Mitigating SMS Spam by Online Detection of Repetitive Near-Duplicate Messages

Baris Coskun
AT&T Security Research Center
New York, NY
baris@att.com

Paul Giura
AT&T Security Research Center
New York, NY
paulgiura@att.com

***Abstract*—Short Message Service (SMS) spam is increasingly becoming a problem for many telecommunication service providers. Not only do SMS spam messages use mobile network resources abusively, but also in many cases they represent malware propagation vectors for mobile devices. In this work, we propose a network-based online detection method for SMS spam messages. The proposed scheme uses robust text signatures to identify similar messages that are sent excessively in the SMS platform and is robust against slight modifications in SMS spam messages. Additionally, the method uses a fast online algorithm which can be deployed in large carrier networks to detect spam activities before too many spam messages are delivered. It does not store SMS message contents, therefore it does not compromise the privacy of mobile subscribers.**

I. Introduction

Like every electronic messaging platform (i.e. Email, Instant Messaging, Web Forums, etc.), Short Message Service (SMS) in mobile networks is plagued by unsolicited bulk messages called **spam**. In fact, SMS spam constitutes $20 - 30\%$ of Asian markets, such as India and China [1]. SMS spam can be used for various purposes, such as unsolicited advertising, phishing, malware propagation, tricking subscribers into calling premium numbers, etc., which make SMS spam not only annoying but also potentially hazardous to subscribers' security and privacy [2]. Additionally, SMS spam raises a particular financial concern as it may incur additional wireless service charges to its victims due to delivered messages. Therefore it is crucial to detect and stop SMS spam messages before they are delivered to their destinations.

Mobile Network Operators (MNOs) have deployed various SMS spam mitigation techniques, such as message volume-based methods and message rate limitation, content-based filtering, subscriber feedback-based methods, sender reputation-based schemes, distributed honeypot and collaborative data sharing techniques and solutions that are based on senders' social networking characteristics [3] [4] [5] [6] [7]. Nevertheless, spammers are continuously adopting more advanced techniques to evade such defenses. Some of such evasion techniques include; **i) sophisticated message modification** to evade content-based (signature-based) spam filters; **ii) slow attacks sending spam in smaller batches** to evade volume-based spam detection methods; **iii) sender number variation** to evade reputation and social networking based schemes [1]. Furthermore, with the advent of smartphones and consequently mobile malware, in the future, spammers are likely to employ thousands of infected mobile devices (i.e. mobile botnets) to send spam messages thereby allowing spammers to employ much more sophisticated evasion techniques.

To keep up with this worrying trend of SMS spam, mitigation techniques should be evolving as well by constantly adding new techniques and algorithms to the arsenal. Motivated by this, we propose an online algorithm which efficiently detects repetitive transmission of similar messages to identify potential SMS spam messages. Our intuition is that, an SMS spam campaign essentially tries to deliver a certain information to millions of subscribers in the form of SMS messages, regardless of whether they originate from a single account or thousands of accounts. Therefore, messages of a certain SMS campaign cannot differ much, although they may be slightly different from each other. More specifically, since each message of a spam campaign is relatively a short text conveying essentially the same information, there is very little room for a spammer to create a substantially different content for each spam message.

Based on these observations, our proposed scheme monitors an SMS platform from a central vantage point within the mobility network and detects potential spam campaigns when it sees unusually high number of messages with similar content being transmitted. Note that the SMS platform employs store and forward architecture which allows monitoring all traffic from a central location [8]. In our algorithm, we use robust content signatures to tell whether the transmitted messages have similar contents. The basic idea is that, if two messages are similar, then certain portions of their content will exactly match with high probability. More specifically, if we consider each message as a set of blocks, *i.e. short sequences of subsequent characters*, then similar messages will have several blocks in common with high probability. As a result, one can detect SMS spam campaigns, when one observes a certain set of blocks being transmitted repeatedly within the SMS

platform.

