RSSAC040
Recommendations on Anonymization Processes for Source IP Addresses Submitted for Future Analysis

A report from the ICANN Root Server System Advisory Committee (RSSAC)
7 August 2018

Preface

This is an Advisory to the Internet Corporation for Assigned Names and Numbers (ICANN) Board of Directors and more broadly to the Internet community from the ICANN Root Server System Advisory Committee (RSSAC). In this report, the RSSAC conducted a technical analysis of harmonizing anonymization procedures for DNS data.

The RSSAC seeks to advise the ICANN community and Board on matters relating to the operation, administration, security and integrity of the Internet's Root Server System. This includes communicating on matters relating to the operation of the Root Servers and their multiple instances with the technical and ICANN community, gathering and articulating requirements to offer to those engaged in technical revisions of the protocols and best common practices related to the operation of DNS servers, engaging in ongoing threat assessment and risk analysis of the Root Server System and recommend any necessary audit activity to assess the current status of root servers and root zone. The RSSAC has no authority to regulate, enforce, or adjudicate. Those functions belong to others, and the advice offered here should be evaluated on its merits.

A list of the contributors to this Advisory, references to RSSAC Caucus members' statement of interest, and RSSAC members' objections to the findings or recommendations in this Report are at end of this document.

Table of Contents

1. Introduction

DNS operators, in particular Root Server Operators (RSOs), are periodically requested to collect query data and submit it to a central storage location where it is accessible for future research. The typical example is the yearly Day In The Life of the Internet (DITL) collections undertaken by the DNS Operations Analysis and Research Center (DNS-OARC). Some operators are uncomfortable sharing IP addresses of the query sources and some are even legally prevented from doing so. In those cases, the compromise has been for the RSO in question to anonymize the source IP addresses of the queries.[1]

When DNS data is anonymized, if each organization anonymizing does this anonymization in a different way, it is difficult for researchers to compare results between the datasets of different organizations. At the other extreme, if no anonymization is carried out by operators, researchers can easily correlate queries seen in different datasets. Finding a good balance between the privacy of queriers and datasets that allow for query correlation is difficult. One idea is to harmonize the anonymization processes, so that query sources are anonymized in the same way by all involved organizations, thereby making it possible to cross-index between data sets and to recognize queries coming from the same source, without divulging the IP address of the source.

Multiple levels of harmonization are possible. Operators may adopt the same algorithms and configuration (such as key or salt values), allowing direct comparison of anonymized IP addresses across multiple operators. Alternatively, they may adopt the same algorithms but with different configurations, thereby simplifying how anonymization is done across operators, without allowing direct comparison of anonymized addresses.

1.1 Statement of Work

The RSSAC Caucus Harmonization of Anonymization Procedures for Data Collecting Work Party was asked to:

1. Consider whether harmonization of anonymization procedures is something to recommend to the RSO community (and possibly to the DNS community at large);
2. If yes to 1, recommend a preferred way to anonymize the data, specifying algorithms and procedures; and finally
3. Consider whether to recommend that anonymization be undertaken by all who submit data, or remain optional for those who see a need to do so.

It is expected that the work will reach into the wider DNS community, and not only the root server community.

[1] Although this text focuses on RSOs, this material applies to sharing of DNS data more generally. We do not believe the technical methods in this work are RSO specific, and invite other groups to consider this work to the extent it applies to their use.

1.2 Scope of this Document

The methods described in this document could be used by authoritative server operators other than the RSOs, and could also be used by recursive resolver operators who want to share their data with researchers.

This document does not recommend harmonization of anonymization. Individual RSOs who want to (or are required to) anonymize IP addresses to increase privacy and researchers who might want to correlate DNS queries have different requirements for anonymization.

Thus, the different methods for anonymization proposed in Section 4 of this document have different properties for preventing or enhancing the ability to correlate queriers. Some methods require a secret random key to be used within a dataset, and that secret would have to be shared among RSOs and kept secret in order to prevent someone from later being able to identify queriers. One method does not require a secret key, but allows easy identification of query sources in the dataset if someone also has a non-anonymized list of query sources for comparison.

