Payload Attribution via Character Dependent Multi-Bloom Filters

Mohammad Hashem Haghighat, Mehdi Tavakoli, and Mehdi Kharrazi

***Abstract*—Network forensic analysts employ payload attribution systems (PAS) as an investigative tool, which enables them to store and summarize large amounts of network traffic, including full packet payload. Hence an investigator could query the system for a specific string and check whether any of the packets transmitted previously in the network contained that specific string. As a shortcoming, the previously proposed techniques are unable to support wildcard queries. Wildcards are an important type of query that allow the investigator to locate strings in the payload when only part of the string is known. In this paper, a new data structure for payload attribution, named Character Dependent Multi-Bloom Filters, will be presented which, in addition to improving the previously proposed techniques, is able to support wildcard queries as well.**

To this end, a theoretical study of the proposed method was conducted in order to evaluate its false positive when responding to queries and subsequently the theoretical analysis is verified through a number of experiments. Furthermore, comparisons are made between the proposed method and the state-of-the-art attribution techniques presented in the literature. The results suggest that, using the Character Dependent Multi-Bloom Filters, one can obtain a data reduction ratio of about 265 : 1 opposed to 210 : 1 as obtained by the previously proposed state-of-the-art techniques assuming a similar false-positive rate. More importantly, the results indicate that a wildcard query with seven unknown characters would take approximately less than 1 second to process, using the proposed method; while given the previous techniques, as an exhaustive search is required, the same query takes about 4500 years to process.

***Index Terms*—Bloom filter, network forensics, payload attribution system, wildcard search.**

I. Introduction

A NETWORK forensic analyst is responsible for a difficult task. Unlike nonvirtual crimes, where the criminal often leaves a trail of evidence, in the cyber world not much evidence is left behind by the cybercriminal. For example, assume that a new worm has spread in your organizational network. How would you go about finding the nodes infected within your network? Who was the source of infection? Was he an insider? If not, did you have proper defenses installed at the edge of your network?

In such cases, one approach would be to capture and store full packet traces for any possible future forensics investigation. This way, next time a worm is propagated in the network, the forensic analyst could use the worm signature to search in traffic, looking for the initial source of infection, the pattern in which infection had propagated, infected hosts within the network, and possible vulnerabilities in network defenses. Nevertheless storing full packet traces, specially for extended time periods, is an extremely difficult, and at times impossible, task given the volume of network traffic present even at the organizational level, let alone in the case of larger networks such as a service provider network. In fact, storing full packet traces would require 10 TBytes of daily storage for a 1 Gbps link. Thus storing data for a few month will push the storage requirements into the order of Peta-Bytes and additionally, given such large data sets, querying for specific information would be quite costly, and with large overhead.

In order to minimize the storage requirements, while collecting full packet traces, Kulesh *et al.* [1] proposed a payload attribution technique based on bloom filters [2] with bounded false positive rate. Therefore, one could submit a query and find out if a specific byte pattern was seen in the network traffic and identify the source and destination of the traffic. Following their work, there have been other proposals [3]–[5] which have improved upon the data reduction ratio while lowering the false positive rate. For a general survey on related techniques, see [6].

However, the current techniques are limited in the type of queries they could respond to. This is an extremely important issue given the current trend in network attacks and the complex signatures required to investigate such attacks [7]–[10]. One example could be the increasing number of polymorphic worms in the wild. These worms change their appearance as much as possible before retransmitting themselves to the next victim. This change in appearance is usually done by reencrypting the malware code by a different encryption key every time which results in a different byte code for the malware. However as noted in [7], some invariant parts can be extracted from the worm payload, which are then separated by random characters (i.e. parts of the malware which were reencrypted). Hence the signature of interest for the worm would be in the form of "A $*$ B", where "A" and "B" are two invariant strings separated by a set of unknown random characters.

To the best of our knowledge, the current state-of-the-art techniques are only able to query for A and B independently and are not able to handle the more generalized queries noted above. In what follows, a new data structure for payload attribution called Character Dependent Multi-Bloom Filters is introduced, abbreviated as CMBF, specifically the contributions are:

- CMBF is able to support wildcard queries, where a limited number of unknown characters are present in the query excerpt, as opposed to the earlier techniques which require exhaustive search in order to respond to such queries. To

Manuscript received June 20, 2012; revised February 08, 2013; accepted February 27, 2013. Date of publication March 12, 2013; date of current version March 29, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yong Guan.

The authors are with DNS Laboratory, Department of Computer Engineering, Sharif University of Technology, Tehran 145888-9694, Iran (e-mail: haghighat@ce.sharif.edu; metavakoli@ce.sharif.edu; kharrazi@sharif.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2252341

clarify the issue, later in the manuscript it will be shown that with CMBF, a query with seven unknown characters would take approximately less than 1 second to process; while using exhaustive search to process the same query would require 4500 years.

- A theoretical study of the proposed method is conducted, in order to calculate the false positive rate of the CMBF. Furthermore, the optimal parameters which minimize the false positive rate while maximizing the data reduction ratio are calculated.
- CMBF is able to achieve a data reduction ratio of about 265 : 1 as compared to the reduction ratios of about 210 : 1 obtained with the state-of-the-art technique proposed earlier, having the same false positive rate.

The remainder of this paper is organized as follow, in Section II a brief overview of related work is presented. In Section III, we introduce the Character Dependent Multi-Bloom Filters. Subsequently, a theoretical evaluation of CMBF is conducted in Section IV, and in Section V the proposed technique is evaluated through a number of experiments based on collected traffic traces. Finally, the paper is concluded in Section VI.

II. Related Work

There have been a number of techniques proposed for payload attribution with the goal of determining information such as the packet source, flooding source, or connection chains. In general, these techniques could be categorized into header and payload based techniques. Header based techniques focus on the packet headers to create audit clues. In this respect, Snoeren *et al.* [11] proposed SPIE[1] with which packet digests (i.e. hash) are calculated from the header as well as the first eight bytes of the payload. These digests are inserted into a bloom filter for a short time duration. When a third party device such as an IDS or a firewall detects a suspicious activity, SPIE could be utilized to trace back the packet to the source of the connection. As another example, Demir *et al.* [12] proposed SBL,[2] with which TCP connection duration based on SYN and FIN packets as well as its source and destination IP addresses are stored (i.e. essentially a simplified netflow). This information is later used for IP trace back and investigation of malicious network activities.

With regards to the payload based techniques, around which the present study revolves, full packet payload is processed and stored. The first work in this area was presented by Kulesh *et al.* [1] in 2004, which introduced HBF.[3] In HBF, packet payloads are blocked and sampled using a fix sized sliding window, and then inserted into a hierarchical bloom filter. An important characteristic of HBF was its ability to store large amounts of data for a long duration through the data reduction ratio obtained by employing bloom filters. Nevertheless it is noteworthy that HBF suffers from the offset collision and alignment problems as noted in [4], [5]. This issue arises when HBF concatenates an offset number to each block before inserting it into the bloom filter in order to preserve the order of the blocks in a packet. However, it should be noted that inserting more than one packet into the bloom filter, would mean keeping more than one block with the same offset number, which leads to the offset collision problem. Additionally, when we want to process a query, it is necessary to find the correct alignment of the blocks, which, in turn, increases the query response time.

To solve the HBF alignment problem, Cho *et al.* [3] proposed RBF[4] which considers all possible alignments at once. In essence, RBF employs several bloom filters each with an assigned fixed size window. It then slides the fixed size windows through the payload in order to insert the blocks into the corresponding bloom filter. However, the reported results indicate that RBF achieves a data reduction ratio similar to the best case of HBF with the same false positive rate.

