangle_{Lt} = \left\{ \begin{array}{cc} \frac{\triangle_h}{\triangle_{hmax}} & if\ h > \mu' \\ 0 & otherwise \end{array} \right\} \tag{4}$$

$$\triangle^2_{Lt} = \left\{ \begin{array}{cc} \frac{\triangle^2_h}{\triangle^2_{hmax}} & if\ h > \mu'' \\ 0 & otherwise \end{array} \right\}. \tag{5}$$

While classic histogram-based detection lends itself to parallelization, some adjustments need to be made for

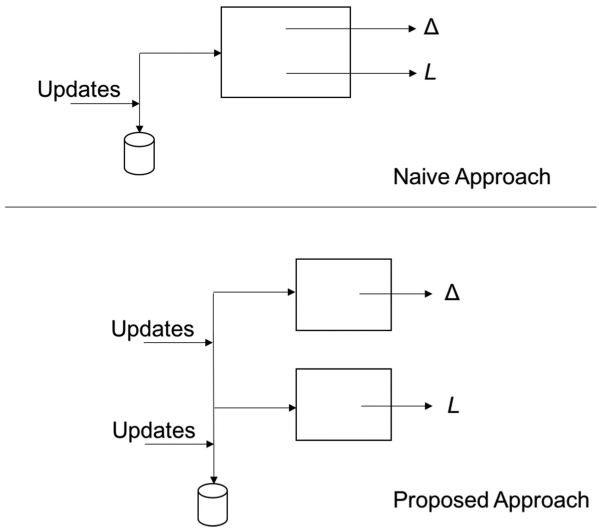

Fig. 9. The difference between a naive approach and estimating $\triangle$ using micro-batches.

achieving an efficient Map Reduce implementation. A scoring equation targeted to Map Reduce implementation is presented in Equation (6). Instead of computing $\triangle$ functions, we simply estimate their values by computing local histograms of micro-batches which are smaller chunks of batch digests that are sent to the system every set period of time t_i. Being timed chunks, they represent the rate of change in the arrival of e-mails. Comparing both anomaly detectors, we found that the second approach has the advantage of allowing two distinct simple mappers to perform $\triangle$ and L computations as shown in Fig. 9.

$$q = \alpha L \ + \ \sum_{i}^{n} f(x). \tag{6}$$

4 Experimental Results and Analysis

In this section, we describe the adversary set-up and the testing environment where we evaluate the functionality and scalability of the proposed encoding.

4.1 Experimental Setup

The spamming business is very profitable, and spammers have perfected their techniques and platforms. We tested our proposal against a spamming platform that includes many features of commercial spamming tools that are being used by spammers worldwide.[4]

First, we give an insight on platforms currently in use by spammers. A high-level overview of the adversary's architecture is shown in Fig. 10. A spammer usually starts by collecting target e-mails to spam by either buying a mailing list or collecting them. This is done using features of spamming software that crawl the web to find addresses (can be risky because of the large number of honeypots). Once a mailing list is created, the details of the campaign are finalized including the selection of the topic of spam and the targeted demographic. The next step is to find a distribution platform by either scanning the network for

4. Such as Atomic Mailer, Dark Mailer and other well-known software.

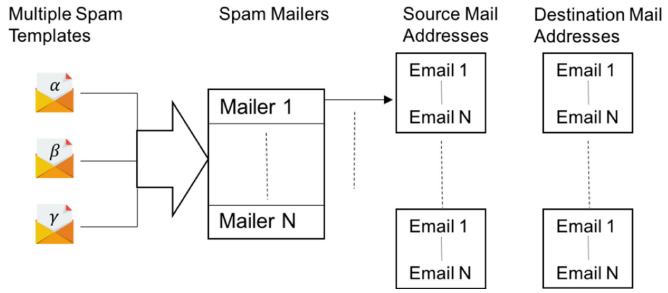

Fig. 10. A spam campain generator.

available non-blacklisted sendmail mailers (with sufficient bandwidth) or by employing a botnet. A spammer starts a campaign by creating a parent spam template which is fed into multiple mail distribution platforms/spamming software which will produce different variations of spams. The spamming software maintains a list of source e-mail addresses to push spams from and a list of destination e-mail addresses to spam. Typically, the software alternates between source and destination e-mail addresses stored in its databases to avoid some countermeasures such as black listing.

