

Count Unique IPs in anonymous way

Jaskaran Veer Singh
(jvsg)

IRC nick: jvsg

E-mail : jvsg1303 *at* gmail *dot* com

PGP Key ID : 9E1A6AD8

Fingerprint : 2814 3FB7 A32D 429B 092E 27F0 8AA3 C532 9E1A 6AD8

Timezone: UTC+5:30

Abstract

Currently, relays and bridges maintain a list of IP addresses of the devices that connect to it for various reasons such as for use by the bridge to check which country has them blocked. This is dangerous because if any of these tor instances get compromised, clients will be de-anonymized. To solve this issue, I propose a new data structure that keeps a track of unique IP addresses seen.


Goals

Implement counting algorithm that keeps a track of the number of unique IPs a relay or a bridge sees in a way that does not compromise the anonymity of the client and also has little error rate.

Threat model and Security Considerations

Consider an adversary with the following powers:

1. Has sufficient computational and storage power to brute force any method that can be brute forced.
2. Can get the recurrent control of the concerned guard-node/bridge.
3. Can interact with the concerned data structure that stores unique-IP-addresses/hash-values/bloom-filter/bitmaps etc.
4. Can also log incoming connections and IP addresses outside the realm of Tor(i.e at the system level or at gateways etc.)
5. Can manipulate the incoming connection with some made up IP address as to observe the working of our proposed solution.
6. As a consequence of previous power, adversary can also inject pattern of IP addresses to observe any pattern in the stored data structure.



An ideal solution would not involve hashing or even if it does, it would manipulate that hash to be stored in such a way that an adversary cannot learn about IP addresses even with a brute force attack.

An ideal solution would not help the adversary observe any pattern in the stored data structure. This could be accomplished by incorporating salted hash or variations of it into the proposed solution. And the salt would be changed everytime we start tracking unique IP addresses.

There is a fundamental limitation to what we can do and that is that we cannot stop an adversary from gaining knowledge of IP addresses at the system level or at gateways etc. But, the thing to cheer about is that in this way, the adversary cannot learn about the users retrospectively.

Research

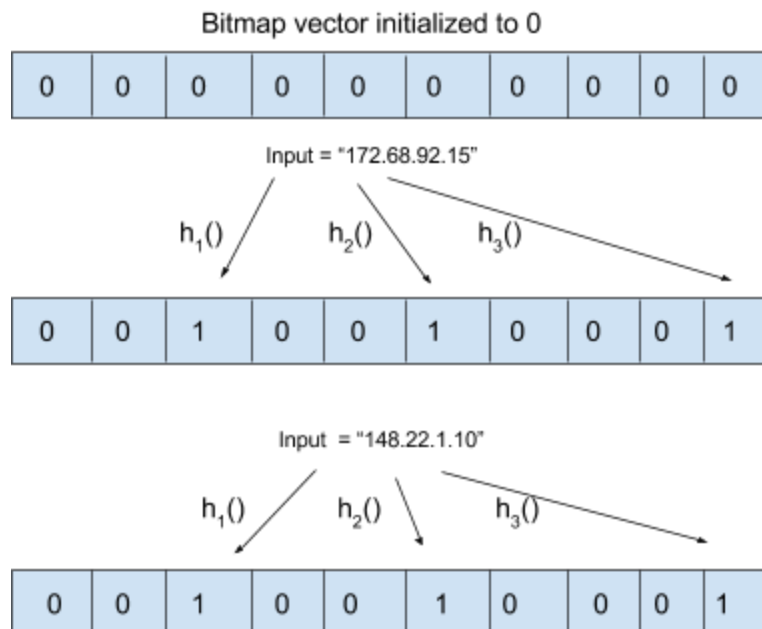
TL;DR - PCSA is the best algorithm suited for this task. It solves almost all security threats we have while also being efficient. Most of the talk about this proposal on the mailing list were about people talking of Hashing algorithms and its variations. But PCSA wasn't discussed. So you can go straight to the PCSA section below if you want to see how the solution works.