With this work we make the following key contributions: **i)**We propose an online algorithm that can detect spam campaigns before they send substantial number of spam messages. **ii)**The proposed algorithm is robust against sender number variation. It can detect spamming campaigns which can potentially employ thousands of infected mobile devices that send spam. **iii)**It is robust against sophisticated message modification. It doesn't require the spam messages to be exactly the same. **iv)**The proposed algorithm is efficient and can be deployed in large carrier networks. It performs simple constant time operation for each message as it arrives at the monitoring center. **v)**Finally, the proposed method preserves user privacy since message contents never leave the SMS platform.

II. Detection Method

The basic idea of the method is to maintain an approximate count of message contents to see whether there is an abnormally high number of similar messages transmitted over a short period of time. For this purpose we use an efficient data structure called Counting Bloom Filter. In this section we elaborate this idea and formally introduce our proposed solution.

A. Problem Formulation

Each SMS message can be represented as a set of blocks, which are small chunks of consecutive characters within a message. Two transmitted messages are similar if they share substantial portion of their blocks. More specifically, we consider an SMS message is defined as a set $M = \{b_1, \ldots, b_m\}$, where $b_1, \ldots, b_m$ represent continuous blocks of characters. To measure similarity, s, between two messages M and M', we use Jaccard similarity metric, defined as follows:

$$s(M, M') = \frac{|M \cap M'|}{|M \cup M'|} \tag{1}$$

Using this, we say that two messages M and M' are similar, and write $M \sim M'$, if $s(M, M') \geq s_0$, where s_0 is the similarity threshold. Note that the number and diversity of blocks for each message has high impact on the similarity metric. Therefore, the method to create message blocks is important. We discuss how we select the blocks of a message later in this section. Having defined above the message representation and the similarity metric, we can formulate the problem as follows:

Consider i SMS messages M_1,M_2,....,M_i received at the SMS monitoring center. For the next received message, M_{i+1}, we try to answer whether there were more than t already received messages similar to M_{i+1}.

It is possible to find an exact solution to this question by storing and processing every observed message. However, this would be inefficient and limiting both in terms of memory and computational complexity, especially when the number of messages is very large. In this work we try to find an approximate answer to this question by using counting Bloom filters, which are described in the next section.

B. Counting Bloom Filters

A standard Bloom filter is a space-efficient probabilistic data structure used for representing a set in order to support membership queries [9]. A Bloom filter is identified by a bit array of size m with all the bits initially set to 0, and k independent hash functions with the range {1,...,m}. When an element of a set is inserted into the Bloom filter, it is first hashed with all k hash functions and all the corresponding zero bits in the bit array are flipped to 1. If one of the corresponding hash functions bits it already set to 1, then a *collision* occurs and the bit is not changed. When an element is tested for membership in a set, the Bloom filter is queried in constant time as follows. First, the element is hashed with all k hash functions and all the corresponding bits in the bit array are checked. If all bits checked are 1 then we say the element was inserted in the Bloom filter with some probability, called the *false positives* rate, introduced because of the collisions in the insertion process. If at least one corresponding bit is 0 then we know precisely that the element was not inserted in the Bloom filter. Thus, a Bloom filter has no false negatives, and space-efficiency is achieved at the cost of a small probability of false positives.

In our method we use a variation of the standard Bloom filter, called *counting* Bloom filter [10] [11], in order to efficiently detect an abnormal number of similar messages sent within a predefined time period. Essentially, a counting Bloom filter uses an array of counting bins rather than a single bit for each array position, representing the number of matches that are encountered for each corresponding position. In the rest of this paper the term Bloom filter always refers to counting Bloom filter unless stated otherwise.

C. Method Description

We base our algorithm on two fundamental characteristics of SMS spam messages: 1) many similar spam messages are sent over relatively short time intervals, 2) the spammer affords only to slightly modify the representation of similar messages, otherwise the content transmitted would get highly distorted, therefore spamming would not be serving its purpose. Suppose there are N messages sent in the network, then the set of all the blocks in the transported messages is $\mathbf{M} = \bigcup_{i=1}^{N} M_i$. We will use counting Bloom filters to insert the blocks of all the messages. A counting Bloom filter is defined as an array of bins $B = [B[1], \ldots, B[m]]$ and a set of hash functions $\mathbf{F} = \{F_1, \ldots, F_k\}$, with each hash function having the range $\{1, \ldots, m\}$. We use a counting Bloom filter for each time period (e.g., minute) when the messages are recoded in the system. As such, we have the time window T_1 for counting Bloom filter B_1, T_2 the time window for Bloom filter B_2, etc. We assume the size of all the counting