Alternatively, Ponec *et al.* [4], [5] proposed a number of attribution techniques, namely, the VBS,[5] WBS,[6] and WMH,[7] in order to solve the alignment and offset collision problems. In addition, these techniques yield better data reduction ratio and accuracy rates when compared to the previously proposed techniques. VBS determines block boundaries using Rabin fingerprinting [13] method. It slides a window through the payload and calculates the fingerprint value for each position of the window. The block boundary is set when the fingerprint mod m is equal to zero, where m can be any arbitrary value. Similarly, VBS fixes the alignment problem by specifying block boundaries based on the payload, however, this causes a new problem. Large blocks maybe constructed, which, in turn, would not allow the investigator to query for smaller excerpts. Similarly, there is also the possibility that a large number of small blocks could be constructed, with which the bloom filter will fill up quickly, where the latter problem is resolved by defining a minimum threshold for the block size.

Alternatively in WBS, boundaries are determined using the winnowing technique [14], which guarantees that the block size could not exceed a specified value. Similar to VBS, a window is slid through the payload and fingerprint values for each position of the window are calculated. Subsequently, another window (winnowing window) is slid through the fingerprint values and the block boundary is set before the maximum value of each position of the window. Lastly, the WMH method operates by running multiple instances of WBS with different parameters in order to decrease the system false positive.

The payload attribution systems discussed above, provide a facility to store a large amount of data for a long period of time, yet they are unable to respond to wildcard queries. In what follows, an alternative to the previously proposed technique is presented, which is able to accept wildcard queries. Furthermore, a better data reduction ratio is also obtained, in comparison to the previously proposed techniques, while maintaining a similar false positive rate.

III. Character Dependent Multi-Bloom Filters

Similar to the previously proposed techniques, CMBF employs bloom filters [2], a simple and efficient data structure, as

[1] Source Path Isolation Engine.

[2] Session Based Logging.

[3] Hierarchical Bloom Filter.

[4] Rolling Bloom Filter.

[5] Variable Block Shingling.

[6] Winnowing Block Shingling.

[7] Winnowing Multi Hashing.

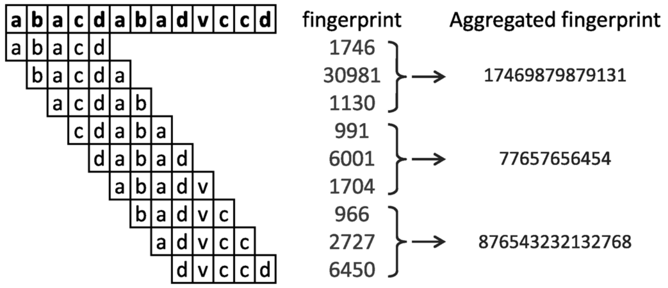


Fig. 1. In this example, the string "abacdabadvccd" is processed resulting in the calculation of the aggregated fingerprints which are in turn inserted into the CMBF. The window size is set to 5, and an aggregation factor of 3 is used.

an essential building block. A bloom filter consists of an m bits array and k different hash functions of range m, in which, each maps an element into a bloom filter cell. At first, the array is set as 0. In order to insert an element, one should hash it, using these k functions to get the k positions and set the corresponding cells to 1. To query a value, the value should be hashed by the hash functions and should be checked to see whether the respective cells have been set to 1 or not.

CMBF employs 256 distinct bloom filters in order to create a one-to-one way mapping for each possible byte to a unique bloom filter. For example, bloom filter number 0 corresponds to character "*null*" or number 65 corresponds to character "*A*". In addition, it should be mentioned that, we use only one hash function for inserting values in the bloom filters. The CMBF insertion phase consists of the following steps:

1) Sliding a window of size w through the string.
2) Computing the fingerprint for each position of the window as below:

$$fingerprint(c_i, c_{i+1}, \ldots, c_{i+w-1}) = (c_i \bmod q) \times p^{w-1} + (c_{i+1} \bmod q) \times p^{w-2} + \cdots + (c_{i+w-1} \bmod q) \times p^0 \quad (1)$$

in which q and p are two random numbers and are smaller than 256 and $q \leq p$. Note that c_i is the i^{th} character of the string. It should be noted that this fingerprinting step results in a false positive, although its contribution to the total false positive value is negligible as shown in Section IV.
3) Aggregating g calculated fingerprints using a preconsidered aggregation function F (hereafter we call g as the "Aggregation Factor"). For example one could simply concatenate the calculated fingerprints.
4) Finding the corresponding bloom filter according to the first byte that was involved in the aggregated fingerprint calculation.
5) Inserting the aggregated fingerprint in the chosen bloom filter.

As an example, suppose that we want to insert "abacdabadvccd" into the bloom filter as shown in Fig. 1. We consider a window of size 5 and slide it through the string. Then we compute the fingerprint for each position of the window. Afterward, we aggregate each 3 fingerprints and finally insert the results in the appropriate bloom filter (e.g. the second aggregated fingerprint is inserted in the bloom filter responsible for character c).

In order to process a query, one should execute steps 1 to 4 from the above list on the query string and then check whether the resulted aggregated fingerprints were inserted in the corresponding bloom filters or not. Note that CMBF has the alignment problem, similar to HBF [1], because the first used character in the aggregated fingerprint calculation is not clear and all possible alignments should be tested.

In order to support wildcard queries, CMBF uses fingerprint modulo (q) in the fingerprint calculation, which maps each string byte to a value (class) between 0 and $q-1$. Therefore the query space size is restricted as one should use a value between the range of $[0, q-1]$ for calculating the fingerprint instead of all possible values between 0 to 255. As an example, suppose string "abcd?eghi" is queried with $q = 4$. The query is processed by constructing four subexcerpts: ("abcd0eghi", "abcd1eghi", "abcd2eghi", "abcd3eghi"). Each subexcerpt is queried independently and if either results in a positive response then the response to the original query would be positive as well. In Section V, we will study the effect of different modulo values on the response time when conducting a wildcard query search.

It is to be noted that CMBF is able to handle more complex queries as well. For example if the query excerpt is "abcde[m-p]fghij" with $q = 8$, then the subexcerpts are created by mapping each of the 4 possible unknown characters (i.e. m, n, o, and p) into the proper class between 0 and $q-1$. These subexcerpts would be "abcde5fghij",[8] "abcde6fghij", "abcde7fghij", and "abcde0fghij". Similarly, other queries could be processed by CMBF. In the following section we will evaluate the accuracy of the proposed CMBF technique through theoretical analysis.

IV. Theoretical Analysis

In general, the system accuracy is measured based on the false positive and false negative rates. A false positive occurs when a query string not inserted into the system is incorrectly stated as present in the system. Alternatively, a false negative occurs when a string inserted into the system earlier, is stated as absent from the system when queried. False negatives are present in the payload attribution systems which employ a fixed size block to sample the packet payload (i.e. CMBF, HBF, etc.). For instance, suppose that string "*abcdefghi*" has been inserted into the CMBF with window size of 4 and aggregation factor of 3. Therefore, two aggregated fingerprints are constructed by considering "*abcdef*" and "*defghi*", which in turn are inserted in the bloom filters. Now if we query for the excerpt "*cdefgh*" CMBF will respond that the excerpt was not found, which is incorrect. It is noteworthy that false negative in CMBF would occur only for the small excerpts, between $w + g - 1$ bytes (i.e. the minimum required length for creating an aggregated fingerprint) and $(w + g - 1) + (g - 1)$ bytes. Calculating the false negative rate is beyond the scope of this work, as it greatly depends on the length of the queries submitted to the CMBF. Therefore we will focus only on calculating the false positive rate in the remainder of this manuscript.

There are two types of false positives with CMBF. First, and as CMBF employs bloom filters, there is the *bloom filter false positive*. Secondly, by mapping string characters into classes be-

[8] Character m would be mapped to class 5 as $ASCII(m) mod\ 8 = 5$.

tween [0] to $[q-1]$ for computing the fingerprints, there is the possibility of collision between two different characters, and hence false positive, which is called *fingerprint false positive*. Interestingly, and as shown later in the manuscript, the fingerprint false positive rate is negligible compared to the bloom filter false positive rate. In what follows we will conduct a theoretical study of the noted false positive types.

