On the Privacy of the Count-Min Sketch: Exploiting Positive Concentration

Silvia Pinilla Jiménez
Universidad Carlos III de Madrid
Puerta de Toledo, 2022
email: 100346880@alumnos.uc3m.es

Abstract— **Count-Min Sketch is a probabilistic data structure currently used by many applications to find out the number of occurrences of an element in a set, without the need to store all the elements. This is done by storing counters on the position of the hash of each element in a limited memory space, which saves resources and increases data processing speed. A well-known application is to store the number of packets of IP addresses of packets flowing through a network. Those can be later used to analyze which IP addresses generate the most traffic on the network, or to detect anomalies in the network, which may indicate a cyber-attack. One of the main disadvantages of Count-Min Sketch is the existence of false positives, but it can be seen as an advantage from a security point of view, as an attacker trying to exploit vulnerabilities will not have the certainty that the data obtained are true positives. In this paper, it is shown that an attacker can extract information about the elements stored on the table of the Count-Min Sketch performing a black-box analysis. The explanation of this comes from the observation of the elements being concentrated in some specific parts of the universe where the count estimates is high, which we called positive concentration. The Count-Min Sketch application of estimating the number of packets for IPv4 addresses of packets traversing a network has been considered as the case study of this work.**

Keywords— privacy, probabilistic data structures, Count-Min Sketch, positive concentration.

I. Introduction

The amount of data that is processed every day is huge, and as time goes by it is growing exponentially. New technologies make this data grow even faster. Therefore algorithms, filters and data structures that allow us to obtain certain characteristics or data from a dataset are increasingly necessary. For example, it can be quite useful to know if a certain IP address is in a set of received packets, or if a malicious URL is in a set of accessed pages, or if the number of connections to a specific host is higher than usual [1]. This information can be used for example to determine the risk of a cyber-attack [2]. However, this requires storing all packets, URLs, or connections for later parsing and processing, which requires a large amount of space. In addition, this leads to other problems such as slower processing speed and higher costs.

Data structures and data sketches techniques provide means to drastically reduce this size, allowing for real-time or interactive data analysis with reduced costs but with approximate answers. There are different types of data structures (array, stack, queue, linked list, etc.) and different data types (boolean, integer, string, pointers, etc.) [3].

A well-known data structure is the Bloom filter [4]. A Bloom filter is a space-efficient probabilistic data structure, that is used to test whether an element is present on a dataset. A Bloom filter has the particularity that false positive are possible, but false negatives are not possible. This means that if we search for a certain element in a dataset and the Bloom filter does not find that element, we can be sure that this element is not really in the dataset. However, if the Bloom filter finds such an element, there is no guarantee that this element is in the set, it could be a false positive. The algorithm behind this data structure consists of an array of *m* bits, initialized to 0, and a set of *k* hash functions, which, given a piece of data, will generate a number between 0 and *m*-1. The hash obtained in each hash function will be the position in the array where the value will be changed to 1. For example, given three words *x, y* and *z* inserted on a Bloom filter with 3 hash functions, the bit array will be modified as shown in Figure 1.

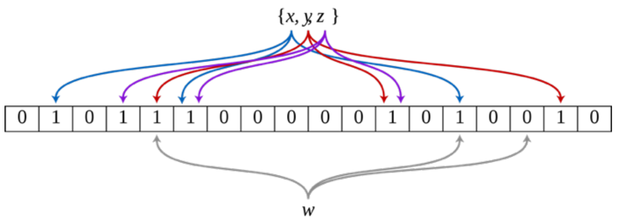

Figure 1. Bloom filter with 3 hash functions {x, y, z}.

Another important data structure that allows us, in this case, to count the number of times an element appears in a dataset is the Count-Min Sketch [5]. Count-Min Sketch is a probabilistic data structure that provides a summary of the number of item occurrences that have taken place in the processed data. Because of its simplicity, Count-Min Sketch has many use cases in networking, databases, and multiple other fields. Its algorithm is very similar to the Bloom filter algorithm, but instead of having a bit array, it has a table of integers.

Cybersecurity is becoming increasingly important in all areas of technology, including privacy [6][7]. However, attackers will find ways to perpetrate it, so it is crucial to study the potential vulnerabilities of each data structure and know what problems we face. In this paper, Count-Min Sketch, its privacy, and positive concentration are going to be studied. Some tests will be performed with the aim of compromising the security of the algorithm and finding patterns that allow to infer the existence of elements within a dataset.

The rest of the document is organized as follows. Section II introduces the Count-Min Sketch, how it works, and mentions previous work on other data structures where it is shown that privacy can be compromised. Section III explains what positive concentration is and how it could be studied in the Count-Min Sketch. Section IV explains how the Count-Min Sketch algorithm has been implemented in Java and which parameters and input data are used for this investigation. Section V explains the proposed adversarial model for attacking positive concentration. The results of the attack are shown and discussed in section VI. Finally, in the last section, advances for future work are proposed and a conclusion is made.

II. Count-Min Sketch

The Count-Min Sketch, also known as CMS, is a probabilistic data structure that counts the number of times an element appears in a dataset, and it does that without storing all the data from the dataset. The algorithm is quite similar to the Bloom filter, but Count-Min Sketch uses a table of integers instead of a bit array [5].

The CMS data structure consists of a table of width *w* and depth *d*. The width of the table represents the memory cells, and the depth represents the number of hash functions used, which must be sufficient to ensure that the probability of two different elements having the same hashes (probability of collision) is as low as possible. Both parameters are configured by the user and are directly related to the space and memory required. These two parameters must be set while maintaining a compromise between the memory and resources used and the efficiency of the data structure.The CMS data structure can be seen in Figure 2.