Our adversary set-up complies with the above architecture. However, we limited our spamming platform to follow a specific path from Fig. 10. Namely, our spam templates were created manually using examples of spam e-mails from CSDMC2010 SPAM corpus [44]. The mailer consists of scripts that are inspired by commercially available spamming software which supports known spamming strategies such as word substitution and hash busting.

The details of the testing environment and the parameters used are shown in Table 2. The functionality test is divided into two parts. For the first set of functionality tests, five distinct parent templates were created each with a predefined number of characters as shown in the first column

TABLE 2
Experimentation Setup

Blade Server Settings	
CPU	Intel Xeon E5-1620v2 3.7/3.9 GHz
RAM	64 GB DDR3 ECC 1600 MHz
Hard Disk	2x 2 TB SATA3
Hypervisor	XenServer 6.5
Virtual Machine Settings	
Environment	Ubuntu 14.04
Software	Hadoop 1.2.1
CPU	1 virtual CPU
RAM	4 GB
Settings and Variables	
Number of spam templates	5
Sliding Window Size	5
Multigram size	3
Channel Count	4
Channel Size	50 occurrences
Total Number of Digests	43,176,780
Batch 1	10,794,195
Batches 2 to 6	6,476,517 each
Number of Micro-batches	6
Second stage Hashes	CRC32, SHA512

TABLE 3
Experimentation Environment: Each Template Generated 1,500 e-mails

Template Char Count	Point of alteration	Number of Groups				
			Nilsimsa		SSdeep	
		Spamdoop	Threshold out of 128		Threshold out of 100	
			54	103	43	80
345	4	1	1	1	1	3
345	6	1	1	1	1	21
501	3	1	1	1	1	1
501	4	1	1	1	1	1
1310	6	13	1	1	1	1
1310	9	157	1	1	1	5
442	3	2	1	1	1	1
442	5	1	1	1	1	9
2474	3	1	1	1	1	1
2474	5	1	1	1	1	1

of Table 3. Each template was used to create 1,500 unique child spam e-mails. The modification capabilities of the spammer was set to inserting from 3 to 8 word substitutions into the template plus a hash buster as shown in the second column of Table 3. Such setting are compatible with known constraints due to available computation time at the spammer site [45].

The goal of the second set of functionality tests is to measure the rate of false positives in the system. Our encoding scheme was applied on a set of one thousand non-spam e-mails from the CSDMC2010 SPAM corpus. The number of groups created and the size of the groups were observed. The parameters used for our encoding were a sliding window of 5 characters that captured trigram combinations and the number of channels was set to 4.

As for the scale of testing, digests of the generated templates and of other e-mails were added to reach 43,176,780 digests, a value suitable to test the ability of the system to handle a significant fraction of a region's (or a large organization's) e-mail traffic.[5] We employed a run-of-the-mill Map Reduce implementation, Apache Hadoop [46].

4.2 Functionality and Scaling of Tests

The results of the first set of functionality tests are reported Table 3 where the third column shows the results obtained by our encoding and the fourth column are the results obtained using Nilsimsa. The goal of this test is to verify if our encoding is able to group messages generated from the parent templates and place them under the same encoding.[6] Thus, a smaller group number means less variety of encodings generated for a spam campaign which translates into better detection of spam campaigns.

Our technique managed to group the child templates under the same encoding except for the third template which was an e-mail that is repeated three times making the trigram count skewed before the spamming software began

5. According to www.radicati.com, the number of e-mails sent and received per day worldwide in 2015 was around 205 billion..

6. In our tests we use ASCII messages and and do not consider e-mails which use different character sets.

TABLE 4
The Effect of Adding More Working Nodes and Different Second Stage Hashes on the Time Needed to Complete Grouping

Number of slaves	time in seconds	Second Stage
1	878.201316118	SHA512
1	373.109747887	CRC32
2	483.20413303	SHA512
2	221.210319042	CRC32
3	428.81635499	SHA512
3	208.976243973	CRC32
4	407.835330963	SHA512
4	217.356626034	CRC32

processing it. Thus, a targeted attack aiming to inject a significant number of a trigram is the most efficient way to force an encoding change. However, this process is computationally expensive and would makes the spam e-mail very unnatural and suspicious. Such attacks can be controlled using more intelligent channel sizes which is out of the scope of this work. Since Nilsimsa and ssdeep do not readily group similar e-mails under the same hash, an extra step had to be performed. Each hash value was compared with all of the other hashes where scores over 56 (recommended by the original paper) and 103 were considered a match for Nilsimsa, while scores over 54 and 80 were considered a match for ssdeep. Nilsimsa scored perfectly in our tests using both thresholds since it was originally designed for such applications. SSdeep also scored perfectly using the lower threshold with minor misses on the higher one.