A. Bloom Filter False Positive

Bloom filter reduces the required space size, however compressing data without incurring any cost is impossible. In fact, there is the possibility of a false positive, where the bloom filter may state that a specific string was inserted into it when it was not. On the other hand, a bloom filter does not have any false negative, in other words, it will never respond negatively to a query for an inserted string. Li Fan *et al.* [15] showed that the probability of false positive in the bloom filter can be computed using the following relation:

$$FP = \left(1 - \left(1 - \frac{1}{m}\right)^{kN}\right)^{k} \tag{2}$$

where m is the bloom filter length, N is the number of inserted items, and k is the number of used hash functions.

Lemma 1: Let n be the total number of stored characters in CMBF, m be the length of each of the 256 bloom filters, and g be the aggregated factor. The false positive for each bloom filter in the CMBF is:

$$a = \left(1 - \left(1 - \frac{1}{m}\right)^{\frac{n}{256g}}\right) \tag{3}$$

As described earlier, CMBF uses only one hash function but employs 256 bloom filters. Therefore, if we assume that the characters are distributed uniformly in the string, the insertion ratio of each bloom filter will be $N = n/256g$, where g is the aggregation factor and n is the total number of inserted characters in the CMBF bloom filters set. As a result, according to relation (2), the false positive rate of each bloom filter is:

$$a = \left(1 - \left(1 - \frac{1}{m}\right)^{\frac{n}{256g}}\right) \tag{4}$$

□

Theorem 1: Let FP_{BF_n} be the bloom filter false positive based on n inserted characters, a be the false positive rate of each bloom filter, g be the aggregation factor, and l be the excerpt length. Then,

$$FP_{BF_n} < \frac{(1 + 255 \times \sqrt[g]{a})^l - 1}{256^l} \tag{5}$$

Proof: Suppose the excerpt S, which was not inserted be queried and a false positive occurs because of a string S' that had been inserted before. This could happen when there are i number of nonequal bytes between S and S', hence all possible values for i ranging from 1 to l should be considered in the false positive calculation:

Lemma 2: Let $FP_{BF_{n,i}}$ be the bloom filter false positive in the case that S and S' differ in i bytes, then,

$$FP_{BF_{n,i}} < \frac{\binom{l}{i} \times (255 \times \sqrt[g]{a})^i}{256^l} \tag{6}$$

The existence of i different characters between S and S' creates at least $\lceil i/g \rceil$ different aggregated fingerprints. The bloom filter false positive occurs in CMBF when there are false positives for all of the considered aggregated fingerprints. On the other hand, according to relation (3), the false positive rate of each bloom filter is a, which is less than one. Thus, the upper bound of the bloom filter false positive in this case is:

$$a^{\lceil \frac{i}{g} \rceil} \leq \sqrt[g]{a^i} \tag{7}$$

In addition, the probability that S and S' differ in i bytes can be calculated by:

$$\frac{\binom{l}{i} \times 255^i}{256^l} \tag{8}$$

As a result, the bloom filter false positive in the case where S and S' differ in i bytes is:

$$FP_{BF_{n,i}} < \frac{\binom{l}{i} \times (255 \times \sqrt[g]{a})^i}{256^l} \tag{9}$$

□

Given the above, the bloom filter false positive can be computed by the summation of $FP_{BF_{n,i}}$ over all possible values of i. Therefore:

$$\begin{aligned} FP_{BF_n} &= \sum_{i=1}^{l} FP_{BF_{n,i}} \\ &< \sum_{i=1}^{l} \frac{\binom{l}{i} \times (255 \times \sqrt[g]{a})^i}{256^l} \\ &= \frac{1}{256^l} \times \left(\left[\sum_{i=0}^{l} \binom{l}{i} \times (255 \times \sqrt[g]{a})^i \right] - 1 \right) \\ &= \frac{(1 + 255 \times \sqrt[g]{a})^l - 1}{256^l} \end{aligned} \tag{10}$$

■

B. Fingerprint False Positive

As noted earlier, when all aggregated fingerprints of noninserted string S are seen in the appropriate CMBF bloom filter, we mistakenly state that the string was seen before. More generally, this situation could happen in two distinct scenarios of fingerprint collision and fragmentation. In what follows each of the mentioned scenarios will be analyzed, and afterward the fingerprint false positive is computed.

1) Probability of Fingerprint Collision: Suppose that S and S' are two different strings. A fingerprint collision occurs when:

- The aggregated fingerprints of S and S' are equal.
- The corresponding aggregated fingerprints from S and S' are inserted in the same bloom filter. In other words, the first byte of each aggregated fingerprint is equal for S and

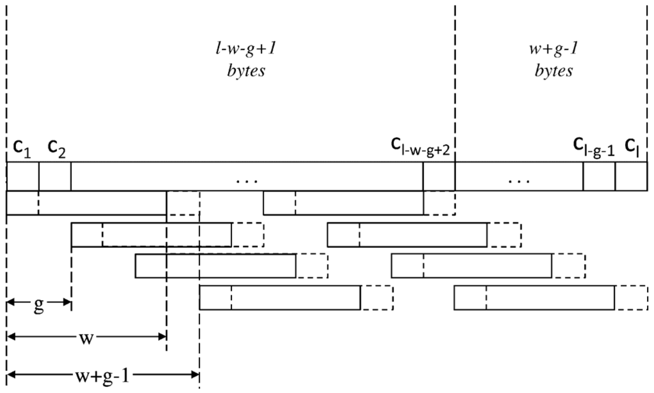


Fig. 2. Sample string could be divided into two parts. The break point is selected so that the characters in the second part create the final aggregated fingerprint for the string. Furthermore, characters used in creating each aggregated fingerprint are shown.

S', which results in selecting the same bloom filter from the set of 256 available bloom filters.

As an example, assume that $S = s_1s_2s_3s_4s_5s_6s_7$ and $S' = s'_1s'_2s'_3s'_4s'_5s'_6s'_7$, and that the window size and aggregation factor have been set to 4 and 2, respectively. The fingerprints collision occurs when:

- $s_1 = s'_1$ and $s_3 = s'_3$
- $[s_2] = [s'_2]$, $[s_4] = [s'_4]$, $[s_5] = [s'_5]$, $[s_6] = [s'_6]$, and $[s_7] = [s'_7]$

Lemma 3: Let S and S' be two random strings each with length l, and let the window length be denoted by w, the number of classes (fingerprint modulo) by q, and the aggregation factor by g. The following relation provides an upper bound on the fingerprint collision probability based on n insertions:

$$P_{C_n} < \left[\left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256}\right)^{g-1}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \times \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256}\right)^{w+g-2} - \frac{1}{256^l}\right] \times n \quad (11)$$

In order to prove lemma 3, one should first identify the characters which determine the proper bloom filter for inserting the aggregated fingerprints. The reason is that these characters have to be equal so that the aggregated fingerprints are inserted into the same bloom filter, where this is a precursor to a fingerprint collision. It is important to note that the characters at the end of the excerpt with which an aggregated fingerprint may not be calculated due to the minimum number of characters required for its calculation, should be ignored from the querying phase. In order to simplify the discussion, we assume that all characters of the query string would be used in the computation of the aggregated fingerprints.

Accordingly, the string could be divided into two parts, as shown in Fig. 2; one with length $l - w - g + 1$ bytes and the other $w + g - 1$ bytes. The characters in the second part create the final aggregated fingerprint, as a minimum of $w + g - 1$ characters are required to create an aggregated fingerprint.