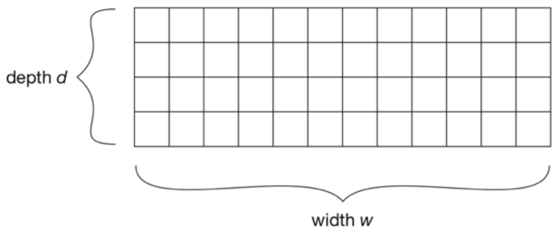

Figure 2. CMS table of depth *d* and width *w* [8].

After defining the dimensions of the table, the following operations are performed:

1) The table is initialized with all its elements to 0.

$$\forall i \in \{1, \dots, w\}, \forall j \in \{1, \dots, d\} : count[i, j] = 0 \qquad (1)$$

2) When an element *a* is inserted, the hashes of all hash functions are calculated. Then, 1 is added to the number on the position in the table indicating that hash. An example of this operation is shown in Figure 3.

$$\forall i \in \{1, \dots, d\} : count\,[i, h_i(a)] = +1 \qquad (2)$$

3) To retrieve the number of occurrences of a specified element *a*, all hashes are calculated and the smallest number appearing in these positions will be returned.

$$min_{i=1}^{d} = count[i, h_i(a)] \qquad (3)$$

As any probabilistic data structure, the counts retrieved from a CMS are approximate; however, there are some theoretical guarantees on error bounds. As Bloom filters, false negatives are not possible in Count-Min Sketch, so the first guarantee is that count cannot be underestimated. Errors are only introduced when cell values are incremented too much because of hash conflicts, so in case of error, the count will be overestimated. To determine how much the CMS overestimates the count, two variables are used: ε and δ. The theorem, where ||count|| is the sum of all counts stored in the data structure, says that:

"With a probability of $1-\delta$, the error is at most $\epsilon * ||count||$. Concrete values for these error bounds ϵ and δ can be freely chosen by setting $w=\lceil e/\epsilon \rceil$ and $d=\lceil ln(1/\delta) \rceil$."

So, parameters *w* and *d* can be configured so that errors are of reasonable magnitude for the application at hand [8]. The second guarantee is that, due to false positives, an attacker cannot generally infer the elements stored in the data structure itself. However, there are some settings in which an attacker can infer information about the elements stored. Previous works have shown that for Bloom filters, an attacker can infer the presence of some elements when the universe is small, and the attacker has access to the data structure type and implementation [9]. This shows that privacy of filters may be compromised, which may indicate that this type of attack is also possible in other types of data structures, such as the CMS.

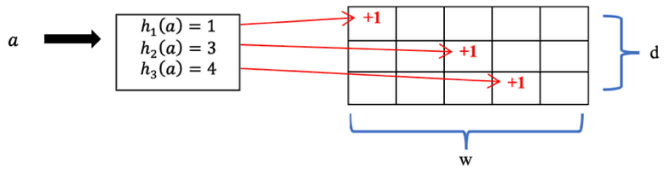

Figure 3. Inserting an element '*a*' on a CMS with 3 hash functions.

III. Positive Concentration

Positive concentration is the tendency of the elements to be concentrated in one or more parts of the whole dataset [10]. This means that the elements inserted on the data structure are not randomly distributed. Instead, false positives are randomly distributed and are not concentrated. In many applications, the universe is naturally divided in many parts, so calculating positive concentration is easier and faster.

In the following, when applying positive concentration to the CMS, we will assume that the elements with highest count estimates will be grouped in one or more zones of the universe. This can be explained with the count estimates obtained after querying the CMS. When the count obtained after querying the CMS with a random element is high, the probability of this element of being a true positive increases. On the other hand, if the count obtained is a small value, the probability of that element of being a false positive increase.

Thus, true positives are concentrated in high values of the count estimates. This is shown in Figure 4, where parts A, B, and C are parts of the CMS table with high values of count estimates. In these parts true positives are concentrated because of the high values of count stored on the CMS table.

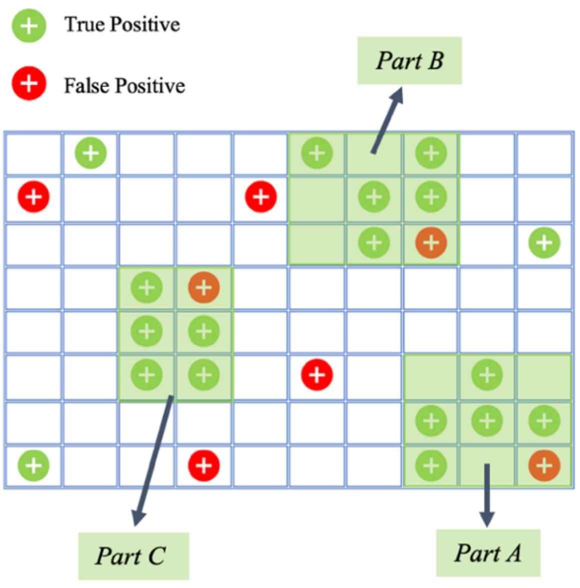

Figure 4. Positive concentration on parts A, B, and C of the CMS.

