Fraud Detection using Heavy Hitters: A Case Study

Bruno Veloso[1], Carlos Martins[2], Raul Azevedo[2], João Gama[1]

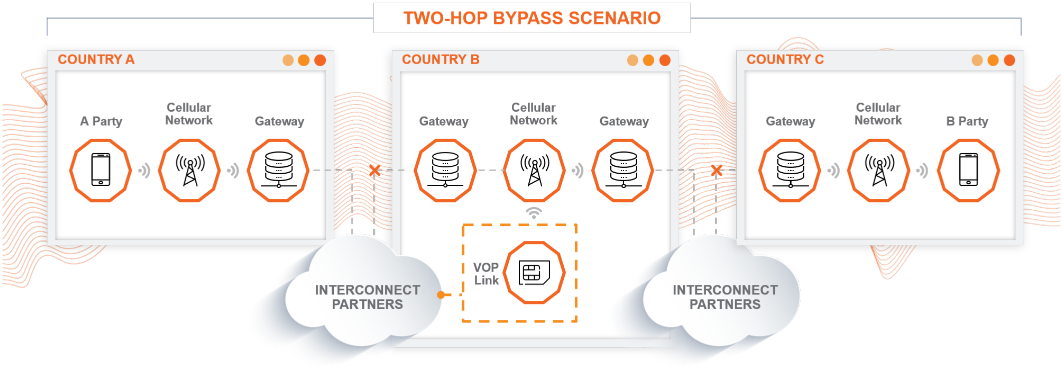
[1]LIAAD-INESCTEC; [2]WeDo Technologies

jgama@fep.up.pt

Contents

Problem definition

In Interconnect Bypass Fraud, one of several intermediaries responsible for delivering phone calls forwards the traffic over a low cost IP connection.



Problem definition

- In the telecommunication world, fraud is defined as the abusive usage of network and services without the intention of paying.
- Following a survey by the Communications Fraud Control Association, interconnect bypass fraud is one of the largest sources of lost revenues and costs network operators.
- This type of fraud is detected by analysing the call patterns of the gateways.
- but, the behaviour of gateways evolve over time, resembling some of them, true SIM Farms, capable of manipulating identifiers, simulating standard call patterns similar to the ones of normal users

Proposed Approach

How to detect interconnect bypass fraud on telecommunications?

Current approaches are based on `blacklists`:

- Inefficient in detecting new frauds
- Inefficient in detecting changes in the patterns

Our approach is data driven and works online. We are looking for:

- High asymmetry of international termination rates.
- High activity with abnormal behaviours.
  - Bursts of calls – a huge amount of calls;
  - Calling large set of numbers
  - Repetition – same pattern of calls during a period of time;
  - Mirror – the huge amount of calls are divided by multiple numbers.

Used Techniques

Detect in real time and as soon as possible:
One pass streaming algorithms!

- Frequent Items
  - Heavy Hitters – provide approximate counts of the frequent items [1].
  - Hierarchical Heavy Hitters – provide a rank of the most frequent items in a specific hierarchy [2].
- We signal alarms, when calling numbers, exhibit activity profile:
  - Large number of phone calls - HH
  - Bursts in activity - HH
  - Calling too many numbers - HHH

---

[1] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," inVLDB'02

[2] G. Cormode,S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data", TKDD

Experimental Work

- Two data sets: different periods
- Each record (one phone-call) contains information about:
  - Origin numbers (A-Numbers).
  - Destination number (B-Numbers).
  - Timestamp.
  - Blacklist Code: if the A-number is in the blacklist or not.

Data set 1

- Collected during three months between 24/07/2018 to 21/10/2018
- 89 days which includes 83.366.367 examples.
- Unique ANumber: 9.006.011
- Unique BNumber: 2.387.932

Data set 2

- Collected during one month between 01/06/2019 to 30/06/2019
- 29 days which includes 32.879.670 examples.
- Unique ANumbers: 3.217.069
- Unique BNumbers: 1.380.235

Experimental Work – HH

- The sequence of A numbers are used as a stream:
  Frauds are originated from A numbers,
- Use the lossy counting algorithm to provide approximate counts of the frequent items