In the first part, the bytes indexed by the multiples of g determine the proper bloom filter from the 256 possibilities for insertion, as shown in the example earlier in this subsection. Hence these characters in S and S' should be equal respectively. The total number of characters in S and S' which have to be equal is:

$$\left\lfloor\frac{l-w-g+1}{g}\right\rfloor \quad (12)$$

As a result, P_{f_1} is the probability that these corresponding characters are equal:

$$P_{f_1} = \left(\frac{1}{256}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \quad (13)$$

On the other hand, the other bytes in the first part are only involved in the aggregated fingerprints calculation, and do not effect the bloom filter selection, hence they should only fall with in the same class. This probability is denoted as P_{f_2}, which is calculated by:

$$\begin{aligned} P_{f_2} &= \left(\frac{\left\lceil\frac{256}{q}\right\rceil}{256}\right)^{(l-w-g+1)-\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \\ &< \left(\left(\frac{1}{q} + \frac{1}{256}\right)^{g-1}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \end{aligned} \quad (14)$$

Therefore, the probability of collision in the first part, P_f, is obtained by:

$$\begin{aligned} P_f &= P_{f_1} \times P_{f_2} \\ &< \left(\frac{1}{256}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \times \left(\left(\frac{1}{q} + \frac{1}{256}\right)^{g-1}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \\ &= \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256}\right)^{g-1}\right)^{\left\lfloor\frac{l-w-g+1}{g}\right\rfloor} \end{aligned} \quad (15)$$

As for the second part of the string, with a length of $w + g - 1$, the final aggregated fingerprint of S and S' must be equal. Accordingly, the first corresponding byte of these strings should be equal and the rest have to be in the same class. Therefore, we have:

$$\begin{aligned} P_s &= \frac{1}{256} \times \left(\frac{\left\lceil\frac{256}{q}\right\rceil}{256}\right)^{w+g-2} \\ &< \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256}\right)^{w+g-2} \end{aligned} \quad (16)$$

Consequently the probability that the aggregated fingerprints of S and S' are equal is:

$$P_{eq} = P_f \times P_s < \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \times \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \quad (17)$$

We should note that in case the aggregated fingerprints of two random strings are equal, then there will be a collision, except for the time when all the bytes in the string are truly equal. As a result, relation (17) is negated with $1/256^l$.

Furthermore, the fingerprint collision is directly related to the number of insertions in the bloom filters. In fact, as more aggregated fingerprints are inserted in the bloom filter set, the chance of collision with the next insertion increases. In lemma 3, we assume that the total number of inserted aggregated fingerprints is n, where there could be collisions among different insertions. In other words, the bloom filter set stores at most n different elements. Therefore, the upper bound on the probability of collision based on n insertions would be:

$$\begin{aligned} P_{C_n} &= \left[P_{eq} - \frac{1}{256^l} \right] \times n \\ &< \left[\left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \right. \\ &\quad \left. \times \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} - \frac{1}{256^l} \right] \times n \quad (18) \end{aligned}$$

□

2) Probability of Fragmentation: As noted earlier, the second contributing factor to the fingerprint false positive is fragmentation. In essence, fragmentation occurs when all fragments of a query string, with certain alignments, are found in the CMBF. From another perspective, when two or more inserted strings *merge*, and a query string is submitted to the system which consists of these two or more strings, then the system replies "yes" incorrectly to the query. We defined the merger operation as:

Definition 1: Let $S = S_1 S_2$ and $S' = S'_1 S'_2$ be two inserted strings, then $S'' = S''_1 C S''_2$ is the merge of S and S' if:

1) S collides with $S''_1 C$.
2) S' collides with $C S''_2$.

Note that definition 1 can be generalized using the merge operator, recursively (e.g. $merge(A, B, C) = merge(merge(A, B), C)$).

For example, and as illustrated in Fig. 3, assume that $S = a_1 a_2 a_3 a_4$ and $S' = b_3 b_4 b_5 b_6$ are two inserted strings into the CMBF, with $w = 3$ and $g = 2$. Accordingly, α and β are the aggregated fingerprints of S and S' respectively. Considering $S'' = merge(S, S') = c_1 c_2 c_3 c_4 c_5 c_6$, then if one queries for S'', the system responds with an answer of "yes" as the aggregated fingerprints for S'' would consist of α and β, resulting in a false positive.

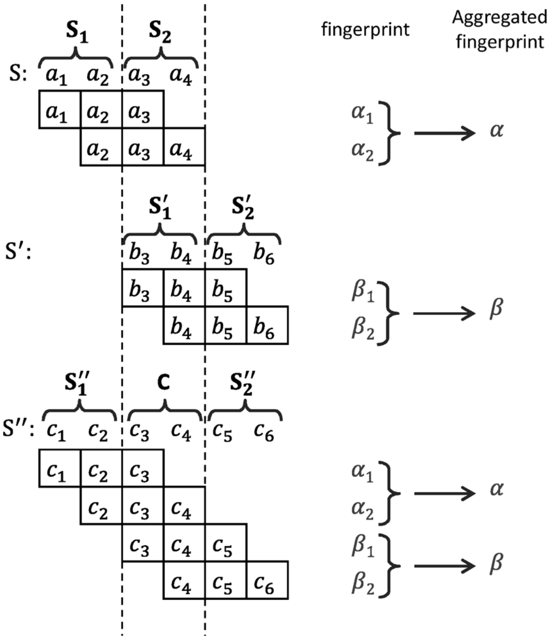


Fig. 3. Assume that only strings S and S' are inserted into the CMBF with $w = 3$ and $g = 2$. The string $S'' = c_1 c_2 c_3 c_4 c_5 c_6$ is constructed by $merge(S, S')$, where c_1 and c_3 should be equal to a_1 and b_3, respectively. Also all other bytes in S'' must be in the same class with the corresponding bytes in S and S'. Therefore, when a query is submitted for S'', a false positive occurs.

It is worth noting that, and as mentioned earlier, c_1 and c_3 should be equal to a_1 and b_3 respectively, in order for false positive to happen (because of the CMBF insertion procedure). Besides, all other bytes in S'' must be in the same class with the corresponding bytes in S and S'.

Lemma 4: Let $P_{frag_{n,i}}$ be the probability of i fragmentations occurring for string S with the length of l based on n insertions, where g is the aggregation factor, w the window size, and q the number of classes. Then:

$$\begin{aligned} P_{frag_{n,i}} &< n^{i+1} \times \binom{\left\lfloor \frac{l-w+1}{g} \right\rfloor - 1}{i} \\ &\quad \times \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \\ &\quad \times \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{i(w-1)} \quad (19) \end{aligned}$$

In order to prove lemma 4, first assume that the string S is constructed by merging $i + 1$ substrings. We can simplify the problem by dividing it into the following subproblems. Note that lemma 4 can be computed by the multiplication of probabilities for these items.

1) How many possibilities are there to select $i + 1$ sequence of contiguous aggregated fingerprints?[9]
2) How many possible points are there in the string at which i fragmentations could occur?
3) What is the probability of $i + 1$ substrings being merged, which results in i fragmentations?

At first, one should choose the sequence of contiguous aggregated fingerprints. As there are n insertions, then at most n

[9] Contiguous aggregated fingerprints are a set of aggregated fingerprints, which are in sequence and have originated from the same inserted string.

independent contiguous aggregated fingerprints would possibly be available. Therefore, one may choose $i+1$ aggregated fingerprints from the set of n possible aggregated fingerprints, which is stated by:

$$\binom{n}{i+1} < n^{i+1} \tag{20}$$

It has to be noticed that n^{i+1} is considered as an upper bound in order to simplify the proof of lemma 4.