Positive concentration can be measured by querying the CMS to retrieve the count estimates of all possible elements, and obtain the elements with the highest values of count. With this data, it is possible to estimate true positives. As the CMS stores different values of count that vary depending on the application, it is needed to establish a criterion for considering a positive as a true positive or as a false positive. This criterion must be based on all the values of the CMS table, identifying the tendency of the counts, and establishing what is a high number of counts, from which to tell whether a positive is a true positive. So, a threshold must be defined so that, if after a query to the CMS, the count estimates of that value is higher than the threshold, the element can be considered a true positive.

In this way, positive concentration can be defined as the number of positives obtained on part A, considering positives as the elements with a count estimate over the threshold, divided by the number of true positives on part A, considering true positives as the elements of the original data that with a count over the threshold. This is defined in equation 4.

$$Positive\ Concentration\ |_{part\ A} = \frac{\#\,positives\ |_{part\ A}}{\#\,true\ positives\ |_{part\ A}} \quad (4)$$

One application of the CMS consists of storing IP addresses of incoming or outgoing packets on a network. The IP address is obtained from the packet header and inserted into the CMS. In this way, IP addresses that generate the most traffic on the network can be identified with a query to the CMS by looking for the highest number of count estimates. This application is the one considered on this paper to analyze positive concentration on the CMS.

IV. Count-Min Sketch Implementation

This section explains the code used for the CMS implementation and the IPv4 traces used as input data. The implementation has been programmed on Java language using Eclipse and it is available on GitHub [11]. Some parts of code are based on the CMS implementation programmed by Jorge Martínez in C++ [12].

A. Code

For the CMS implementation, the program requires three input values: width *w*, depth *d*, and a file with the input data, in our case, IPv4 packet traces. In this case, values of *w* and *d* chosen guarantee that, with a probability of 81.7%, the error is at most 0.000166*||count||, as described in equation 5.

$$\begin{cases} d=4=ln\frac{1}{\delta} \rightarrow \delta=0.183 \\ w=16384=\frac{e}{\varepsilon} \rightarrow \varepsilon=1.66\times10^{-4} \end{cases} \quad (5)$$

After setting these values, each element of the input file is inserted on the CMS. The hash functions used are Robert Sedgwicks and Justin Sobel hash functions, known as RS Hash and JS Hash [13], [14]. As there could be as many hash functions as the user indicates on the *d* value, and to keep the consistency of the algorithm, the hashes for each value are calculated as a linear combination of the two hash functions mentioned. Then, the position of the table with column equal to *d* and row equal to the hash calculated is located, and a 1 is added to that value. This is detailed on Figure 5.

```
for (int i = 0; i < depth; i++) {
    hash = (a*RSHash*value + b*JSHash*value + value) mod w;
    CMS[i][hash] +=1;
}
```

Figure 5. Calculating hash for each element and incrementing the count.

Also, a class to query the CMS has been implemented. This class performs the reverse operation: given an input element, it calculates the hashes and looks for the values on those positions. The minimum value of the ones stored on the *d* positions is returned.

B. *IPv4 traces*

The input data used for this investigation are real packet traces obtained from the Center for Applied Internet Data Analysis (CAIDA) [15]. These traces contain the header values, in more detail the 5-tuple formed by the source and destination IP addresses and transport ports as well as the protocol field of each packet. From that information, the number of packets of each source address have been computed and written in file with two columns: the frequency of an IP in a network and the address of that IPv4. A small fragment of one of the files used is shown detailed in Figure 6.

To insert the data into the CMS, a loop has been created that enters the IP address indicated in the second column as many times as the frequency in the first column indicates.

Frequency	IP
1344800	77.141.180.230
1315374	77.141.177.6
909986	186.75.214.134
859418	77.141.160.155
462376	186.75.212.243
216386	186.75.209.126
93450	201.96.35.138
89658	128.122.155.95
57411	189.156.197.98
53868	201.89.210.83
48853	48.116.70.254
…	…

Figure 6. Example of the traces used on this investigation.

V. Exploiting Positive Concentration

In this section the adversarial model considered for the attacker is presented, detailing all the steps and the analysis that the attacker can perform.

A. *Adversarial Model*

The adversarial model is based on the following assumptions:

- The attacker has not access to the CMS implementation or to the hash functions used.
- The attacker does not know the values *w* and *d*.
- The attacker does not know the content of the table of the CMS.
- But the attacker can perform as many queries as he/she wants.

B. *Proposed Attack*

In this way, consider a CMS that stores IPv4 addresses of packets traversing the network. An attacker can know all the IPv4 addresses on the table and its count, by querying the CMS with the entire IPv4 universe, taking into account that some of them may be false positives. Therefore, the attacker must establish a criterion for considering whether a positive is a true positive. A first approach can be:

1. Iterate all the IPv4 universe and save the count for every address.
2. Define a threshold from which consider a positive as a true positive.
3. Iterate again the whole IPv4 universe but saving only the IPv4 addresses with a count higher than the threshold.
4. The resulting IPv4 addresses can be considered by the attacker as true positives, and as addresses generating the most traffic on that network.

C. *Performing the Attack*

Several Java classes have been implemented to perform this attack [6]. The implementation consists of making queries to the CMS, asking for all the addresses in the IPv4 universe and saving the results in a file. This file will be subjected to filtering, where only those addresses that exceed a threshold of occurrences will be saved. Therefore, it is important to define a good threshold, achieving a trade-off between number of IP addresses and percentage of true positives. The higher the threshold, the more likely it is that the IP addresses are true positives, but the fewer IP addresses the attacker will obtain.

The attacker can establish this threshold by trial and error: testing several values until results can be considered valid. A good approximation could be using as a threshold the average number of occurrences of all IP addresses. The average can be obtained by the attacker since the entire IPv4 universe, and their corresponding occurrences have been consulted on the CMS.