Bloom filters is m bins. For each bin, $B[i]$, we maintain a bin threshold $t[i]$ representing the maximum counter value allowed for that bin in a counting Bloom filter. The threshold value is based on the initial observed data. The bin threshold purpose is to capture expected number of occurrences for each block during a time window, thereby enabling us to detect sudden changes in the sending rate of similar messages. Our SMS spam campaign detection method works as follows:

- **Initialization:** Insert messages M_1, ..., M_i into counting Bloom filters $B_1, \ldots, B_j$ corresponding to time periods T_1, ..., T_j. Then compute bin thresholds, $t[1], \ldots, t[m]$. Bloom filter B_j is the current filter corresponding to time period T_j. Note that typically $i >> j$.
- **Step 1:** Upon receiving of new message M_{i+1}, partition new message into blocks. Suppose the message yields z blocks. Compute hashes each of these z blocks and check if incrementing the corresponding bin counters in the Bloom filter exceeds the corresponding bin thresholds. If the bin thresholds are exceeded for more than $s_0 z$ of the blocks, then raise an alarm and mark the message as potential spam.
- **Step 2:** When the time slot for Bloom filter T_j expires create a new empty counting Bloom filter B_{j+1} for new time period T_{j+1}. Discard Bloom filter B_1 for time period T_1 an update the bin thresholds to include values from filter B_j.

Blocks Selection:
A trivial method to create blocks for an SMS message M is to select a block size z and consider the first block b_1 containing the first z characters, block b_2 the next z characters, and so on. However, this simple blocking scheme does not capture the consecutiveness of the blocks. That is, even two messages have exactly the same blocks and their similarity is 1 (e.g. perfect similarity) based on the metric defined in Equation 1, the arrangement of the blocks might be different, the content of the messages being completely different. In [12] authors suggest to use Rabin fingerprinting and shingling in order to solve the consecutiveness problem when blocking network traffic payload. The Rabin fingerprinting methods select the boundaries of the blocks by computing a rolling polynomial function over the content itself. Shingling methods, on the other hand, solve the consecutiveness problem by appending to each block a small overlap of the next block. Since SMS messages are essentially short strings of text and since Rabin fingerprinting methods don't guarantee a minimum number of blocks for a given message size, we chose to use a shingling variation for building messages blocks called n-grams selection. The n-grams of a text message are all the substring of size n for that message. They can be enumerated by getting the first substring of size n, and then shifting one position the start and the end of the new substring until reaching the end of the message. In this way adjacent blocks will always overlap and the consecutiveness problem is addressed.

Adversarial Model:
Given two messages M and M' our goal is to detect if the messages have similar meaning by using the Jaccard similarity function defined in Equation 1. The spammer's goal, on the other hand, is to avoid detection by slightly altering each spam message. We quantify the resilience of the proposed detection scheme by the edit distance d between two messages that the similarity metric can withstand. The edit distance between M and M' is defined as the number of operations (replace, insert, delete) needed to transform message M into M'. If message M has z characters, and n is the size of n-grams, then there are at most $z - n$ different blocks in the message. Suppose a spammer wants to create a message M' with edit distance d starting from M. Then, for each modification of a character c in M, she will spoil all the n-grams that contain c, that is n blocks. Thus, the maximum number of different blocks that can be spoiled after d operations is $n \cdot d$. Hence, the fraction of blocks that can determine the similarity is $\frac{d \cdot n}{z-n}$. Therefore, in order to withstand d modifications from message M with size z, to message M', the similarity threshold parameter has to satisfy:

$$s_0 < 1 - \frac{d \cdot n}{z - n} \tag{2}$$

For instance consider two messages with $z = 100$ characters, differing at $d = 10$ positions. In order to declare these two messages being similar when the block size is set to $n = 5$, the similarity threshold should be $s_0 < 1 - \frac{10 \cdot 5}{100-5} \approx 0.53$. However, notice that this represents the worst case and in practice $d = 10$ different characters may not spoil all $10 \cdot 5 = 50$ blocks. Therefore setting s_0 greater than it's worst case bound will still yield sufficient detection results as we will discuss in the next section.