The second item refers to the number of possible fragmentation points. The total number of aggregated fingerprints is $\lfloor (l-w+1)/g \rfloor$, and in turn the number of possible fragmentation points is $\lfloor (l-w+1)/g \rfloor - 1$, where l is the length of S. As i fragmentation points are to be selected, therefore:

$$\binom{\left\lfloor \frac{l-w+1}{g} \right\rfloor - 1}{i} \tag{21}$$

Finally, when it comes to computing the probability of $i+1$ fragments that are being merged, all the aggregated fingerprints of S are required to be seen in the bloom filters, where this probability can be computed by P_{eq}, relation (17). Furthermore, the merge operator requires that the second part of the first operand of the merge operator S_2, collides with the first part of the second operand S_1', as illustrated in Fig. 3, where each has a length of $w-1$. Therefore, it is sufficient that the $w-1$ corresponding bytes in the two fragments be in the same class, so that we have:

$$P_{w_i} = \left(\frac{\left\lceil \frac{256}{q} \right\rceil}{256} \right)^{i(w-1)} < \left(\frac{1}{q} + \frac{1}{256} \right)^{i(w-1)} \tag{22}$$

Accordingly, the following relation represents the probability of merging i + 1 substrings:

$$\begin{aligned} P_{eq} \times P_{w_i} < & \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \times \frac{1}{256} \\ & \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{i(w-1)} \end{aligned} \tag{23}$$

Consequently, the following relation expresses the fragmentation false positive, resulting from the occurrence of i fragmentations, with respect to n insertions:

$$\begin{aligned} P_{frag_{n,i}} < & \, n^{i+1} \times \binom{\left\lfloor \frac{l-w+1}{g} \right\rfloor - 1}{i} \\ & \times \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \\ & \times \frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{i(w-1)} \end{aligned} \tag{24}$$

□

Based on relations (11) and (19), one can compute the fingerprint false positive now.

Theorem 2: Let n be the number of insertions, q be the number of classes, w be the window length, l be the excerpt length, and g be the aggregation factor. Then the fingerprint false positive is upper bounded by:

$$\begin{aligned} FP_{F_n} < & \frac{1}{256} \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \\ & \times \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \\ & \times \left[1 + \left(n \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w-1} \right) \right]^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \times n \end{aligned} \tag{25}$$

Proof: Suppose that excerpt S, which was not inserted earlier into CMBF, is queried and a fingerprint false positive occurs due to the fingerprint collision, P_{C_n}, and fragmentation, $P_{frag_{n,i}}$. Therefore, the probability of fingerprint false positive can be computed by:

$$\begin{aligned} FP_{F_n} = & P_{C_n} + \sum_{i=1}^{\left\lfloor \frac{l-w+1}{g} \right\rfloor - 1} P_{frag_{n,i}} \\ < & n. \left[\left(\frac{1}{256} \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \right. \\ & \left. \times \frac{1}{256} \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \right] + \sum_{i=1}^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \\ & \times \left[\binom{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor}{i} . \left(\frac{1}{256} . \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \right. \\ & \left. \times \frac{1}{256} \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} . \left(\frac{1}{q} + \frac{1}{256} \right)^{i(w-1)} .n^{i+1} \right] \\ < & \frac{n}{256} \left(\frac{1}{q} + \frac{1}{256} \right)^{w+g-2} \\ & \times \left(\frac{1}{256} \times \left(\frac{1}{q} + \frac{1}{256} \right)^{g-1} \right)^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \\ & \times \left[1 + \left(n \times \left(\frac{1}{q} + \frac{1}{256} \right)^{w-1} \right) \right]^{\left\lfloor \frac{l-w-g+1}{g} \right\rfloor} \end{aligned} \tag{26}$$

■

As highlighted earlier in this section, the CMBF false positive originates in both bloom filter and fingerprint false positives, respectively denoted by FP_{BF_n} and FP_{F_n}, and can be computed by the summation of relations (5) and (25). A notable point in these relations is that the growth rate of the bloom filter false positive is much faster than that of the fingerprint false positive. More specifically, the number of insertions, n, appears as a multiplier in the FP_{F_n} relation, whereas the FP_{BF_n} is an exponential function of n. As a result, raising the number of insertions in CMBF leads to a much sharper increase in the FP_{BF_n} than the FP_{F_n}. To validate this observation, the two false positive rates were calculated using different values for n, as illustrated in Fig. 4, where FP_{BF_n} increases and approaches one sharply as the number of insertions increases. But FP_F rises steadily to a value near 10^{-80} for a similar value of n. As such one can ignore FP_{F_n}, and assume $FP_{CMBF} \simeq FP_{BF_n}$.

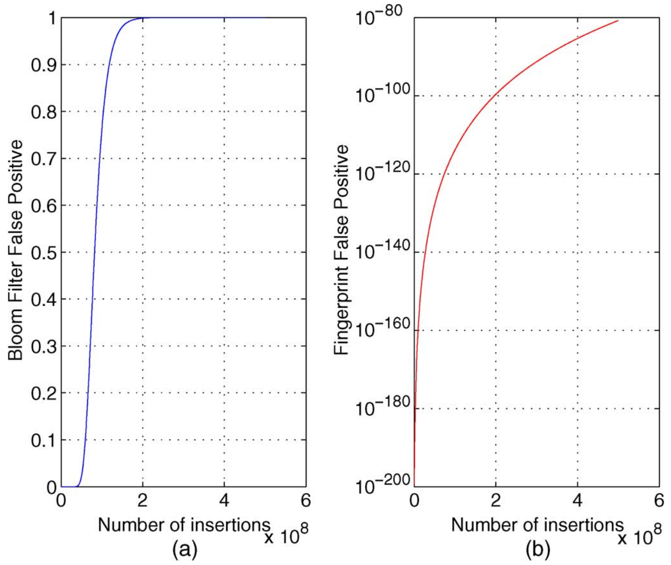


Fig. 4. (a) Bloom filter false positive of CMBF with different number of insertions between 50 000 to 500 000 000 characters, $w = 8$, $g = 3$, $q = 8$, $m = 25\,000$, and $l = 150$. (b) Fingerprint false positive of CMBF with the same noted parameters. As observed, the fingerprint false positive (b) is negligible with respect to the bloom filter false positive (a).

C. Optimization

Given the false positive formulation in the previous subsection, one could optimize the involved parameters in order to obtain the minimum false positive rate. As Mitzenmacher [16] shows, the number of hash functions used effects the false positive rate of the bloom filter. Increasing the number of hash functions results in an increase in the system's resolution and consequently decreases the false positive. On the other hand, using a large number of hash functions results in an increase in the number of insertions in the bloom filter, which increases the chances of collision and in turn the false positive rate.

In the proposed CMBF technique, the aggregation factor g plays the same role as the number of hash functions used for insertion in the bloom filter. Decreasing the aggregation factor results in increasing the number of insertions in the bloom filters and in turn brings about an increase in the false positive rate. Similarly, when the aggregation factor is increased, less resolution is used when processing a query, which results in a higher bloom filter false positive. To find the optimum value of g, with which the false positive rate of the CMBF is minimized, one should find the root of the false positive rate, as defined by relation (5), with respect to the aggregation factor as below (note that a is the single bloom filter false positive as specified in relation (3)):

$$\begin{aligned}
\frac{\partial FP_{CMBF}}{\partial g} &= \frac{\partial\left(\frac{(1+255\times\sqrt[g]{a})^{l}-1}{256^{l}}\right)}{\partial g} = 0\\
&\Longrightarrow \frac{l}{256^{l}} \times (1+255\times\sqrt[g]{a})^{l-1} \times \frac{\partial(1+255\times\sqrt[g]{a})}{\partial g} = 0\\
&\Longrightarrow \frac{\partial(1+255\times\sqrt[g]{a})}{\partial g} = 0\\
&\Longrightarrow \frac{\partial\left(\left(1-\left(1-\frac{1}{m}\right)^{\frac{n}{256g}}\right)^{\frac{1}{g}}\right)}{\partial g} = 0
\end{aligned} \tag{27}$$

Replacing variable g with $1/k$, we would be able to find the optimum value for g as below:

$$\begin{aligned}
&\Longrightarrow \frac{-k^{2}\times\partial\left(\left(1-\left(1-\frac{1}{m}\right)^{\frac{nk}{256}}\right)^{k}\right)}{\partial k} = 0\\
&\Longrightarrow \frac{\partial\left(\left(1-\left(1-\frac{1}{m}\right)^{\frac{nk}{256}}\right)^{k}\right)}{\partial k} = 0
\end{aligned} \tag{28}$$

Relation (28) is the single bloom filter false positive derivation with $n/256$ insertions, which was computed in [16] as:

$$k = \frac{256\times m}{n}\ln 2 \tag{29}$$

Therefore, the following relation computes the optimum value for the aggregation factor:

$$\Longrightarrow g = \frac{n}{256\times m\times\ln 2} \tag{30}$$

Having relation (30), the bloom filter length and the number of insertions for the expected false positive can be computed by:

$$m = \frac{-n\times\ln\left(\frac{\sqrt[l]{256^{l}\times FP_{CMBF}+1}-1}{255}\right)}{256\times(\ln 2)^{2}} \tag{31}$$

$$n = \frac{-256\times m\times(\ln 2)^{2}}{\ln\left(\frac{\sqrt[l]{256^{l}\times FP_{CMBF}+1}-1}{255}\right)} \tag{32}$$

where l is the excerpt length and FP_{CMBF} is the CMBF false positive.