A threshold has been defined after testing several values, so it may vary between different input elements and CMS applications. It has been set to 10 times[1] the average count of all IPv4 addresses on the CMS. Then, the attacker can filter all the IPv4 addresses obtained keeping only those exceeding the threshold. These addresses are the ones considered as true positives by the attacker.

D. Evaluation

To check if the addresses obtained by the attacker are really the correct ones and what the attacker would have gained from his results, the addresses have been compared with the real data. To determine positive concentration on elements of the CMS, addresses and count estimates obtained by the attacker have been compared with real addresses and real counts, calculating the positive concentration as explained on equation 4. The evaluation has been performed to determine the probability of a random IPv4 address being a true positive based on the count estimates obtained after querying the CMS. In general terms:

- For each of the addresses considered by the attacker as positives, a search has been performed on the actual data. If the true number of occurrences of the IP address exceeds the threshold, the address obtained by the attacker is considered true positive.
- If the true number of occurrences of the IP does not exceed the threshold, the address obtained by the attacker is considered false positive.

To study the probability of an IPv4 address being a true positive, addresses with the same number of count estimates have been evaluated together. This analysis has been carried out as follows:

- Each IP will be classified as true or false positive, depending on whether the true value of counts exceeds the threshold, as explained before. In this way, the number of positives with the same number of count estimates obtained by the attacker and the number of true positives is obtained.
- Then, the probability of true positive for all addresses with the same number of counts is calculated. This is done by dividing the number of positives obtained with the same number of count estimates by the number of true positives with the same count.

In this way, we can evaluate how true the data obtained by the attacker is and how the probability of true positive varies depending on the number of the count estimates.

Results show that, the higher the number of occurrences of an IP address, the higher the probability of true positive. This is logical: the more occurrences an IP address has, the more likely it is to truly generate a lot of traffic on the network. Therefore, the fewer occurrences it has, the less likely it is to be a true positive.

VI. Results

Several experiments have been carried out to demonstrate the positive concentration in the CMS using different packet traces. Specifically, three real packet traces have been used [15]. The traces correspond to one minute of traffic containing millions of packets and the number of packets per flow is highly skewed [2].

After entering the traces into the CMS and querying the universe of IPv4 addresses, a high number of positives has been obtained compared to the actual entries. This is expected as the CMS has limited space and the total number of IPv4 addresses is more than 4 billion. Based on the content of the CMS, it is possible to observe how many values per column *d* exceed the threshold, and it is possible to calculate the probability that, given a random IP, it exceeds that threshold. This calculation can be done following equation 6, where *i* is the count obtained after querying the CMS with a random IPv4 address, and *N* is the number of values per column exceeding the threshold.

$$P\ (i > threshold) = \left(\frac{N}{w}\right)^d \tag{6}$$

As the number of IPv4 addresses is 2^{32}, it is possible to know approximately how many IP addresses the attacker would get as false positives after querying the IPv4 universe.

For example, analyzing results from trace 1, that contains 208.464 entries, and using values discussed in Section IV:

$$\begin{cases} w = 16384 \qquad d = 4 \\ N = 2251 \ \rightarrow P\ (\ i > threshold) \approx 0.03563\% \\ 2^{32}\ \times\ 0.0003563\ \approx 1.530.296\ addresses \end{cases} \tag{7}$$

After querying the CMS with the IPv4 universe, 1.651.171 addresses have been obtained. Notice that, the number expected of false positives is the one obtained on equation 7 plus the number of entries of the CMS, which makes a total of 1.738.760 addresses. So, the number expected is more than 8 times the number of IPv4 addresses introduced on the CMS.

[1] Threshold has been defined considering that more than a half of IP addresses have only one occurrence, so that the average decreases significantly.

[2] Traces correspond to CAIDA 2014 dataset [10] and are equinix-chicago.dirA.20140619- 130900 (Trace 1), equinix-chicago.dirB.20140619-132600 (Trace 2), and equinix- sanjose.dirA.20140320-130400 (Trace 3).

With the IP addresses and counts obtained, the probability of true positive for all IP addresses with the same number of counts has been calculated, giving the results for each packet trace shown in Figures from 7 to 12. Note that, as the graphics have been derived from the results, the first count value observed is approximately the one corresponding to the threshold of 10 times the average.

For all traces we can see that even having chosen a threshold apparently high, we still have a high probability of false positives for all traces in the first values. It is from about 20 times the average, that the probability of true positive rises to 90%.

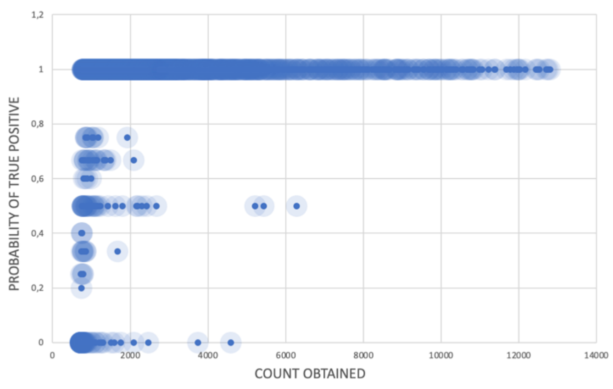

Figure 7. Positive concentration on Trace 1 when using values of the CMS w=16384 and d=4.