Experimental Work – Contribution

Lossy Count

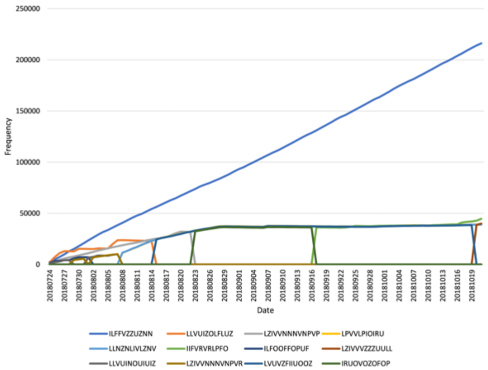
input: S: A Sequence of Examples; ϵ: Error margin;
begin
    $n \leftarrow 0$; $\Delta \leftarrow 0$; $T \leftarrow 0$;
    foreach *example* $e \in S$ **do**
        $n \leftarrow n + 1$
        if e *is monitored* **then**
            Increment $Count_e$
        else
            $T \leftarrow T \cup \{e, 1 + \Delta\}$
        end
        if $\left\lceil \frac{n}{\epsilon} \right\rceil \neq \Delta$ **then**
            $\Delta \leftarrow \frac{n}{\epsilon}$
            foreach *all* $j \in T$ **do**
                if $Count_j < \Delta$ **then**
                    $T \leftarrow T \backslash \{j\}$
                end
            end
        end
    end
end

Lossy Count with Forgetting

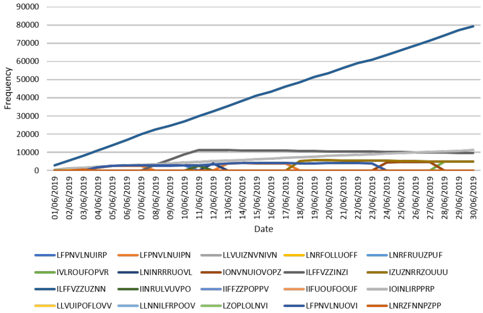
input: S: A Sequence of Examples; ϵ: Error margin; α: fast forgetting parameter
begin
    $n \leftarrow 0$; $\Delta \leftarrow 0$; $T \leftarrow 0$;
    foreach *example* $e \in S$ **do**
        $n \leftarrow n + 1$
        if e *is monitored* **then**
            Increment $Count_e$
        else
            $T \leftarrow T \cup \{e, 1 + \Delta\}$
        end
        if $\left\lceil \frac{n}{\epsilon} \right\rceil \neq \Delta$ **then**
            $\Delta \leftarrow \frac{n}{\epsilon}$
            foreach *all* $j \in T$ **do**
                $Count_j \leftarrow (1 - \alpha) * Count_j$
                if $Count_j < \Delta$ **then**
                    $T \leftarrow T \backslash \{j\}$
                end
            end
        end
    end
end

Experimental Work: Lossy Counting: Top-k A numbers

Data set 1

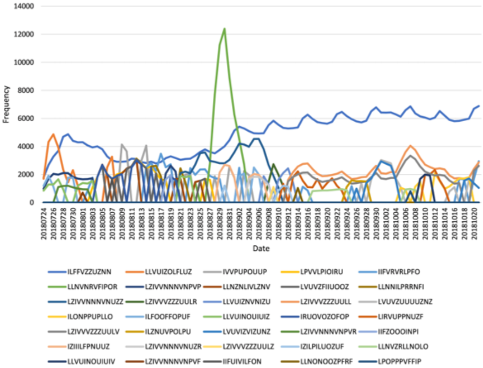


Data set 2

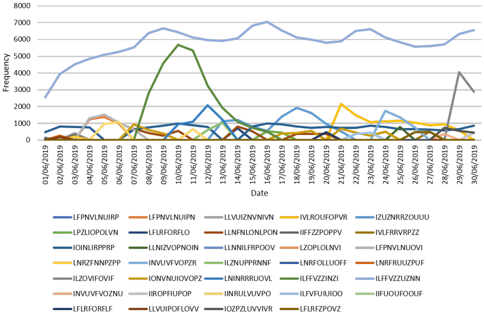


Experimental Work: Lossy Counting w/ Forgetting Top-k A numbers

Data set 1



Data set 2



Sensitivity Analysis

- Forgetting Parameter Sensitivity; α is the forgetting parameter and UAN is Unique A-Numbers

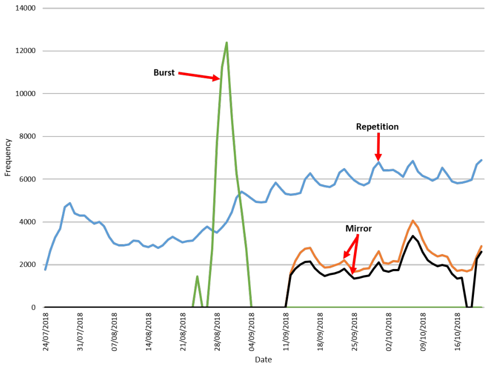
α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.99
UAN	211	210	203	192	180	175	158	123	93	66	12

- Performance comparison of the Lossy Counting (LC) vs Lossy Counting with Fast Forgetting (LCFF)