Moreover, one could calculate the data reduction ratio for CMBF, using the optimum value for the aggregation factor. As noted earlier, 256 distinct bloom filters are employed, which occupy $256\times m$ bits of the storage to store n bytes. Thus, the CMBF data reduction ratio, in short DRR, is:

$$\begin{aligned}
DRR &= \frac{8\times n}{256\times m}\\
&= \frac{-8\times(\ln 2)^{2}}{\ln\left(\frac{\sqrt[l]{256^{l}\times FP_{CMBF}+1}-1}{255}\right)}
\end{aligned} \tag{33}$$

In the next section, an evaluation of the proposed CMBF technique will be offered, based on a number of experimental studies.

V. Experimental Evaluation

In this Section, we evaluate CMBF with the help of captured network traces. This evaluation serves a number of purposes. First, validity of the theoretical analysis conducted in the previous section is examined, second, the CMBF performance is studied when using wildcard queries, and last, CMBF is compared in terms of accuracy, data reduction, and performance to the previously proposed techniques in the literature. In fact, in addition to implementing CMBF, we have implemented the state-of-the-art techniques introduced by Ponec *et al.* [4], [5] which include FBS, VBS, WBS, and WMH, in order to make

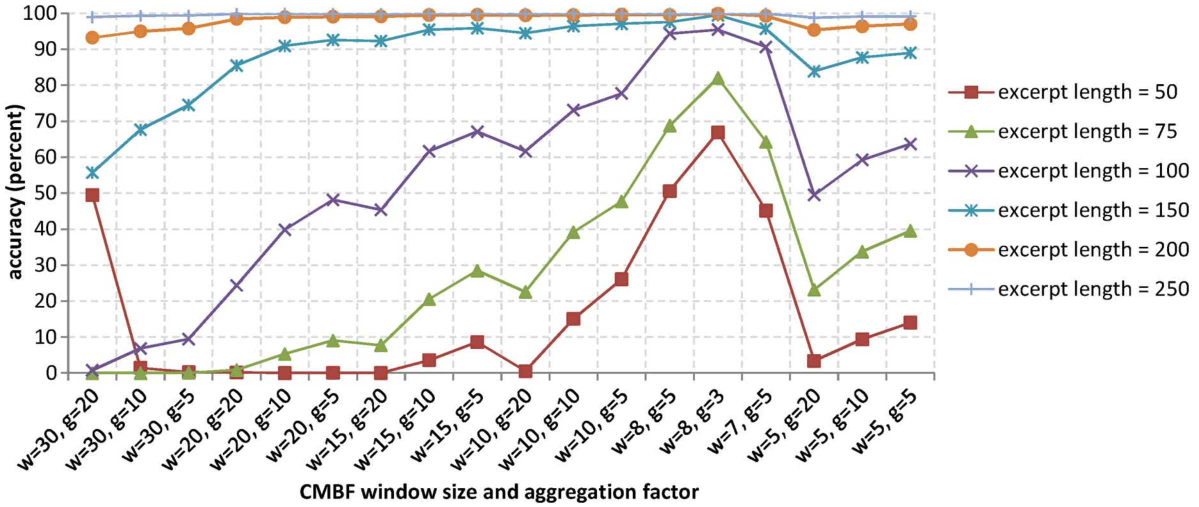


Fig. 5. Impact of the window size and aggregation factor parameters on the CMBF false positive (the fingerprint module is set to 8). The results were obtained by using 20 000 random queries for each possible excerpt length, where the excerpt lengths ranged between 50 to 250 byte, for a range of window size and aggregation factors noted on the figure.

a fair and accurate comparison between these different techniques. Furthermore, we have verified the correctness of our implementation by comparing the obtained results in our experiments with those reported by the authors in [4], [5]. As for the data set, 5 GBytes of network traces were collected from our Department's core switch, which consist of TCP and UDP packets. This traffic was then stored using the different attribution techniques noted.

A. False Positive Validation

In order to compare the experimental false positive rates with the theoretical limits calculated earlier, we first stored the captured network traffic into the CMBF using a data reduction ratio of 100 : 1. Subsequently 20000 different random strings were queried from the CMBF for each possible lengths of 25, 50, 75, 100, 150, 200, 250, and 300 bytes. As random strings were used, the probability that the strings actually were inserted into the data set (P_{sc}) is less than 10^{-46} for an excerpt of size 25 bytes, and this probability would be even smaller for larger excerpts. For more details on how to calculate this value, see Appendix A.

Using relation (5), the false positive rate was also computed, with excerpt sizes ranging from 25 to 300 Bytes. Both experimental and theoretical results are summarized in Table I, and as observed, the two sets of numbers closely follow each other for excerpts larger than 100 Bytes. However, given excerpts of size smaller than 100 Bytes experimental results indicate a much higher false positive rate. This can be attributed to the fact that, for small excerpt sizes, the number of constructed aggregated fingerprints is quite small, giving a rather poor query resolution, and consequently, a higher false positive rate.

It is also noticeable that the window size and aggregation factors have a major impact on accuracy value as illustrated in Fig. 5. In fact, with these two parameters, the CMBF accuracy could be worse than FBS or, contrarily, better than WMH. A more detailed comparison of these techniques is presented in Section V-C. Furthermore, the maximal CMBF accuracy rate is obtained when $w = 8$ and $g = 3$, and thus, these values are used in the experiments conducted in this study.

TABLE I
CMBF FALSE-POSITIVE EVALUATION

CMBF		
Excerpt Length	Theoretical Result	Experimental Result
25	43.15	87.4
50	18.62	33.1
75	8.03	17.91
100	3.47	4.56
150	0.65	0.51
200	0.12	0.07
250	0.02	0.01
300	0.004	0

The comparison between the theoretical false positive results obtained in Section IV, and the experimental results. Where in the experimental results 5 GBytes of network traces were inserted into the CMBF and then 20 000 random queries, for each excerpt length noted on the figure, were submitted to the CMBF and the resulting false positive calculated. The parameters used were $w = 8$, $g = 3$, and $q = 8$.

B. Supporting Wildcards

Another important feature of CMBF is its ability to respond to wildcard queries in a reasonable time. In order to evaluate the response time, we inserted a string in the system and consecutively used the same string to query the system, while replacing b random bytes of the excerpt with question marks (i.e. unknown characters). We measured the response time for different values of b. Moreover, this process was repeated for different values of q (fingerprint modulo), the results of which are shown in Fig. 6.

As an example, CMBF answers the excerpt with seven unknown characters (i.e. $b = 7$) in less than one second, using $q = 8$; however, it responds to the same query in about 600 seconds if we set $q = 16$. The reason is that for $q = 8$, about two million different strings are checked in the bloom filter set. Yet for $q = 16$, the search space increases to more than 200 million strings. Consequently, one can estimate that the previously proposed techniques respond to the same query in more than 4500 years, as they employ $q = 256$ as the fingerprint modulo. In the rest of this work we have set $q = 8$, as this is the largest q value for which we still obtain a reasonable response time for large number of unknown characters.