There are 4 ways to solve this problem. All the three ways are actually Big Data Algorithms. A few problems arise for each of these algorithms since they are made for big data but the data we would provide is not necessarily big. Let's have a look at the 4 solutions :

NOTE : In the following, I've tried to lead the reader from one topic to the other while pointing out the flaws that each method has. Later, I've tried to lead the reader to the conclusion that PCSA solves our problems.

Bloom Filter

A bloom filter maps the input to positions on the bitmap after passing through 2 or more hash functions. Later any new input are mapped onto this bitmap in the same way to check whether this value is already present in the set. The feature of this bitmap is that collisions could happen. And this collision creates deniability.



According to the above diagram, two different inputs map to the same output. On the one hand, one of these inputs doesn't count to be unique (although in reality, it is), and on the other hand, this is beneficial since this creates deniability. The person who gets hand on this data structure could never be 100% sure about the original inputs.

Obstacle

Suppose if the number of inputs is small. Let's say we receive just 1 connection in a day from some small, less busy country like Estonia. In that case, there might not be any chance for collision and the adversary could determine the IP address with some brute force. Hence this algorithm isn't suited for us.

Rappor¹ (adapted form)

Randomized **A**ggregatable **P**rivacy **P**reserving **O**rdinal **R**esponses is an algorithm where the system adds some deterministic and nondeterministic noise to the data that has to be stored. This creates deniability. In our case, we don't need to have deterministic noise added at first stage. So we'll just stick to adding non deterministic noise and storing it in a bloom filter.

One thing to note here is that, we should not accept output of the non deterministic randomizer which can be traced back to any IP address of Class E since those IP

¹ <https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/42852.pdf>

addresses are not in use and the adversary could easily know that those have been produced after adding random noise.

Obstacle

Using brute force technique, the adversary could check to see whether the IP address stored is the correct one or produced using random noise. In this technique, the adversary could compare those IP address (obtained via brute force) to the directory of what IP addresses are allotted to what country. The one that does not match, is the one that has been faked by using random noise.

Probabilistic Counting with Stochastic Averaging² (PCSA)

It is one of the classical algorithms in the world of Probabilistic counting. Error is bounded by $0.78/\sqrt{\text{substream count}}$. Below is the algorithm written in pseudo-code:

```
m = 2^b # with b in [4...16]
bitmaps = [[0]*32]*m # initialize m 32bit wide bitmaps to 0s

# Construct the PCSA bitmaps
for h in hashed(data):
    bitmap_index = 1 + get_bitmap_index( h,b ) # binaryaddress of the rightmost b bits
    run_length = run_of_zeros( h,b ) # length of the run of zeroes starting at bit b+1
    bitmaps[ bitmap_index ][ run_length ] = 1 # set bit based on run length observed

# Determine the cardinality
phi = 0.77351
DV = m / phi * 2 ^ (sum( least_sig_bit( bitmap ) ) / m) # the DV estimate
```

Hence the cardinality is $DV := m \cdot 2^{z/m} / \Phi$

Φ is 0.77351 and m is the number of bitmaps

Obstacle

This algorithms stated above is made for use on large databases. Infact, these were invented to save time and space while doing basic set operations on data with high

² <https://research.neustar.biz/tag/pcsa/>

cardinality. But the data we would provide as an input is not necessarily of high cardinality. Since we would be counting numbers for each country separately, so the expected value of the input cardinality would be :

$$0 \leq C \leq 2500 = 0.78/\text{sqrt}(m)$$

where m is the number of bitmaps.

WorkAround³

As a workaround, we could introduce a correction term for small values of cardinality. So our final formula could look something like:

$$DV(S_1, \dots, S_m) := m \cdot (2^{Z/m} - 2^{-K \cdot Z/m}) / \rho$$

Where K is chosen to be around 1.75.

Timeline

I. May 4 - May 30 (Community Bonding Period)

- Focus on Simulating the PCSA algorithm (with corrections) to weed out any chances for high error probability in output.
- Write a proposal in plain text that goes into the torspec.
- Also mail this proposal onto the mailing list so that everyone could provide their views.

II. May 30 - June 9 (Possibility of exams during this time)

I'm not sure as the exams date haven't been out till now. But if there are exams during this time, I'll try to begin work during Community bonding period itself.

III. June 10 - June 17 (Fix #8786)

- Add extra-info line that tracks the number of consensus downloads of each pluggable transports.
- This would count consensus fetches instead of direct connections to users.