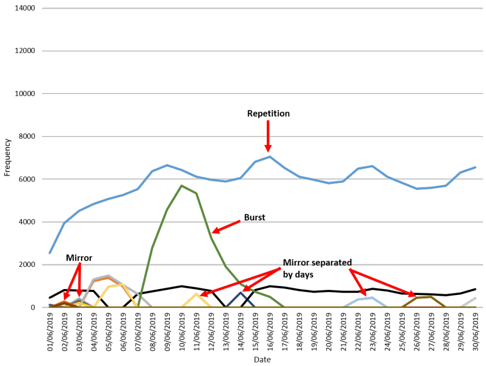
Algorithm	Runtime (s)	Memory (MB)	SpeedUp (Examples/s)
LC	88	75.8	947 345
LCFF	72	28.8	1 157 866

Experts Annotation

Data set 1



Data set 2



Discussion

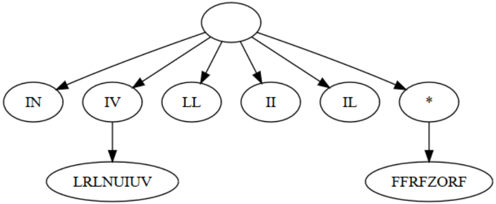
- **Contributions:**
  - **Application level:** Real-time identification of suspicious behaviours of A numbers.
  - **Methodology level:** with the extension of the Lossy Counting algorithm with a fast forgetting mechanism to rapidly detect abnormal behaviours.
- **Achievements:**
  - Inability of the Lossy Counting algorithm to detect recent items with abnormal behaviours.
  - The results show that our proposal improved the detection of these recent items.
  - The forgetting mechanism reduces the execution and memory used to compute the data stream, increasing the speedup of the algorithm.

Experimental Work – HHH

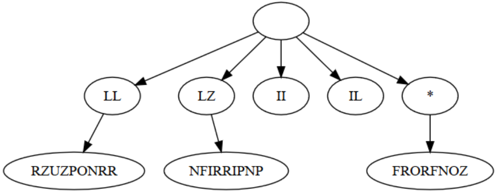
- Each A number is described by (Example phone number "IVLRLNUIUV"):
  - Country code – first two digits – "IV"
  - Sub-range – five digits – "LRLNU"
  - Number – last one, two or three digits – "IUV"
- Use a hierarchical heavy hitters, to find the most frequent items in a specific hierarchy

Experimental Work – HHH – Country Code

Data set 1



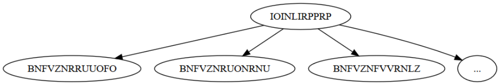
Data set 2



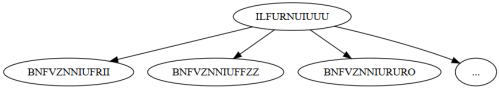
Experimental Work – HHH: ANumber-BNumber

A-numbers that call to too many B-numbers

Data set 1



Data set 2



Experimental Work: Top-k ANumber-BNumber

Data set 1

Rank	ANumber	# BNumber
1	IVVPUPOUUP	26868 (301)
2	LLNZNLIVLZNV	26686 (299)
3	LPVVLPIOIRU	26478 (297)
4	IRUOVOZOFOP	26473 (297)
5	LVUVZFIIUOOZ	26399 (296)
6	LLNVZRLLNOLO	23342 (262)
7	LLVUIZOLFLUZ	19116 (214)
8	LIRVUPPNUZF	13703 (153)
9	ILFFVZZUZNN	12000 (134)
10	IOINLIRPPRP	8595 (96)

Data set 2

Rank	ANumber	# BNumber
1	ILFFVZZUZNN	6002 (207)
2	IOINLIRPPRP	5782 (199)
3	ILFFVZZINZI	5055 (174)
4	LFPNVLNUIPN	3654 (126)
5	LFPNVLNUOVI	3643 (125)
6	LFPNVLNUIRP	3517 (121)
7	ILFURNUIUUU	2855 (98)
8	ILZOVIFOVIF	2782 (96)
9	LLNRUZIORILO	2220 (77)
10	ILZNUPPRNNF	2214 (76)

Experimental Work: ANumber-BNumber HHH vs HH

Data set 1

Rank	ANumber	# BNumber
10	IOINLIRPPRP	8595

- One new ANumbers identified by the HHH when compared with HH

Data set 2

Rank	ANumber	# BNumber
7	ILFURNUIUUU	2855
9	LLNRUZIORILO	2220

- Two new ANumbers identified by the HHH when compared with HH

Conclusions

The experiments shows:

- Real-time identification of anomalous behaviors.
- Approximate counting algorithms are efficient to identify anomalous beaviours:
  - The Lossy Counting algorithm can be improved with forgetting techniques.
  - efficient to detect recent items with abnormal behaviours:
    burst of calls, repetition and mirror behaviours
- The hierarchical heavy hitters can identify the ranges and numbers with higher volumes of calls with a defined structure

Thank you!

Questions: jgama@fep.up.pt

References I

G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pp. 346–357, Elsevier, 2002.

G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 4, p. 2, 2008.