To evaluate if Spamdoop can scale up, we devised a test in which the performance of the system is measured based on the time spent by working nodes to group 43,176,780 digests. The effects of adding working nodes and the choice of second stage hashing on the time are also reported.

Initially, the digests were set to be grouped by a single virtual machine, then we gradually increased the number of working machines. The virtual machine settings are shown in Table 2.

We start by measuring the processing time of a single batch containing all of the digests in Table 4. A single node is able to group the entire data set in 878.20 seconds in the case SHA512 being used as the second stage encoding. However, the run-time can be reduced by 373.109 seconds if CRC32 is used as the final representation because it is more efficient to manipulate shorter digests and the file size is much smaller.

To measure the performance of computing $\triangle$ as discussed in Section 3.3, the digest file was partitioned to six batches where the grouping time of these batches are measured. The first batch contains 10,794,195 digests and the remaining five contained 6,476,517 digests each. The first batch is the largest because we assume that the system was not previously operational, thus the amount of collected e-mails is larger than other batches where the data is regularly collected after time t.

The detailed run-time of all batches are demonstrated in Tables 5 and 6. In a full deployment, participants send their contributions every set of time (t) and thus to compute the total run-time of processing all the batches we do not include the wait time. Tables 5 and 6 shows that the processing time of the batches are considerably longer than that of the single batch approach. In fact, adding the fourth node had a negative effect on performance and switching from SHA512 to CRC32 yielded much smaller improvements. That is due to the initialization time of Hadoop and the assignment of Map Reduce nodes and jobs.[7]

Comparing our run-time to the life time of a spam campaign, which usually ranges from 12 hours to 8 days [13], we find that our platform is capable of processing a large amount of e-mails in a much shorter time (few minutes), blocking campaigns well before they conclude.

The results of the second set of functionality tests are shown in Fig. 11. We find that the median group size is 1 and the largest group contains 91 e-mails. By inspecting the e-mails and the groups they were mapped to, we found that the larger groups were heterogeneous. We found that a sample of the e-mails in the larger group contains hyperlinks, which led us to believe that special symbols and hyperlinks contribute to routing e-mails to similar groups. However, the larger groups are still too small to be considered as a spam campaign (considering the scale of our tests and the thresholds used in the previous tests) and thus are not flagged as spam (nor considered as false positives). Finally, we would like to remark that the obtained group sizes can be either decreased or increased using a different number of channels or different channel sizes for each trigram.

4.3 Privacy and Compression

In this section we discuss the reversibility of our encoding. Accuracy, design choices and targeted attacks on the encoding are also discussed.

Given that a common attacker model used in cloud platforms is a passive adversary, we remark that our proposed encoding ensures that a honest-but-curious platform hosting the Spamdoop service will not able to read any data post second stage hashing. Instead, an active adversary will try to compute an e-mail given the output of the two stages of encoding. That means that both stages of encoding would have to be reversed.

Let us start analyzing reversibility from the second stage. By using a cryptographic hash function such as SHA, Spamdoop encoding inherits its non-reversibility properties, meaning that it is computationally unfeasible to revert the output of stage two back to stage one. However, in some cases where security is not an issue and performance is prioritized, a short representation such as CRC, which is easily reversible, could be favored. While the first stage of obfuscation offer no privacy guarantees, it offers basic anonymization of e-mails because of three factors:

- The output contains the entire language model regardless of the trigram count where the appearance of the trigram is masked.
- The output of the first stage replaces trigram counts with channels that represent a range of possible occurrences.

7. A wide variety of Hadoop optimizations are possible to further speed up various aspects of Spamdoop performance, but Hadoop configuration details are beyond the scope of this paper.