Given these considerations, and as seen in Section 6 of this document, this document does not make a specific recommendation on which method to choose.

1.3 Terminology

Root Server Operator (RSO)
A **root server operator** is an organization responsible for managing the root service on IP addresses specified in the root zone and the root hints file.[2]

Day in the Life of the Internet (DITL)
The Day in the Life of the Internet[3] (DITL) collections coordinated and archived by the Domain Name System Operations Analysis and Research Center (DNS-OARC) that are contributed to by root server and other DNS operators at least once a year.

2. Introduction to Anonymization

When one discusses "anonymization" of data, the typical assumption is that someone reading the data could not identify the source of the DNS query in the data. In the case of anonymizing DNS requests, the part of the data that is anonymized is the source IP address. The methods described

[2] See RSSAC026: RSSAC Lexicon
[3] See https://www.dns-oarc.net/oarc/data/ditl

here for anonymizing source IP addresses can also be used to anonymize other addresses, such as IP addresses that appear in the data portion of responses.

One common example of the desire for anonymization is for the DITL data, which has, among other things, a large, time-ordered set of requests to all root server instances collected on the same day. Currently, only a few RSOs anonymize their DITL data, but more might do so in the future.

Researchers have a desire to positively associate all the queries from one query source in order to see the sequence of requests coming from the source. They also have a desire to associate queries from a single source that go to different root servers.

2.1 Benefits and Drawbacks of Harmonization of Anonymization

If a set of Root Server Operators (RSOs) uses the same form of anonymization and the same secret value for one or more datasets, researchers can correlate queries from the same source across the datasets. For example, seeing the anonymized address “221.42.83.191” in the datasets from different RSOs would allow a researcher to assume that the same source asked the different root servers.

However, if each RSO uses their own form of anonymization, datasets cannot be compared across RSOs. In such a scenario, it would be impossible to know which IP addresses are sending queries to multiple roots. Even if RSOs use the same form of anonymization, when different secret values are used (as described in Section 3), datasets cannot be compared across RSOs. That is, using a different secret value for one particular form of anonymization reduces the benefits of harmonization (the primary remaining benefit is that users of the data will know the harmonization *method*, which might be useful if they understand its constraints on their analysis).

Some RSOs might even use different anonymization across instances within the RSO, which would make it impossible to see if the same source was querying multiple Anycast instances for a single RSO, such as due to routing changes during the data collection.

A drawback of harmonized anonymization with the same secret value as other RSOs use is that all effective methods that allow correlation across RSOs require the RSOs to share a secret value and for that secret to be kept secret forever. If the secret is accidentally revealed, it would be easy for anyone with that secret to de-anonymize all of the datasets. This could have repercussions on the RSOs who are anonymizing for legal reasons and potentially on users whose DNS queries might be revealed.

Thus, there should be a distinction between consistent anonymization algorithms and consistent secret values across RSOs. If there was a standard way of anonymizing IP addresses and secret values, researchers would benefit because they would have much better insight into how resolvers use the entire Root Server System (RSS). Although it is impossible to estimate the value of this increased insight, it is generally assumed that better research for the RSS will increase stability and reliability in the system. The downside for the RSOs is that one RSO accidentally revealing the key would affect all RSOs who used that key.