IV. June 18 - June 23 (Continue with fixing #8786)

- Would improve the granularity of the bridge statistics.

³ <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.304.6339&rep=rep1&type=pdf>

- Add a dirreq-v3-transport line in the stats. This denotes the bridge statistics.

V. June 24 - June 30 (Start implementing PCSA)

- Remove the existing data structure that store IP addr.
- Replace it with a data structure that has a separate counter for each country.

----- Phase-1 Evaluation -----

VI. July 1 - July 7 (Wrap up with PCSA implementation)

- Make sure to implement the formula correctly.
- Dry run it on a few inputs to check for the optimal value of constant Kappa.

VII. July 7 - July 14 (Write test for the code written so far)

- Rewrite the existing tests that require geoip functionality.
- Write test cases for the new algorithm.
- Test cases should have data ranging from zero to 6000 connections as that is near to the amount that tor receives.

VIII. July 15 - July 21 (Write test for the code written so far)

- Continue with previous week's work, as that will take quite some time.

IX. July 22 - July 28 (Documentation)

There are some places (in logs and elsewhere) where we state the unique number of IP addresses seen. That needs to be replaced by a message telling that the numbers are not exact but rather probabilistic followed by the numbers itself.

----- Phase-2 Evaluation -----

X. July 29 - August 04 (Buffer Period)

Would use this time to improve upon my code, or fix bugs if they are noticed

XI. August 5 - August 11 (Buffer Period)

Would use this time to improve upon my code, or fix bugs if they are noticed

XII. August 12 - August 18 (Buffer Period)

Would use this time to improve upon my code, or fix bugs if they are noticed

----- Pencils Down -----

About Me / Why I'm fit for this project

My friends call me "annoying privacy evangelist" as I keep poking them with the mistakes they regularly do online that compromises their privacy. I have been a privacy/anonymity advocate and a supporter of FOSS for long.

I was selected and completed a project in Libreoffice during GSoC 2016 and my work involved C++ and a HUGE codebase. So I guess now I'm fairly trained in handling large codebases. I have been associated with Libreoffice even after completing my GSoC, and most probably soon I will be a member of the TDF (Libreoffice's parent organisation).

I am associated with the Tor project since some time. I used to lurk around at the dev channel at OFTC and tried to learn by the discussions that used to happen there.

Point us to a code sample:

My work in Libreoffice (around 60 commits, mostly C++) : <https://goo.gl/eIvldP>


My work in Orcus (around 50 commits, mostly C++) : <https://goo.gl/kjDHR>

I had realized there aren't many File Encryption and storage programs available for Linux unlike Mac and Windows. So I've created this : <https://github.com/jvsg/mitron> . It is a work under progress.

Why do I want to work with The Tor Project in particular?

I come from a country where Right to Privacy isn't considered a fundamental right. So every citizen is responsible for his/her privacy. I believe that freedom of speech can only be fully exercised when a person is anonymous.

Quite a large percentage of people do not care about their anonymity/privacy online and more efforts should be taken to reach out to them. Few of the establishments and



organisations have interest in breaking the anonymity system that Tor provides. Hence it becomes even more important to constantly seek ways of improving Tor.

Not only philosophically and politically, but I'm also fascinated by Tor from a technological point of view. I have been using Tor since a long time now and I believe it's my turn now to give back to the community.

Will I be working full-time on the project for the summer?

Yes, but most probably during the first week of June I would have my exams. So apart from just one week, I would be available full time during the rest of the summer.

My Academic Pursuits

I am a undergrad student at a University in New Delhi, India. My major is Electronics and Communication Engineering. Sometime in the future, I would like to pursue research in the area of Anonymity systems.

Is there anything else that you should know that will make you like my project more?

There was a concern whether this project needs to be implemented. So I'd like to make this sure to everyone that implementing this projects would have it's benefits. We'll be able to get rid of the problematic data structure which IMO is dangerous to have for any anonymity system.

Am I applying to other projects for GSoC and, if so, what would be my preference if you're accepted to both? Having a stated preference helps with the deduplication process and will not impact if you accept my application or not.

Nopes. This is the only organisation I'm applying to.