TABLE 5
Individual Batch Run-Times in Seconds Using CRC32

Stage 2: CRC	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6	Total
1 Slave	261.96301	255.9453	255.94553	255.93754	255.92305	255.90723	1541.622
2 Slaves	168.96397	156.8959	156.83579	156.89052	156.85789	155.87952	952.3236
3 Slaves	150.87205	138.8188	133.94329	135.85188	136.84365	137.84446	834.1741
4 Slaves	147.97175	136.9069	134.93146	134.89545	133.02509	135.13934	822.87

TABLE 6
Individual Batch Run-Times in Seconds Using SHA512

Stage 2: SHA512	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5	Batch 6	Total
1 Slave	368.3527	327.1311	327.10202	318.03333	332.07158	321.01609	1993.707
2 Slaves	206.9189	177.8994	180.90103	176.82687	181.91243	188.91305	1113.372
3 Slaves	174.9691	149.8416	153.03948	151.01346	149.9121	152.8793	931.6551
4 Slaves	183.9834	147.03694	158.9786	154.38748	151.9147	155.9728	952.27392

- The position of the occurring trigrams are masked. The output does not show the order in which the trigrams appeared in the e-mail.

Having neither an exact count of the trigrams nor the set of occurring trigrams (we recall the entire language model is included and set to channel one) makes it nowhere near trivial to reverse the first stage to the original e-mail. However, we stress that the first stage's output is not suitable for data sharing because it does not offer any privacy guarantees. Furthermore, the representation obtained from the first stage is very large, thus the cryptographic hash in second stage provides both a more compact and more secure output.

Given the output of the first stage, an attacker can try to guess if a word is present in an e-mail by looking for improbable trigrams. If an unlikely or rarely occurring trigram (in the English language) is found in the first stage output, then the choice of words generating that trigram can be narrowed down. However, would find difficulty because the entire language model (all possible trigrams) are included in the representations and are set to channel one, thus no educated guess can be made on which trigram is in the e-mail unless it occurs often enough to put it into the next channel which can be unlikely for rarely occurring trigrams. On the other hand, an attacker can guess with a probability if a trigram does not occur at all or often enough. However, that information is seldom useful to attackers.

A more sophisticated attack on our encoding is to specifically alter the messages so that a certain trigram is increased

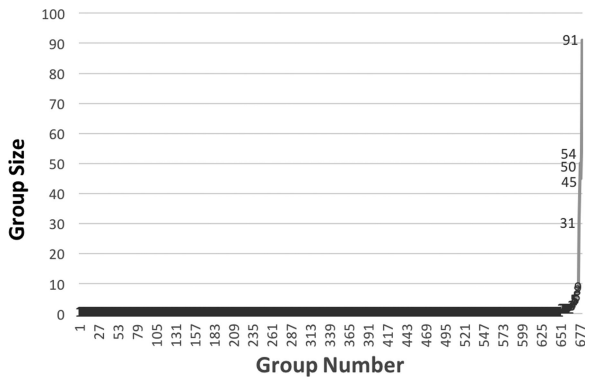

Fig. 11. Second functionality test results showing the number of groups created and their sizes.

or decreased enough to change the output of the first stage, thus changing the final hash output. To achieve this, the spammer has to inject a specific trigram enough to move it from its channel to the neighboring channel and half of the neighboring channel (Fig. 6). by doing this, the similarity between both e-mails cannot be detected because of the effects of the cryptographic hash. However, this would substantially increase the time needed to generate the spam messages from the template because the attacker is also bound by computational constraints and needs to push a large number of e-mails quickly.

One of the most well documented trade-offs of any spam detection platform regards achieving performance, accuracy and privacy at the same time. This remains true in our proposal. Indeed, we sacrifice a small degree of accuracy for performance by introducing the first stage. For the second stage, however, the user can choose to prioritize performance over privacy depending on the chosen hash. Due to the progress of in-memory computation for Map Reduce, populating the entire possible trigrams combinations of a character set is a computationally inexpensive.

Finally, we would like to discuss compressing an e-mail using our digest. The language model created in the first stage contains every possible trigram combination. However, the value associated with a trigram can range from a single digit to multiple digits. Since a cryptographic hash function with a fixed output length is used for our final representation, the large size of the language model is always compressed to a fixed size regardless of the trigram count. However, the choice of final representation is up to the developer to decide.