3. Overview of Methods for Anonymizing IP Addresses

We propose five primary ways to anonymize the IP addresses in a data set:

1. Cryptographically mix each new address seen with AES-128 and a secret value, then truncate the output of IPv4 addresses to 32 bits
2. Cryptographically mix each new address with the Cryptopan method (mix bit-by-bit with a secret value)
3. Cryptographically encrypt each new IPv4 address seen with ipcrypt and a secret value, and encrypt each new IPv6 address with AES-128
4. Zero out the least-significant bytes from the address (the last byte for IPv4 addresses, and the last eight bytes for IPv6 addresses)
5. Assign each new address seen to an existing list of mapped addresses
6. Cryptographically mix each new IPv4 address seen with AES-128 and a secret value, leaving the output to be 128 bits

It is important to note that anonymizing source addresses might not be sufficient to prevent a determined attacker or researcher from unblinding some of those addresses. In large datasets, analysis of the IP header checksum field and the UDP checksum field in queries could narrow the possibilities of the real value of the anonymized source IP address. Such an attack can be prevented by zeroing the two fields.

Appendix A contains pseudocode for a system that embodies mechanisms 1 through 4.

3.1 Using Table Lookups To Speed Anonymization

Methods 1 through 3 create lists that can be used to look up already seen values. Using such a list makes sense because it can reduce the computational cost (CPU and memory space) of anonymizing as long as the input dataset has not grown exceedingly large, for example, due to a Denial of Service (DoS) attack with a large number of spoofed sources. For example, in the 2017 DITL collection, even for the largest root server identity, there were fewer than 5 million unique IP addresses. Thus, the address set is relatively small and with full source address anonymization by an attacker, each address repeats roughly a thousand times per “real” address. (Method 4 does not need to use a lookup method, and in fact would run more slowly if such a list was used.)

3.2 Using Salt to Improve Anonymization

In the current Internet, the vast majority of DNS requests seen at root servers are over IPv4, so most of the anonymized data would be IPv4 addresses. Anonymizing data that only has a small number of possible values using cryptographic functions without mixing additional, secret information (cryptographic salt) into the data makes the values easy to determine. Although 4 billion might initially seem like a large number, it is feasible to run whatever algorithm is chosen over all 4 billion addresses to get the full map. This kind of pre-computed map is called a *rainbow table*.[4]

The use of a large secret number (ideally randomly chosen) mixed into the function prevents this type of attack. Anonymizing data using cryptographic methods has been a common practice for decades. Three common methods for anonymizing are encrypting with a secret key, encrypting with a secret value added, and keyed hashes. Mixing using encryption is typically done with AES-128; mixing using hashes is typically done with SHA-256. The mixing functions of both AES-128 and SHA-256 are considered completely secure. Because AES-128 is usually faster than SHA-256 for small inputs such as IP addresses, only mixing with AES-128 is described here.

In methods 1 through 3, a secret key is used for an entire dataset. In this context, the size of a “dataset” is up to the RSO who is doing the anonymizing. Changing the secret key inherently causes a new dataset to be created because the anonymized source addresses will immediately change when the key changes. An RSO can decide how often it wants to change the secret key.

Methods 1 through 3 have the following limitations:

- Unless the secret value that is mixed into the address is shared across datasets, addresses in each dataset will result in different outputs; harmonization across such datasets is impossible.
- If long-term anonymization is desired, the secret value used for anonymizing a particular dataset must be kept secret forever. A common way to keep a secret forever is to forget it, that is, not have it available to anyone ever again.
- Efficient use of any of these methods relies on looking up already seen values, and thus can become inefficient during a DoS attack; in DoS attacks, the source addresses may be spoofed and are often very numerous.

Method 4 (zeroing out the least significant byte or bytes) has the limitation that it might not anonymize sufficiently due to some IP address ranges having few resolvers, but this method does not have any of the limitations listed immediately above.

[4] See https://en.wikipedia.org/wiki/Rainbow_table

3.3 Methods Not Recommended

Only methods 1 through 4 are considered in this document for deployment by RSOs; they are described in Section 4. This subsection describes the two methods that are not recommended.

3.3.1 Anonymization by Assigning New Addresses to a List

This method might seem sensible, but fails under a common scenario. This is the easiest method for anonymization, but becomes inefficient for datasets where the source addresses are randomized, for example, during a DDoS attack.

In this method, each new unique address seen is mapped to a different address from a list. For example, for IPv4 the first unique address in the dataset is mapped to 0.0.0.0, the next one to 0.0.0.1, and so on. The map is queried each time an address is analyzed.