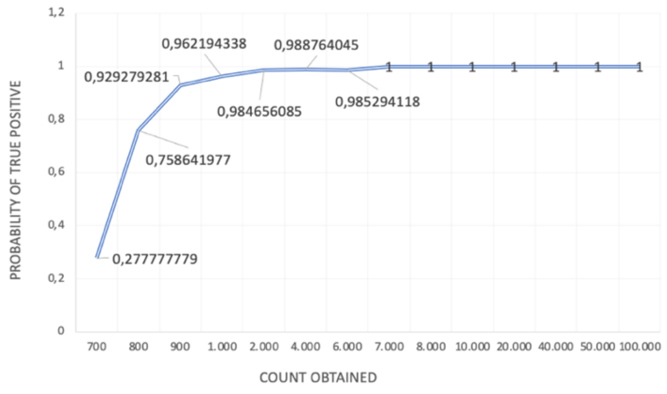

Figure 8. Average positive concentration on Trace 1 when using values of the CMS w=16384 and d=4.

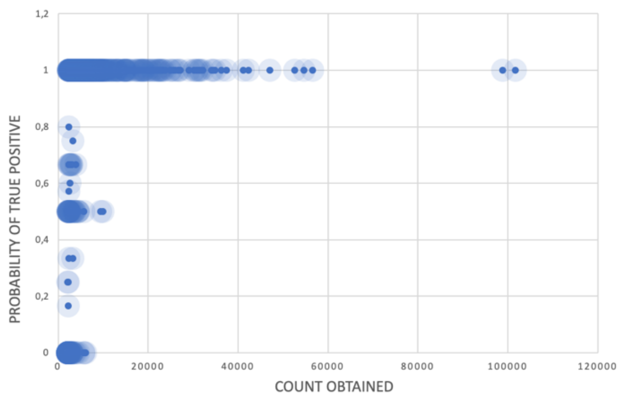

Figure 9. Positive concentration on Trace 2 when using values of the CMS w=16384 and d=4.

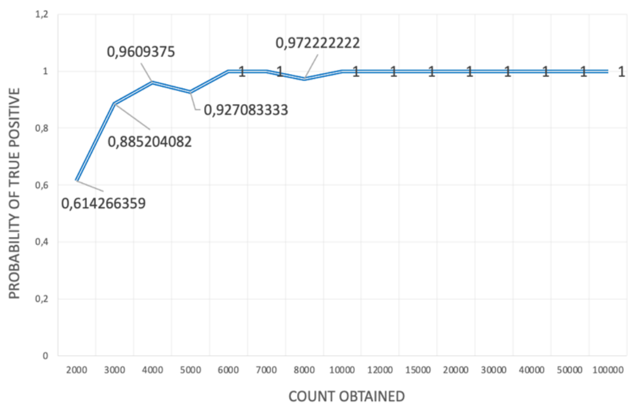

Figure 10. Average positive concentration on Trace 2 when using values of the CMS w=16384 and d=4.

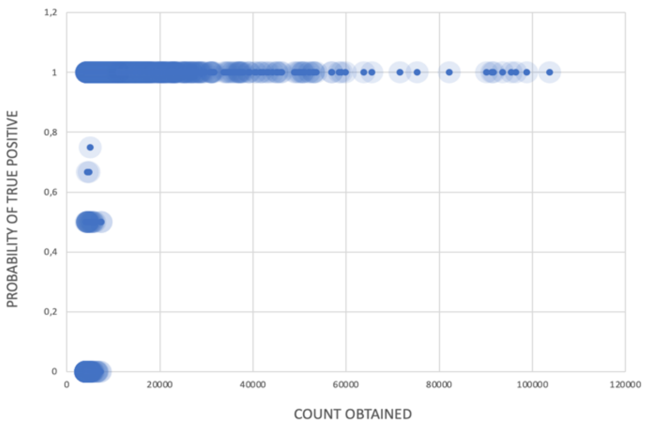

Figure 11. Positive concentration on Trace 3 when using values of the CMS w=16384 and d=4.

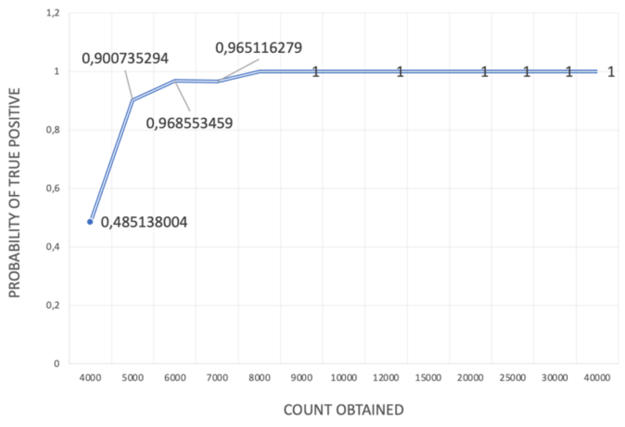

Figure 12. Average positive concentration on Trace 3 when using values of the CMS w=16384 and d=4.

As the results may vary depending on the width and depth of the CMS, results have been obtained for different combinations of CMS dimensions. The results show that when the depth value is increased, the positive concentration increases considerably, being almost 100% at values close to the set threshold, but when the width value is increased, the positive concentration increases only slightly. In following Figures, results of calculating positive concentration with the CMS values specified are shown.