TABLE II
WRONG ANSWER DETAIL

Query	CMBF			FBS			VBS			WBS			WMH		
Length	Yes	N/A	No	Yes	N/A	No	Yes	N/A	No	Yes	N/A	No	Yes	N/A	No
25	17480	0	2520	0	20000	0	8001	8334	3665	0	20000	0	0	20000	0
50	6619	0	13381	19514	0	486	8908	1373	9719	5937	6067	7996	9800	3912	6288
75	3582	0	16418	19998	0	2	5978	181	13841	5343	0	14657	5412	0	14588
100	911	0	19089	18386	0	1614	3511	13	16476	2122	0	17878	2035	0	17965
150	101	0	19899	6121	0	13879	1248	0	18752	353	0	19647	323	0	19677
200	14	0	19986	884	0	19116	448	0	19552	51	0	19949	61	0	19939
250	2	0	19998	90	0	19910	174	0	19826	12	0	19988	4	0	19996
300	0	0	20000	14	0	19986	57	0	19943	1	0	19999	1	0	19999

The obtained responses when querying for 20 000 different excerpts using CMBF, FBS, VBS, WBS, and WMH. All the studied techniques store the data set with data reduction ratio 100 : 1 and they should answer "No" to all queries. A "Yes" response would results from a false positive, and an "N/A" response would mean that the system was unable to provide a response to the query (i.e., small query excerpt length). The experiment was carried out using eight different excerpt lengths, ranging between 25 to 300 bytes. The parameters used were $w = 8$, $g = 3$, and $q = 8$.

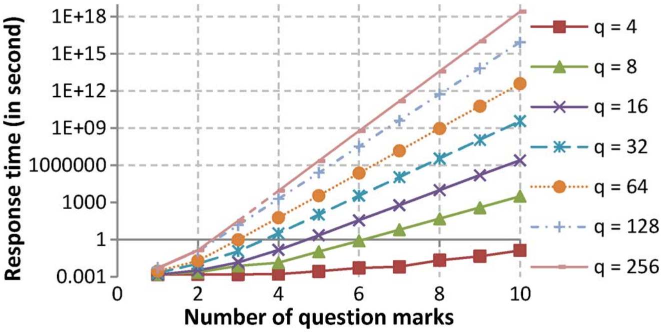


Fig. 6. CMBF response time to wildcard queries for different numbers of wildcard characters and different values of q, where the response time is shown in logarithmic scale. The parameters used were $w = 8$ and $g = 3$.

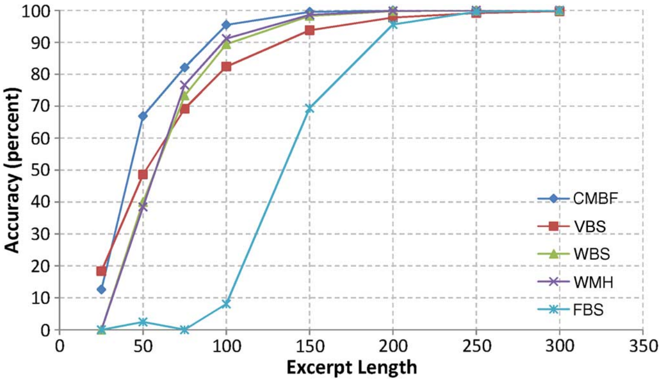


Fig. 7. True-positive rate of CMBF, FBS, VBS, WBS, and WMH techniques which is computed based on 20 000 random queries for each possible excerpt length, having a data reduction ratio 100 : 1. The parameters used were $w = 8$, $g = 3$, and $q = 8$.

C. Accuracy

As discussed earlier in this section, 20000 random query excerpts were used for each of the eight possible lengths, ranging from 25 to 300 bytes, to evaluate the false positive of CMBF. The same procedure was carried out for the FBS, VBS, WBS, and WMH techniques, the results of which are summarized in Table II. With all of the above mentioned techniques, the data reduction ratio is set to 100 : 1. As discussed before, with a probability practically equal to 1, the response to all 20000 queries should be an answer of "No". If a query results in an answer of "Yes", then a false positive has occurred. Alternatively, if the system is unable to respond to a query, it results in an answer of "N/A".

It is observable, that, CMBF is the only method with no instances of a "N/A" answer, while the other studied techniques are unable to handle small queries quite well. Even FBS, WBS, and WMH were unable to answer 25 byte query excerpts. These results could be explained by the fact that in the case of CMBF, given the chosen data reduction ratio, small values for the window size and aggregation factor could be selected, while in WBS and WMH, larger window sizes have to be selected. Additionally, as for VBS, due to its block boundary selection procedure, more than 8000 excerpts with the size of 25 bytes, were remained unanswered.

On the other hand, and unlike the other methods, CMBF is found to be capable of answering 300 byte queries with no false positive. As described in Section III, it uses 256 distinct bloom filters and only one hash function. Thus the aggregated fingerprints are distributed across bloom filters and only one bloom filter is affected with each insertion, which reduces the false positive rate. In contrast, FBS provides the worst rate. It creates fixed sized blocks, hence it suffers from the alignment problem. In fact, the best accuracy for the FBS, through our experiments, was achieved when a fixed block size of about 30 Bytes was used. Therefore, for a query excerpt, all the 30 possible alignments of the first blocks should be tested, which dramatically increases the false positive rate. Although the CMBF method suffers from the alignment problem as well, only all possible aggregated fingerprint positions are required to be checked, where this is 3 possible positions when CMBF operates optimally. This results in a less false positive rate for the CMBF method when compared to the FBS technique. Fig. 7 represents the calculated true positive rates for each of the studied methods based on 20000 different queries for each of the eight studied lengths.

D. Data Reduction Ratio

Another important characteristic of any payload attribution technique is its ability to reduce the volume of data stored. In order to compare CMBF to the other noted techniques, the accuracy rate (i.e. true positive rate) was set to 90%. Then we found the proper reduction ratio, given a query length and based on 20000 queries, where the query lengths ranged from 50 to 300 bytes. Fig. 8 depicts the detailed results.

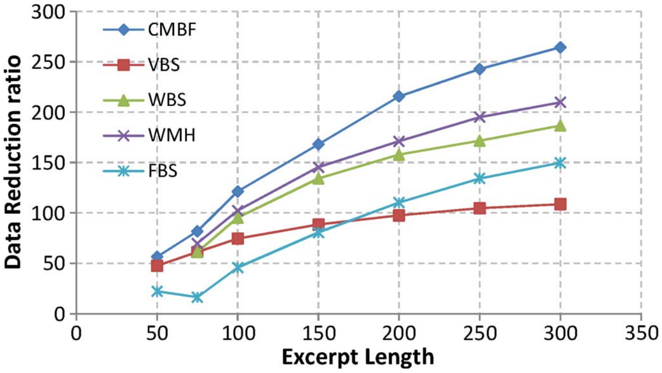


Fig. 8. Data reduction ratio for CMBF, VBS, WBS, and WMH methods when the true-positive rate has been fixed at 90%. The true positive was obtained by using 20 000 excerpts for each noted length. The parameters used were $w = 8$, $g = 3$, and $q = 8$.

TABLE III
INSERTION AND QUERYING COSTS

	CMBF	VBS	WBS	WMH	FBS
Inserting cost	n	n	$2n$	$2ni$	n
Querying cost	ng	n	$2n$	$2ni$	ns

The insertion and querying costs of CMBF, VBS, WBS, and WMH, where n is the string length, g is the CMBF aggregation factor, and i is the number of WBS instances employed in WMH, and s is the FBS window size.

According to Fig. 8, CMBF can store the data set in a very small space given the noted accuracy, while WMH, as the state-of-the-art technique falls noticeably shorter. For instance, WMH achieves a data reduction ratio of about 210 : 1, in the case that the excerpt length is 300 bytes, while CMBF achieves a reduction ratio of about 265 : 1 given the same excerpt length. It should be noted that due to memory space constraints, we were unable to evaluate the WBS and WMH techniques for 50 byte queries, requiring a 90% accuracy.