5 Conclusion

Spamdoop aims to facilitate collaborative spam detection by taking into account the privacy of all the participants and the scale of collective data. A major innovation of our stage encoding based is representing and then hashing the entire language model. This allows us to group spam e-mails generated from the same parent template into one bucket. Our encoding scales well on Map Reduce platforms, outperforming distance-preserving hashing techniques [23], [34]. Also, an efficient bucketing technique was deployed to

simplify grouping of digests. The histogram based anomaly detection we used to distinguish between ham and spam readily lends itself to Hadoop implementation; however, we remark that our framework is agnostic with respect to the specific anomaly detection technique. We used an adversary platform mimicking real spamming platforms to test the effectiveness of our encoding and the performance of our parallel classifier against digests of more than 43 million synthetic e-mails.

For a single large batch, our tests showed that it is possible to reduce the grouping time of digests by 53 percent when distributing the work across four nodes. Furthermore, the computation time was further decreased by 57 percent when using CRC32 on a single working node and 46 percent in the case of four nodes compared to SHA512. On the other hand, processing the six batches was 52 percent faster on four nodes compared to only one node and 13 percent when switching to CRC32. We believe these results to show clearly that Big data spam detection technique are ripe for in-production deployment.

The spam detection mechanism currently uses the e-mail body only. However, the first stage of obfuscation can be applied to trigram techniques such as [21] which can be used for grouping binary data such as images. Instead of using the entire language base to produce the first stage output, one can intelligently remove unlikely trigrams if the spammer has no way of knowing which. Finally, different strategies for choosing channel sizes can be employed where channels can be of different sizes.

Acknowledgments

We would like to thank the Japanese-French Laboratory for Informatics for providing the source code for their project which gave us the initial inspiration for this work. This work has been partly funded by the European Commission under the H2020 TOREADOR project (contract n. 688797).

References

[1] S. Heron, "Technologies for spam detection," *Netw. Secur.*, vol. 2009, pp. 11–15, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485809700078

[2] D. Wang, D. Irani, and C. Pu, "A study on evolution of email spam over fifteen years," in *Proc. 9th Int. Conf. Collaborative Comput.: Netw. Appl. Worksharing*, 2013, pp. 1–10.

[3] Kaspersky Lab, "Spam and phishing statistics report Q1–2014," [Online]. Available: https://usa.kaspersky.com/internet-security-center/threats/spam-statistics-report-q1-2014/, Accessed on: Jun. 02, 2016.

[4] Kaspersky Lab, "Spam and phishing statistics for 2016, " [Online]. Available: https://www.kaspersky.com/about/press-releases/2016_kaspersky-lab-reports-significant-increase-in-malicious-spam-emails-in-q1-2016, Accessed on: Jun. 02, 2016.

[5] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley, "2016 Internet security threat report," *Symantec*, vol. 21, pp. 27–36, 2016.

[6] G. Cormack, "Email spam filtering: A systematic review," *Found. Trends Inf. Retrieval*, vol. 1, pp. 335–455, 2007.

[7] A. Khraisat, A. Alazab, M. Hobbs, J. Abawajy, and A. Azab, "Trends in crime toolkit development," in *Network Security Technologies: Design and Applications: Design and Applications*. Hershey, PA, USA: IGI Global, ch. 2, 2014, pp. 28–43.

[8] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns," in *USENIX Workshop Large-Scale Exploits Emergent Threats*, 2011, Art. no. 4.

[9] M. Crawford, T. Khoshgoftaar, J. Prusa, A. N. Richter, and H. Al Najada, "Survey of review spam detection using machine learning techniques," *J. Big Data*, vol. 2, 2015, Art. no. 23.

[10] M. Sheikhalishahi, A. Saracino, M. Mejri, N. Tawbi, and F. Martinelli, "Fast and effective clustering of spam emails based on structural similarity," in *Proc. Int. Symp. Found. Prac. Secur.*, Berlin, Germany: Springer, 2015, pp. 195–211.

[11] J. Francois, S. Wang, W. Bronzi, R. State, and T. Engel, "BotCloud: Detecting botnets using MapReduce," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, 2011, pp. 1–6.

[12] M. V. Gayoso, Á. F. Hernández, and E. L. Hernández, "State of the art in similarity preserving hashing functions," in *Proc. Int. Conf. Secur. Manage.*, 2014, pp. 1–7.

[13] L. Zhuang, J. Dunagan, D. Simon, H. Wang, I. Osipkov, and D. Tygar, "Characterizing botnets from email spam records," in *Proc. USENIX Workshop Large-Scale Exploits Emergent Threats*, 2008, pp. 1–9.