III. Experiments

A. Data Set

In our experiments we use a dataset of 104,809 YouTube [13] comments from different YouTube videos. We first collected the data by crawling YouTube comments using YouTube public API, starting from few seed videos. Then, we preprocessed the raw data to get a data set which is closer in properties to the real SMS data. We used publicly available YouTube data for two main reasons: a) the data is widely accessible and there are no privacy concerns associated with accessing it, and b) the data characteristics are very similar to real SMS data (i.e. short strings of text generated by users), thus making YouTube data very suitable for simulating SMS traces.

B. Preprocessing

After collecting the YouTube videos comments, we first partition each comment in short sets of 140 octets (160 characters) keeping also the trailing block whose length could be smaller than 160 characters. If the comment is

not at least 140 octets long then we consider the entire comment as an SMS message. Based on the instances of SMS spam messages that we observed, we concluded that a spammer needs to send a message with at least a minimum number of characters in order to increase the chances of communicating the message across to the spam victims. Thus, in our experiments we selected the minimum message size to be 50 characters. Therefore, from the list of all the messages collected as described before, we consider only those whose length is larger than 50 characters. Finally, we remove the symbols in the messages that don't carry any information such as spaces, multiple punctuation signs, etc. For cases when we have a repetitive punctuation sign (e.g. !!!!, ???, etc) we maintain a single occurrence of such a sign.

C. Block Size Selection

Message blocks size is an important parameter for the accuracy of our spamming campaign detection method. Basically, using too small blocks will result in high robustness against slight message modifications but will also cause unrelated messages appearing very similar to each other according to the similarity metric described in Section II. On the other hand, too large blocks will successfully distinguish unrelated messages from each other while sacrificing robustness against message modifications. Based on this observations, we select the block size parameter as the smallest value that has the potential to yield the smallest similarity metric between any two random messages.

To determine such a block size value, we design an experiment to assess the effect of the block size on the similarity metric of any two messages selected at random. In the experiment, we randomly select 1000 pairs of messages and compute the similarity metric for each pair for each block size, from 1 to 20 characters. Based on the results of this experiment, we select the block size parameter to be 5, which is the smallest value that yields very small similarity metric between any two random messages. We use this value for the rest of results reported in our experiments.

D. Detection and False Positive Results

In this section we investigate detection rate and false positive probability of the proposed method under different parameter settings. To assess these probabilities, we first split our dataset into two parts for training and testing. We use the training part to establish Bloom filter bin thresholds. Basically we insert N_M randomly selected messages from the training set into N_{BF} Bloom filters, each using $k = 2$ hash independent functions. During this process, we randomly pick messages without replacement in order to make sure that each message is used only once and each Bloom filter has different sets of messages. Once we insert the training messages into Bloom filters, we first compute the average value for each bin as $\overline{B}[i] = \frac{1}{N_{BF}} \sum_{j=1}^{N_{BF}} B_j[i]$, where i is the bin number and B_j is the j^{th} Bloom filter. Then we set the threshold value for each bin as $t[i] = max(\overline{B}[i], 1)$, which makes all threshold values at least 1, thereby ensuring that a spam campaign must have at least two similar messages in order to be classified as a spam campaign. Otherwise, the proposed scheme would potentially classify previously unseen single messages as spam.

As for testing, we use two Bloom filters, one for determining detection rate and one for determining false positive rate. We insert N_M randomly selected messages, again without replacement, from the testing data into the first Bloom filter representing the case where there is no spam activity. To represent spam activity, on the other hand, we first pick a random message from the training set as the spam seed. Then we insert N_S different variations of the seed message into the second Bloom filter. We generate different spam variations by substituting d characters at random positions of the seed message with random characters. Notice that d is essentially the edit distance between different variations of the spam messages and the seed spam message. After inserting spam messages, we then fill the rest of the second Bloom filter with $N_M - N_S$ randomly selected messages. As a result, we obtain two Bloom filters with N_M messages, one containing a spam activity and one not.