The values of width and depth tested for the CMS are:

- *width* = 16384, *depth* = 3
- *width* = 16384, *depth* = 5
- *width* = 8192, *depth* = 3
- *width* = 8192, *depth* = 5

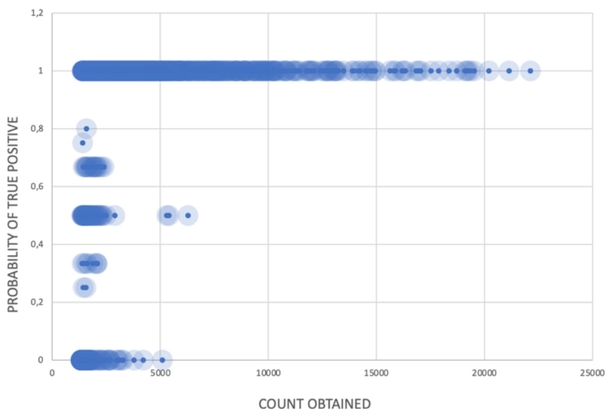

Figure 13. Positive concentration on Trace 1 when using values of the CMS w=16384 and d=3.

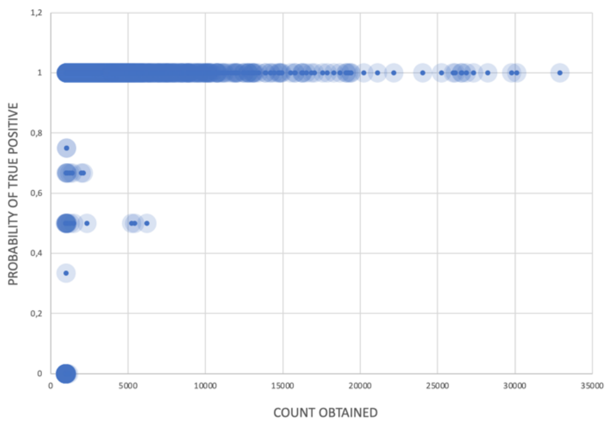

Figure 14. Positive concentration on Trace 1 when using values of the CMS w=16384 and d=5.

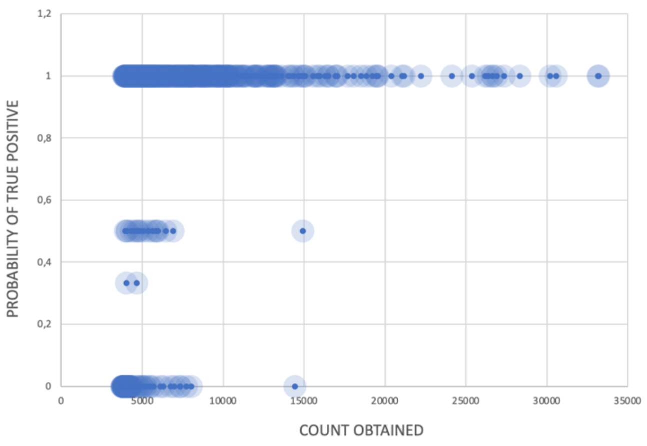

Figure 15. Positive concentration on Trace 1 when using values of the CMS w=8192 and d=3.

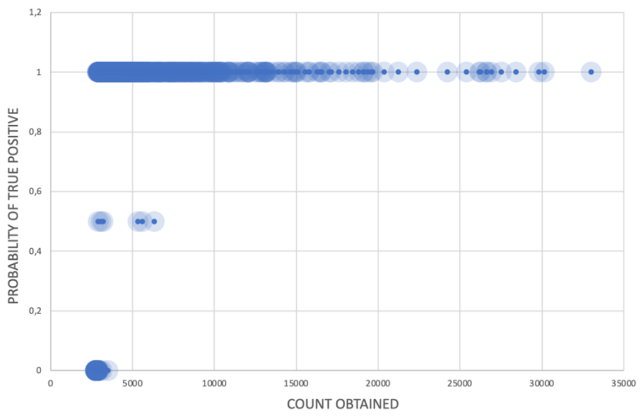

Figure 16. Positive concentration on Trace 1 when using values of the CMS w=8192 and d=5.

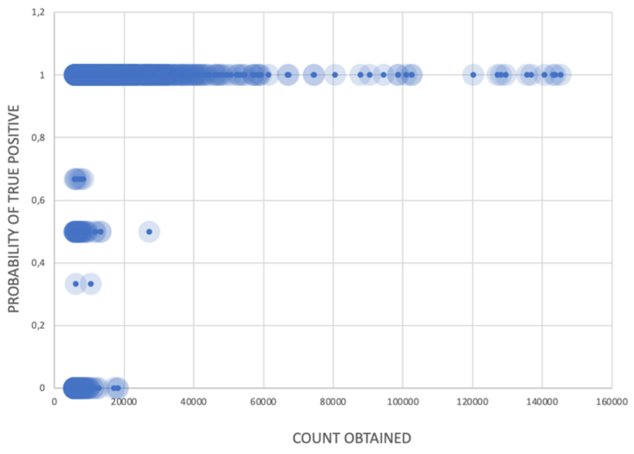

Figure 17. Positive concentration on Trace 2 when using values of the CMS w=16384 and d=3.

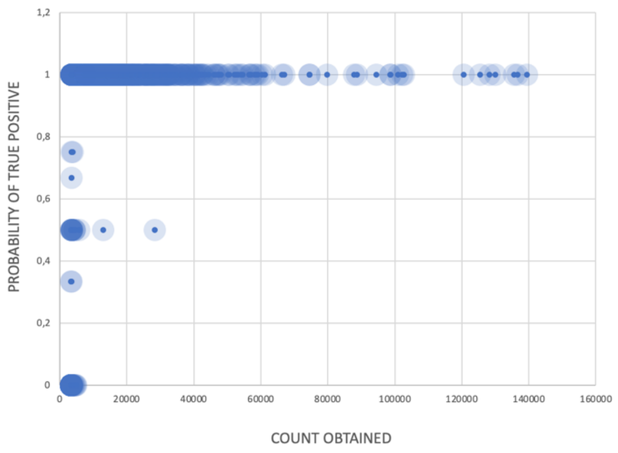

Figure 18. Positive concentration on Trace 2 when using values of the CMS w=16384 and d=5.