E. Performance

A payload attribution technique with a great accuracy and data reduction ratio will not be useful if it is not able to insert and respond to queries in a timely manner. This becomes important specially given the higher bandwidth links being deployed. If the system can not keep up with the data input rate in the insertion phase, then packets will be lost and this directly affects the accuracy of the system. Therefore, the system performance plays an essential role in the payload attribution systems.

Table III shows the insertion and querying costs of the listed methods in detail. It can be viewed that CMBF, FBS, and VBS have the least insertion costs as they process the string of characters only once. WBS uses the winnowing window technique to determine the boundaries, which forces the system to process the string twice. WMH uses i instances of WBS and therefore the cost of insertion will be i times more than WBS.

When responding to queries, VBS has to process the excerpt only once. On the other hand, WBS requires the excerpt to be processed twice and accordingly WMH requires i times the WBS processing on the excerpt. In the case of CMBF, though, the response time is a function of the string length as well as the aggregation factor g. This is as the first character used in aggregated fingerprint is unclear.[10] In other words, one should consider all possible alignments in order to find the starting point of the aggregated fingerprint calculation. In CMBF, g different alignments should be tested, where as in FBS, s different alignments had to be considered (s is the block size). Therefore, this problem is not as serious as in the FBS, since generally $g \ll s$.

TABLE IV
PROBABILITY THAT A RANDOM STRING WAS INSERTED INTO THE CMBF BLOOM FILTER SET PREVIOUSLY

Query Length (Byte)	P_{sc}
25	6.68×10^{-47}
50	4.16×10^{-107}
75	2.59×10^{-167}
100	1.61×10^{-227}
150	6.23×10^{-348}
200	2.41×10^{-468}
250	9.35×10^{-589}
300	3.62×10^{-709}

VI. CONCLUSION

In this paper we presented a new data structure for payload attribution, called Character Dependent Multi-Bloom Filters, which uses 256 bloom filters to store the full packet payload with small storage requirements. The main contribution of CMBF is its ability to respond to wildcard queries in a reasonable time. In fact doing exhaustive search in response to a wildcard query, which is done by prior techniques, is practically infeasible.

Moreover, a theoretical analysis of the proposed CMBF method was conducted and the validity of this analysis was ensured through a number of experiments based on actual network traffic. Additionally, comparisons were made between the proposed CMBF method and the state-of-the-art techniques previously proposed in the literature, in terms of their accuracy, data reduction ratio, and performance. It was shown that CMBF is able to achieve a data reduction ratio of about 265 : 1, while previously proposed techniques obtained a data reduction ratio of around 210 : 1 in the best case, assuming similar test scenarios. As part of the future work, we are working on applying the proposed CMBF data structure to other security applications such as signature based malware detection and intrusion detection systems to name a few examples.

APPENDIX A

The probability of a random string was previously inserted into the CMBF bloom filter set, can be calculated by the following relation:

$$P_{sc} = \frac{n \times \#queries}{256^l} \tag{34}$$

where l is the excerpt length and n is the number of inserted elements. In order to calculate an upper bound, it is assumed that each byte of data inserted in the CMBF results in an aggregated fingerprint insertion. Therefore, as 5 GBytes of data is used, there will be 5×2^{30} inserted elements in the CMBF. Table IV represents the calculated value of P_{sc} for different query lengths.

[10] As discussed earlier CMBF like FBS has the alignment problem.

ACKNOWLEDGMENT

The authors would like to thank F. Razmi, F. Kamaly, and Q. Emadi for their help in implementing some of the techniques studied in this work. They would also like to thank M. S. Dousti for his help in verifying some of the proofs in the manuscript. Furthermore, they would like to thank the reviewers for their constructive comments which has led to an improved version of the manuscript.

REFERENCES

[1] K. Shanmugasundaram, H. Brönnimann, and N. Memon, "Payload attribution via hierarchical bloom filters," in *Proc. 11th ACM Conf. Computer and Communications Security*, 2004, pp. 31–41, ACM.

[2] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[3] C. Cho, S. Lee, C. Tan, and Y. Tan, "Network forensics on packet fingerprints," *Security and Privacy in Dynamic Environments*, pp. 401–412, 2006.

[4] M. Ponec, P. Giura, H. Brönnimann, and J. Wein, "Highly efficient techniques for network forensics," in *Proc. 14th ACM Conf. Computer and Communications Security*, 2007, pp. 150–160, ACM.

[5] M. Ponec, P. Giura, J. Wein, and H. Brönnimann, "New payload attribution methods for network forensic investigations," *ACM Trans. Information and System Security (TISSEC)*, vol. 13, no. 2, pp. 1–32, 2010.

[6] E. S. Pilli, R. Joshi, and R. Niyogi, "Network forensic frameworks: Survey and research challenges," *Digital Investigation*, vol. 7, no. 1–2, pp. 14–27, 2010.

[7] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Proc. 2005 IEEE Symp. Security and Privacy*, 2005, pp. 226–241.

[8] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos, "An empirical study of real-world polymorphic code injection attacks," in *Proc. 2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'09)*, Boston, MA, USA, 2009.

[9] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic blending attacks," in *Proc. 15th Conf. USENIX Security Symp.*, Berkeley, CA, USA, 2006, vol. 15 [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267353, ser. USENIX-SS'06, USENIX Association

[10] Z. Li, M. Sanghi, Y. Chen, M. Y. Kao, and B. Chavez, "Hamsa: Fast signature generation for zero-day polymorphicworms with provable attack resilience," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2006, pp. 32–47 [Online]. Available: http://dx.doi.org/10.1109/S2006.18, ser. SP'06, IEEE Computer Society

[11] A. Snoeren, "Hash-based ip traceback," in *Proc. ACM SIGCOMM Computer Communication Rev.*, 2001, vol. 31, no. 4, pp. 3–14, ACM.

[12] O. Demir, P. Ji, and J. Kim, "Session based logging (sbl) for ip-traceback on network forensics," in *Proc. 2006 Int. Conf. Security and Management*, 2006, pp. 233–239.

[13] M. Rabin, Fingerprinting by Random Polynomials, Aiken Computation Laboratory, Univ., 1981.

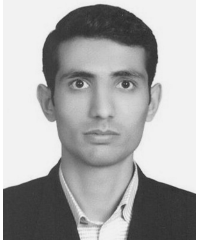
[14] S. Schleimer, D. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *Proc. 2003 ACM SIGMOD Int. Conf. Management of Data*, 2003, pp. 76–85, ACM.

[15] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," in *Proc. ACM SIGCOMM Computer Communication Rev.*, 1998, vol. 28, no. 4, pp. 254–265, ACM.

[16] M. Mitzenmacher, "Compressed Bloom filters," *IEEE/ACM Trans. Networking*, vol. 10, no. 5, pp. 604–612, Oct. 2002.

Mohammad Hashem Haghighat received the B.S. degree in computer engineering from Shiraz Azad University, Shiraz, Iran, in 2008, and the M.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2010.

He is currently a researcher with the DNS Laboratory at Sharif University of Technology. His research interests include network security, information forensics, and formal verification of security properties in protocols.



Mehdi Tavakoli received the B.Sc. degree in computer science, in 2004, from Shahid Beheshti University, Tehran, Iran, and the M.Sc. degree in computer science from Sharif University of Technology, Tehran, Iran, in 2012.

He is currently a researcher with the DNS Laboratory at Sharif University of Technology. His research interests include networking, network security, and digital forensics.

Mehdi Kharrazi received the B.E. degree in electrical engineering from the City College of New York and the M.S. and Ph.D. degrees in electrical engineering from the Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, New York, in 2002 and 2006, respectively.

He is currently an Assistant Professor with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. His current research interests include network and multimedia security.