To test the detection capability of the proposed scheme, we check a random variation of the seed spam message against the second Bloom filter, which contains previously inserted spam messages. If the fraction of blocks, whose all corresponding bins exceed the corresponding threshold values, is greater than s_0, we declare the test message as a spam message as it is evident that messages with similar contents have been previously observed. To compute the detection rate, we repeat this procedure 1000 times. To test the false positives, on the other hand, we check a randomly selected message against the first Bloom filter, which contains no spam activity. Again we repeat this procedure 1000 times.

Detection Rate: We present detection rate for different Bloom filter sizes and different message rates in Figure 1. For this figure, we use $N_{BF} = 3$ Bloom filters for training and we set detection threshold to $s_0 = 0.7$. We also fix number of spam to ($N_S = 50$) and spam edit distance to ($d = 10$). We make several observations in Figure 1. First of all detection rate tends to decrease as the number of messages increases. This is expected since the number of spam messages is constant (i.e. $N_S = 50$) and it becomes harder to pick up the traces of spam activity as the number of messages in the background increases. We also observe that the detection performance improves drastically when we increase the Bloom filter size. More specifically, for Bloom filters larger than $500,000$, the proposed scheme reaches almost 100% detection rate even when only 5 spam messages are blended in total of $10,000$ SMS messages. This is due to the fact that larger Bloom filters have less collision rates–*i.e. two different blocks being mapped*

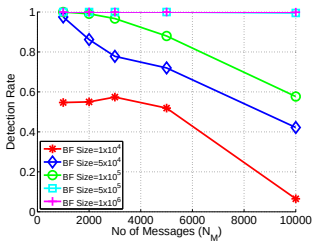

Fig. 1. Detection Rate for different Bloom filter sizes and different number of messages (N_M), where $N_{BF} = 3$, $N_S = 50$, $d = 10$ and $s_0 = 0.7$

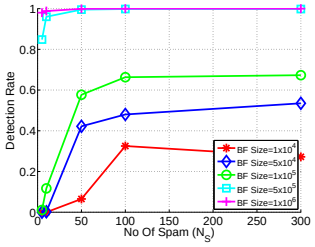

Fig. 2. Detection Rate for different number of spam (N_S) and different Bloom filter sizes, where $N_{BF} = 3$, $N_M = 10,000$, $d = 10$ and $s_0 = 0.7$

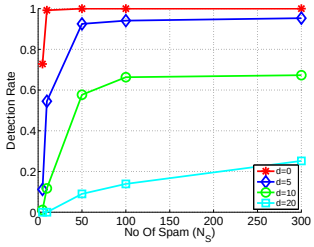

Fig. 3. Detection Rate for different spam edit distance values (d) and different number of spam (N_S), where $BFSize = 100,000$,$N_{BF} = 3$, $N_M = 10,000$ and $s_0 = 0.7$

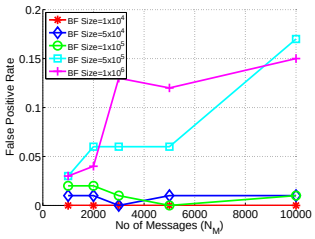

Fig. 4. False Positive Rate for different Bloom filter sizes and different number of messages (N_M), where $N_{BF} = 3$ and $s_0 = 0.7$

to a same bin–, thereby keeping a more accurate count of inserted elements. In other words, if more blocks are mapped to a same bin in Bloom filters, then the bin threshold values computed during training ($t[i]$) will be high. As a result, some spam activity will not be sufficient to have enough number of bins exceeding threshold values.

Similarly, Figure 2 and Figure 3 shows the effect of number of spam and effect of spam edit distance respectively. As expected the proposed algorithm performs better as the number of spam messages increases and the edit distance between spam messages decreases.

False Positives: We clearly see from detection results presented above that, in order to achieve high detection rates in all cases, we need to use large Bloom filters (i.e. $BFSize \geq 500,000$). However, besides requiring more memory, large Bloom filters also cause high false positive rates, as we observe in Figure 4. This is because, bin thresholds are relatively low for large Bloom filter since very few different blocks are mapped to the same bins. As a result, some messages, especially the ones which contain high number of commonly used words and phrases, have a good chance of causing many of the bins to exceed corresponding threshold values. One way to mitigate this is to employ a white list of commonly occurring words, which would be omitted by the proposed algorithm.