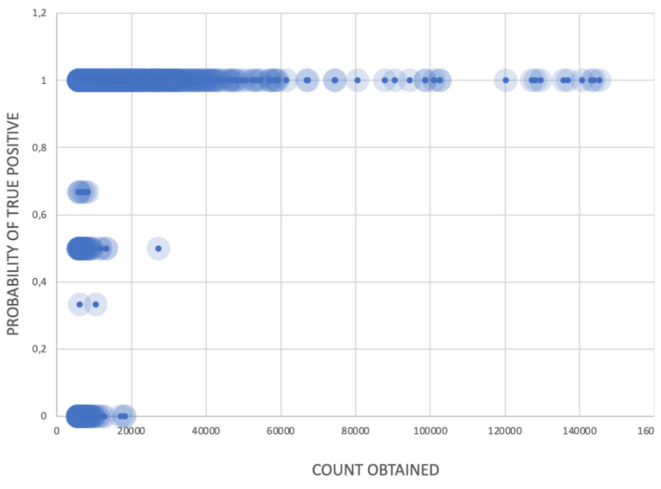

Figure 19. Positive concentration on Trace 2 when using values of the CMS w=8192 and d=3.

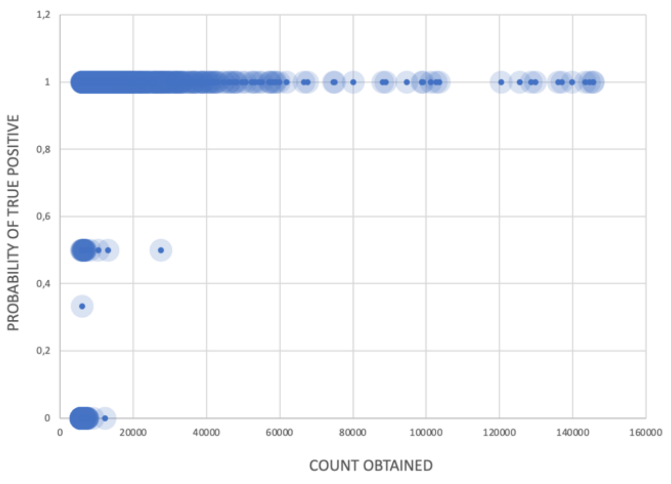

Figure 20. Positive concentration on Trace 2 when using values of the CMS w=8192 and d=5.

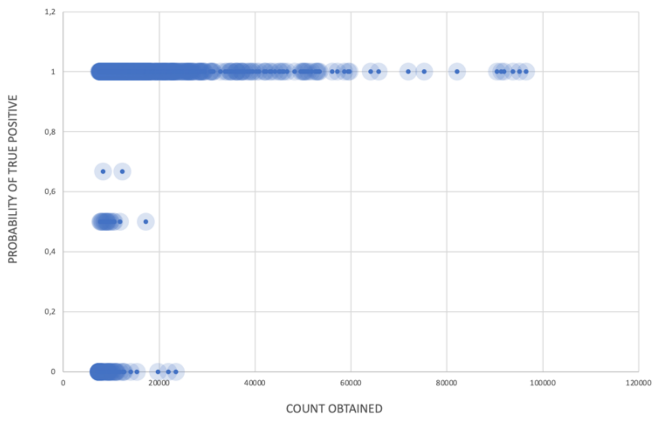

Figure 21. Positive concentration on Trace 3 when using values of the CMS w=16384 and d=3.

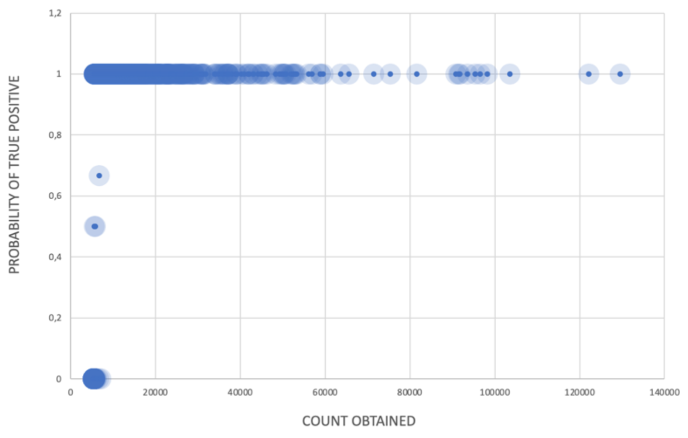

Figure 22. Positive concentration on Trace 3 when using values of the CMS w=16384 and d=5.

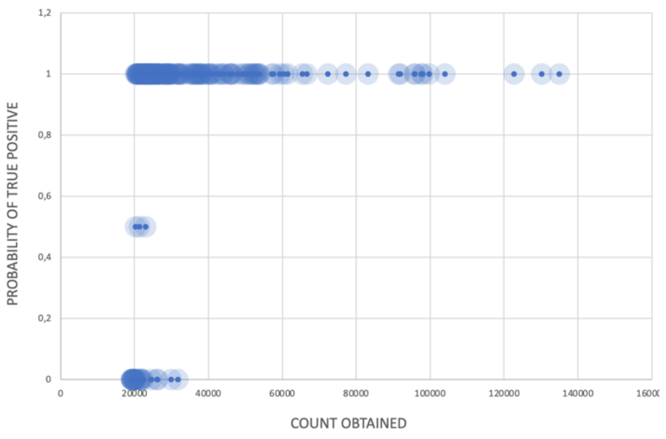

Figure 23. Positive concentration on Trace 3 when using values of the CMS w=8192 and d=3.