[14] J. Chen, R. Fontugne, A. Kato, and K. Fukuda, "Clustering spam campaigns with fuzzy hashing," in *Proc. Asian Internet Eng. Conf.*, 2014, Art. no. 66.

[15] Vipuls razor, [Online]. Available: http://razor.sourceforge.net, Accessed on : Feb. 02, 2017.

[16] V. Prakash and A. O'Donnell, "Fighting spam with reputation systems," *Queue*, vol. 3, pp. 36–41, 2005.

[17] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digit. Investigation*, vol. 3, pp. 91–97, 2006.

[18] V. Roussev, G. Richard, and L. Marziale, "Multi-resolution similarity hashing," *Digit. Investigation*, vol. 4, pp. 105–113, 2007.

[19] F. Breitinger and H. Baier, "Similarity preserving hashing: Eligible properties and a new algorithm MRSH-v2," in *Proc. Int. Conf. Digit. Forensics Cyber Crime*, 2012, pp. 167–182.

[20] E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati, "An open digest-based technique for spam detection," in *Proc. Int. Workshop Secur. Parallel Distrib. Syst.*, 2004, pp. 559–564.

[21] J. Oliver, C. Cheng, and Y. Chen, "TLSH-a locality sensitive hash," in *Proc. 4th Cybercrime Trustworthy Comput. Workshop*, 2013, pp. 7–13.

[22] N. Smart, V. Rijmen, B. Warinschi, G. Watson, K. Patterson, and M. Stam, "Algorithms key sizes and parameter report-2013 recommendations," *Eur. Union Agency Netw. Inf. Secur. (ENISA)*, Enisa Report Version 1.0, 2013.

[23] C. Winter, M. Schneider, and Y. Yannikos, "F2S2: Fast forensic similarity search through indexing piecewise hash signatures," *Digit. Investigation*, vol. 10, pp. 361–371, 2013.

[24] N. Spirin and J. Han, "Survey on web spam detection: Principles and algorithms," *ACM SIGKDD Explorations Newslett.*, vol. 13, pp. 50–64, 2012.

[25] W. Shi and M. Xie, "A reputation-based collaborative approach for spam filtering," in *Proc. Conf. Parallel Distrib. Comput. Syst.*, 2013, pp. 220–227.

[26] M. Sirivianos, K. Kim, and X. Yang, "SocialFilter: Introducing social trust to collaborative spam mitigation," in *Proc. IEEE INFOCOM*, 2011, pp. 2300–2308.

[27] G. Caruana and M. Li, "A survey of emerging approaches to spam filtering," *ACM Comput. Surveys*, vol. 44, pp. 1–27, 2012.

[28] K. Kumar, G. Poonkuzhali, and P. Sudhakar, "Comparative study on email spam classifier using data mining techniques," in *Proc. Int. MultiConf. Eng. Comput. Scientists*, 2012, pp. 14–16.

[29] D. Conway and J. White, *Machine Learning for Hackers*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.

[30] G. Caruana, M. Li, and M. Qi, "A MapReduce based parallel SVM for large scale spam filtering," in *Proc. Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2011, pp. 2659–2662.

[31] G. Caruana, "MapReduce based RDF assisted distributed SVM for high throughput spam filtering," Ph.D. dissertation, School Eng. Des., Brunel University, London, U.K., 2013.

[32] C. Tseng, P. Sung, and M. Chen, "Cosdes: A collaborative spam detection system with a novel e-mail abstraction scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 5, pp. 669–682, May 2011.

[33] Apache Mahout: Scalable machine learning and data mining. [Online]. Available: https://mahout.apache.org/, Accessed on: Jun. 02, 2016.

[34] S. Dinh, T. Azeb, F. Fortin, D. Mouheb, and M. Debbabi, "Spam campaign detection, analysis, and investigation," *Digit. Investigation*, vol. 12, pp. 12–21, 2015.