IV. Limitations and Potential Improvements

Our proposed scheme is designed to process high number of messages in realtime at a cost of some accuracy. Consequently, the major limitation of our proposed scheme is high false positive rates especially when very large Bloom filters are used. As mentioned earlier, this is mostly due to commonly used words in the data set such as, *'the', 'and'*, etc. and can be mitigated by employing a white list of commonly used words. Such whitelists can also help the proposed scheme to mitigate commonly occurring phrases such as *'Happy Birthday'*, etc. Furthermore, the output of the proposed scheme can be further processed by more accurate methods, which tend to be computationally more complex, to improve accuracy. With such a setting, our proposed scheme can be considered as a preprocessing step to computationally more intensive filtering schemes, thereby enabling them to scale up to large and realtime deployments.

The proposed scheme requires at least few similar SMS messages in order to detect spam campaigns. Therefore, it may miss extremely slow spammers. Such spammers can usually be detected by content-based spam filters which process each message separately and independently.

V. Related Work

There have been several works previously proposed in the context of SMS spam detection. In [14] [15] [16] authors explore content based filtering of SMS spam messages. While these methods are mainly inherited from email spam filtering schemes, authors acknowledge that short messages don't contain sufficient information in order for complex machine learning algorithms to be applied. Therefore in [16] authors propose to employ other features such as character bigrams, trigrams, etc, to increase performance. To further improve detection accuracy, in [17] authors suggest to use length of message, and appearance of certain kind of information (phone number, url), and in [18] authors propose to include features from stylistic aspects of SMS messages. In [19] authors describe a scheme which employs a fast string matching algorithm to filter SMS spam containing predefined keywords. Despite being effective, this method is only limited to the set of blacklisted words and falls short when each spam message is slightly modified. In [20] authors propose a clustering based SMS spam detection method, where they observe that SMS spam messages have similar contents therefore they are likely to be clustered in a random subspace. In [7] authors propose using social and temporal characteristics of message senders to identify spammers. These methods run into scalability problem and it is challenging to deploy these methods in large networks. As a different perspective, in [21] authors propose to mitigate mobile spam, by displaying CAPTCHA to senders. This method, however, increases burden on SMS platform and requires client side applications on mobile devices.

Standard Bloom filters [9] and their variations [10], [11] are used in a wide range of applications such as representing dictionaries for spell checking, database tables record membership for distributed storage systems [22],

or network forensic applications to store network packets digests [12], [23]. In [24] authors propose a large suite of payload partitioning methods used for a network payload attribution system, and insert the payload blocks into Bloom filters.

Our proposed method essentially sketches the observed SMS traffic and tries to answer certain questions based on this sketch. Several different sketching-based methods are proposed in different contexts. In [25] authors propose CountMin sketch which is used to answer range and point queries about the input data stream. Similarly, [26] proposes an algorithm to efficiently maintain quantiles of the input. In the context of network security, in [27], authors propose a network flow sketching algorithm to efficiently detect correlated flows within a stream of network data.

VI. Conclusion

We present an efficient method to quickly identify an SMS spamming campaign by detecting an unusual number of similar messages sent in a network over a short period of time. Our method uses counting Bloom filters to maintain approximate count of message content occurrences. Experimental results show that we can achieve a detection rate of nearly 100% with a counting Bloom filter of size larger than 500,000 bins for detecting as few as 10 similar spam messages that differ by at most 20 characters within $10,000$ regular SMS messages. The proposed scheme is a significant addition to existing SMS spam arsenal and will achieve high performance and accuracy when used in coordination with existing spam detection schemes. As future work, we seek to test the method with real SMS traffic and to address the challenges of online deployment that uses extremely large datasets.

References

[1] GSMA White Paper, "Sms spam and mobile messaging attacks - introduction, trends and examples: A white paper," January 2011, www.gsmworld.com/documents/srs_attacks_threats.pdf.

[2] Cloudmark White Paper, "CloudmarkÕs definitive guide to sms spam," June 2011, http://www.cloudmark.com/releases/docs/sms_spam_guide.pdf.

[3] GSMA Spam Reporting Service, www.gsmworld.com/documents/srs_overview.pdf.