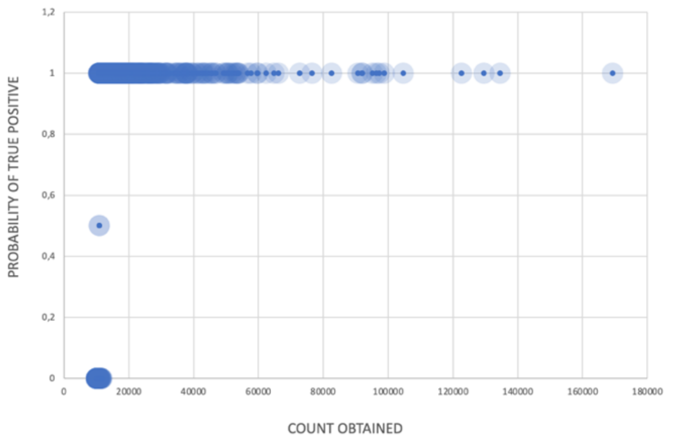

Figure 24. Positive concentration on Trace 3 when using values of the CMS w=8192 and d=5.

VII. Conclusion and Future Work

In this paper, positive concentration in the probabilistic data structure of the Count-Min Sketch has been evaluated. A CMS that stores the IPv4 addresses of packets circulating in a network has been considered, taking three real packet traces to evaluate them independently.

The positive concentration has been defined as the ratio between the positives obtained in an area of the universe and the true positives, considering positive as that value of the CMS estimator that exceeds a threshold, in this case, 10 times the mean. Positive concentration therefore depends on the value of the CMS estimator. Based on this, the traces have been evaluated and it has been shown that it is possible to establish a value of the CMS estimator at which the positive concentration in an area of the universe is almost 100%.

In addition, the traces have been evaluated for different values of CMS width and depth, demonstrating that to increase the positive concentration and define threshold values close to the real value, it is more accurate to increase the depth value than the width value. In figures from 13 to 25 it is shown that, when using same values of width and varying only depth value, positive concentration is almost 100% with a CMS value smaller for higher values of depth.

As future work, it would be useful to analyse whether, in addition to positive concentration based on the CMS estimator, it also exists in specific areas of the universe, such as IPv4 address prefixes. In this way, the results obtained by the attacker would be more accurate and false positives could be reduced.

Acknowledgement

I would like to thank my tutor, Pedro Reviriego, for his great help and involvement in this thesis from the very beginning. It has been a pleasure to work with you.

References

[1] Y.Liu, W.Chen, andY.Guan, “Identifying high-cardinality hosts from network-wide traffic measurements, ”IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 5, pp. 547–558, 2016.

[2] W. Chen, Y. Liu and Y. Guan,“Cardinality change-based early detection of large-scale cyber-attacks’,’ in Proceed- ings IEEE INFOCOM, 2013.

[3] Apache Data Sketches Library. [Online]. Available: https://datasketches.apache.org/docs/Community/KDD_Tutorial_Summary.html

[4] B. Bloom, “Space/time tradeoffs in hash coding with allowable errors,” Communications of the ACM, vol. 13, no. 7, 1970.

[5] G. Cormode, and S. Muthukrishnan, “An Improved Data Stream Summary: The Count-Min Sketch and its Applications”

[6] M. Langheinrich, “The golden age of privacy?” IEEE Pervasive Computing, vol. 17, no. 4, pp. 4–8, 2018.

[7] N. Kshetri and J. Voas, “Thoughts on general data protection regulation and online human surveillance,” Computer, vol. 53, no. 1, pp. 86–90, 2020.

[8] F. Hartman, “Count-Min Sketch”, July 2019. [Online]. Available: https://florian.github.io/count-min-sketch/

[9] P. Reviriego, A. Sánchez-Macián, S. Walzer, E. Merino-Gómez, S. Liu, and F. Lombardi, “On the privacy of counting Bloom filters,” IEEE Transactions on Dependable and Secure Computing, pp. 1–1, 2022 (in press).

[10] P. Reviriego, A. Sánchez-Macián, E. Merino-Gómez, O. Rottenstreich, S. Liu and F. Lombardi “Attacking the Privacy of Approximate Membership Check Filters by Positive Concentration” IEEE Transactions on Computers (in press).

[11] S. Pinilla “On the Privacy of the Count-Min Sketch: Exploiting Positive Concentration”. [Online]. Available: https://github.com/silpinj/Privacy-CMS

[12] J. Martínez, “On the Privacy of Count-Min Sketch”. [Online]. Available: https://github.com/mladron/CMS-Privacy

[13] General Purpose Hash Functions Algorithm: RS Hash Function [Online]. Available: https://www.partow.net/programming/hashfunctions/#RSHashFunction

[14] General Purpose Hash Functions Algorithm: JS Hash Function [Online]. Available: https://www.partow.net/programming/hashfunctions/#JSHashFunction

[15] CAIDA, San Diego, CA, USA, Realtime Passive Network Monitors, 2014. [Online]. Available: https://www.caida.org/catalog/datasets/passive_dataset_download/