This method has the following advantage for those anonymizing addresses:

- The description of the method is extremely simple.

This method has the following advantage for those using the datasets with anonymized addresses:

- The result will always be a one-to-one match with the source.

This method has the following disadvantage for those anonymizing addresses:

- This method relies on storing and looking up already seen values, and thus will become inefficient in the face of DDoS attacks where the source addresses are randomized and typically numerous. The search time for lookups will grow approximately linearly with the number of unique addresses seen.

3.3.2 Anonymization by Mixing IPv4 Addresses to 16-byte Values

This method might seem sensible, but that would likely cause significant problems for processes that are reading the anonymized records.

This method has the following advantage for those anonymizing addresses:

- The description of the method is extremely simple.

This method has the following advantage for those using the datasets with anonymized addresses:

- The method has the advantage over method 1 because there is no truncation and therefore no collisions of resulting addresses.

This method has the following disadvantage for those anonymizing addresses:

- The datasets become larger because 4-byte IPv4 addresses take up 16 bytes.

This method has the following disadvantage for those using the datasets with anonymized addresses:

- Software that is meant to work with all types of anonymization will not be able to tell whether each original source address in these sets was IPv4 or IPv6.

4. Major Proposals for Methods of Anonymizing IP Addresses

4.1 Mixing Full Addresses with Truncation

Addresses can be anonymized by encrypting the address using AES-128 with a secret key. For mixing full addresses with AES-128, truncation to 4 bytes for IPv4 addresses is required. Truncating IPv6 addresses after mixing is not required because AES-128 output is the same length as the IPv6 input. Using this method, the IPv6 addresses can be decrypted using the same key as was used for encryption, but IPv4 addresses cannot be decrypted at all. In this method, an IPv4 input address is padded to 128 bits by copying itself four times for the input; an IPv6 input address does not need to be padded.

When using cryptographic mixing and truncating the output to the input length of IPv4 addresses, collisions can result when two real IPv4 addresses map to the same output after truncation. This problem is known as the *birthday problem*[5], and the expected number of collisions is proportional to the number of unique addresses in the input. For example, if 4,000,000 IPv4 addresses (about the number seen at a root server identity per day) are mapped to 2^{32} possible values, on average there will be about 3,700 collisions (or 0.1%). A statistic like "number of unique sources" will be very slightly underestimated if computed using anonymized data. Formally expressed, the number of collisions $C=n(1 - (1 - 1/N)^{(n-1)})$ for a mean of C collisions from n unique inputs into a space of N addresses.[6]

Although the birthday problem has few collisions when the number of active IPv4 addresses is small, it is much worse when the number is large. For example, reports of the Nov. 30, 2015 DDoS attack on the roots indicate that roots saw about 895M unique addresses. If these had been mixed with truncation, there would have been 170M collisions. While many of these addresses were spoofed, using such a count represents one factor in estimating the cost of some DDoS defenses.

This method has the following advantages for those anonymizing addresses:

[5] See https://en.wikipedia.org/wiki/Birthday_problem

[6] See problem description and solution at, https://math.stackexchange.com/questions/35791/birthday-problem-expected-number-of-collisions

- It uses cryptography that has been well-known for decades.
- The description of the method is relatively simple.

4.2 Mixing Bit-By-Bit: Cryptopan

Crytpopan[7][8] is a mechanism that uses encryption with a key to anonymize IP addresses in a way that adds two properties beyond mixing full addresses:

- There is a one-to-one correlation between input and output addresses (that is, collisions are not possible in the output).
- The resulting addresses are "prefix-preserving", the prefix relationship is preserved in pairs of output addresses. For example, if two addresses in the input share the same /19, then they will also share the first 19 bits of the output addresses.

The algorithm used in Cryptopan is briefly described in Appendix B. Dnsanon[9] is a current implementation of Cryptopan that uses AES-128 as the encryption algorithm.