[4] V. V. Prakash and A. O'Donnell, "Fighting spam with reputation systems," *Queue*, vol. 3, pp. 36–41, November 2005. [Online]. Available: http://doi.acm.org/10.1145/1105664.1105677

[5] V. V. Prakash and A. J. OÕDonnell, "How collaborative filtering can stop future forms of messaging abuse," *Cloudmark White Paper*, June 2011, www.cloudmark.com/releases/docs/whitepapers/How_Collaborative_Filtering_Can_Stop_Future_Forms_of_Messaging_Abuse_v05.pdf.

[6] A. J. OÕDonnell and V. V. Prakash, "Applying collaborative anti-spam techniques to the anti-virus problem," *Cloudmark White Paper*, www.cloudmark.com/releases/docs/whitepapers/Applying_Collaborative_AntiSpam_Techniques_to_AntiVirus_Problem_v04.pdf.

[7] C. Wang, Y. Zhang, X. Chen, Z. Liu, L. Shi, G. Chen, F. Qiu, C. Ying, and W. Lu, "A behavior-based sms antispam system," *IBM J. Res. Dev.*, vol. 54, November 2010.

[8] J. Brown, B. Shipman, and R. Vetter, "Sms: The short message service," *Computer*, vol. 40, December 2007.

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422–426, July 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[10] M. Mitzenmacher, "Compressed bloom filters," in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, ser. PODC '01. New York, NY, USA: ACM, 2001, pp. 144–150. [Online]. Available: http://doi.acm.org/10.1145/383962.384004

[11] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, pp. 281–293, June 2000. [Online]. Available: http://dx.doi.org/10.1109/90.851975

[12] M. Ponec, P. Giura, H. Brönnimann, and J. Wein, "Highly efficient techniques for network forensics," in *Proceedings of the 14th ACM conference on Computer and communications security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 150–160. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315265

[13] YouTube, www.youtube.com.

[14] G. V. Cormack, J. M. G. Hidalgo, and E. P. Sánz, "Feature engineering for mobile (sms) spam filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07, 2007.

[15] G. V. Cormack, J. M. Gómez Hidalgo, and E. P. Sánz, "Spam filtering for short messages," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ser. CIKM '07, 2007.

[16] J. M. Gómez Hidalgo, G. C. Bringas, E. P. Sánz, and F. C. García, "Content based sms spam filtering," in *Proceedings of the 2006 ACM symposium on Document engineering*, ser. DocEng '06, 2006.

[17] W.-W. Deng and H. Peng, "Content based sms spam filtering," in *Machine Learning and Cybernetics, 2006 International Conference on*, 2006.

[18] D.-N. Sohn, J.-T. Lee, and H.-C. Rim, "The contribution of stylistic information to content-based mobile spam filtering," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ser. ACLShort '09, 2009.

[19] J. Liu, H. Ke, and G. Zhang, "Real-time sms filtering system based on bm algorithm," in *International Conference on Management and Service Science (MASS), 2010*, 2010.

[20] S. Dixit, S. Gupta, and C. Ravishankar, "Lohit: An online detection & control system for cellular sms spam," in *IASTED Communication, Network, and Information Security*, 2005.

[21] P. He, Y. Sun, W. Zheng, and X. Wen, "Filtering short message spam of group sending using captcha," in *Proceedings of the First International Workshop on Knowledge Discovery and Data Mining*, 2008.

[22] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, pp. 4:1–4:26, June 2008. [Online]. Available: http://doi.acm.org/10.1145/1365815.1365816

[23] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, 2005. [Online]. Available: http://www.internetmathematics.org/volumes/1/4/Broder.pdf

[24] M. Ponec, P. Giura, J. Wein, and H. Brönnimann, "New payload attribution methods for network forensic investigations," *ACM Transactions on Information and System Security*, vol. 13, 2010.

[25] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, pp. 29–38, 2004.

[26] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "How to summarize the universe: Dynamic maintenance of quantiles," in *In VLDB*, 2002, pp. 454–465.

[27] B. Coskun and N. Memon, "Online sketching of network flows for real-time stepping-stone detection," in *ACSAC'09: 25th Annual Computer Security Applications Conference, Honolulu, HI*, Dec 2009.