[35] Distributed checksum clearinghouses. [Online]. Available: https://www.dcc-servers.net/dcc/, Accessed on: Jun. 02, 2016.
[36] E. Damiani, S. di Vimercati, S. Paraboschi, and P. Samarati, "P2P-based collaborative spam detection and filtering," in *Proc. 4th Int. Conf. Peer-to-Peer Comput.*, 2004, pp. 176–183.
[37] J. Kong, B. Rezaei, N. Sarshar, V. Roychowdhury, and O. Boykin, "Collaborative spam filtering using e-mail networks," *IEEE Comput.*, vol. 39, no. 8, pp. 67–73, Aug. 2006.
[38] Z. Zhong, L. Ramaswamy, and K. Li, "ALPACAS: A large-scale privacy-aware collaborative anti-spam system," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008, pp. 556–564.
[39] K. Li, Z. Zhong, and L. Ramaswamy, "Privacy-aware collaborative spam filtering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, pp. 725–739, 2009.
[40] K. Church, T. Hart, and J. Gao, "Compressing trigram language models with golomb coding," in *Proc. Joint Conf. Empirical Methods Natural Language Process. Comput. Natural Language Learn.*, 2007, pp. 199–207.
[41] R. Fontugne, J. Mazel, and K. Fukuda, "Hashdoop: A MapReduce framework for network anomaly detection," in *Proc. IEEE Comput. Commun. Workshops*, 2014, pp. 494–499.
[42] J. Iedemska, G. Stringhini, R. Kemmerer, C. Kruegel, and G. Vigna, "The tricks of the trade: What makes spam campaigns successful?" in *Proc. IEEE Secur. Privacy Workshops*, 2014, pp. 77–83.
[43] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *Proc. German Conf. Artif. Intell.: Poster Demo Track*, 2012, pp. 59–63.
[44] CSDMC2010 SPAM corpus, "Spam email datasets," [Online]. Available: http://csmining.org/index.php/spam-email-datasets-.html, Accessed on : Oct. 02, 2016.
[45] C. Karlberger, G. Bayler, C. Kruegel, and E. Kirda, "Exploiting redundancy in natural language to penetrate Bayesian spam filters," in *Proc. 1st USENIX Workshop Offensive Technol.*, 2007, pp. 1–7.
[46] Apache Hadoop, "Welcome to apache hadoop," [Online]. Available: http://hadoop.apache.org/, Accessed on: Oct. 02, 2016.

Abdelrahman AlMahmoud received the BSc degree in information technology from the United Arab Emirates University, where he won the distinguished student award, along with the first place in the Senior Graduation Project Competition, and the MSc degree by research in engineering from Khalifa University, in 2014, where he is currently working toward the PhD degree. He joined EBTIC in 2011 where he is currently working as an Associate Researcher. His research interests are Privacy-Preserving analysis and Cybersecurity. He is a member of the IEEE.

Ernesto Damiani is a full professor with the Universita degli Studi di Milano, where he leads the SESAR research lab, and the leader of the Big Data Initiative at the EBTIC/Khalifa University, Abu Dhabi, UAE. He is the Principal Investigator of the H2020 TOREADOR project. He was a recipient of the Chester-Sall Award from the IEEE IES Society (2007). He was named ACM Distinguished Scientist (2008) and received the IFIP TC2 Outstanding Contributions Award (2012). He is a senior member of the IEEE.

Hadi Otrok received the PhD degrees in ECE from Concordia University. He holds an associate professor position in the Department of ECE, Khalifa University, an affiliate associate professor in the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, and an affiliate associate professor in the Electrical Department, École De Technologie Supérieure (ETS), Montreal, Canada. He is an associate editor at: the *Ad-Hoc Networks* (Elsevier), the *IEEE Communications Letters*, the *Wireless Communications and Mobile Computing* (Wiley). He co-chaired several committees at various IEEE conferences. Moreover, he is a TPC member of several conferences and reviewer of several highly ranked journals. He is a senior member of the IEEE.

Yousof Al-Hammadi received the bachelor's degree in computer engineering from KUSTAR (previously known as Etisalat College of Engineering), UAE, in 2000, the MSc degree in telecommunications engineering from the University of Melbourne, Australia, in 2003, and the PhD degree in computer science and information technology from the University of Nottingham, United Kingdom, in 2009. He is currently a director of Graduate Studies and assistant professor in the Electrical & Computer Engineering Department, Khalifa University of Science, Technology & Research (KUSTAR), Abu Dhabi, United Arab Emirates. His main research interests include the area of Information security which include Intrusion detection, botnet/bots detection, viruses/worms detection, artificial immune systems, machine learning, RFID security and mobile security. He is a member of the IEEE.