Note that Cryptopan, in order to achieve the two additional properties, is slower than simply mixing full addresses because it needs to do one mixing iteration per bit of source address.

This method has the following advantage for those anonymizing addresses:

- It has been used by multiple organizations for over a decade.

This method has the following advantages for those using the datasets with anonymized addresses:

- It does not suffer from a possible *birthday problem* as described in section 4.1.
- Prefix preservation may be useful for correlating addresses.

This method has the following disadvantage for those anonymizing addresses:

- Prefix preservation may expose correlation in addresses and thus have privacy implications.

4.3 Encrypting Addresses with ipcrypt and AES-128

ipcrypt[10] is a relatively new algorithm for directly encrypting 32-bit values like IPv4 addresses. In this mechanism, IPv4 addresses are encrypted with ipcrypt and IPv6 addresses are encrypted with AES-128, both using a secret key. The resulting addresses can be decrypted with the same key.

[7] See https://www.cc.gatech.edu/computing/Telecomm/projects/cryptopan/
[8] See http://www.cc.gatech.edu/computing/Networking/projects/cryptopan/icnp02.ps
[9] See https://ant.isi.edu/software/dnsanon/
[10] See https://github.com/veorq/ipcrypt

It is important to note that there has been no published analysis of ipcrypt. Although its author, Jean-Philippe Aumasson, is a well-respected cryptographer and the design is based on a popular hashing function, using a function before it has been analyzed is considered quite risky. The author describes ipcrypt "as a low-security toy cipher". In a message to the Internet Research Task Force (IRTF) Crypto Forum Research Group (CFRG),[11] he goes on to list some problems with ipcrypt:

- Because it uses 32-bit blocks, a chosen-plaintext codebook attack will work in time 2^{32} (or much less for specific IP ranges).
- Known-plaintext codebook attacks will work similarly, but in O(n log n), or 2^{37} due to the *coupon collector problem.*[12]
- There is a generic ~2^{16} distinguisher that works by looking for a collision in a sequence of blocks.
- Jason Donenfeld found a high-probability differential that seems detectable with fewer than 2^{24} chosen plaintext pairs, and which may speed up key recovery.

This method has the following advantage for those anonymizing addresses:

- The description of the method is relatively simple.

This method has the following advantage for those using the datasets with anonymized addresses:

- There are no collisions when using encryption.

4.4 Zeroing The Least Significant Bytes

An IPv4 address can be pseudo-anonymized by setting the last byte to all zeros, and an IPv6 address can be pseudo-anonymized by setting the last eight bytes to all zeros. That is, the source address is made zero to the /24 (IPv4) or /48 (IPv6) boundary. For example, 162.29.190.42 would become 162.29.190.0, and 2a04:e9cd:15::a0 would become 2a04:e9cd:15::0.

Note that there is disagreement in the technical community about what address boundaries are appropriate if this method is used. Some people suggested shorter boundaries (that is, zeroing more bits) for both IPv4 and IPv6 from the 24 (IPv4) or /48 (IPv6) boundary given above. An operator choosing this method to anonymize must also decide the boundaries on which to anonymize.

This method has the following advantages for those anonymizing addresses:

- It is so simple that it does not require the use of a lookup table to store previously-converted values. The method can be used on every address in less time than it takes to do a lookup.

[11] See https://www.ietf.org/mail-archive/web/cfrg/current/msg09494.html
[12] See https://en.wikipedia.org/wiki/Coupon_collector's_problem

- The description of the method is relatively simple.

This method has the following disadvantages for those anonymizing addresses:

- If there is only one resolver, or a small number of resolvers, and the boundaries chosen are too large, it is easy to de-anonymize the resolvers.
- The operator must choose boundary lengths for IPv4 and IPv6 for anonymizing, and every choice will either increase or decrease the anonymity of the input addresses, and will either increase or decrease the usefulness of the resulting addresses for researchers.

5. Saving Additional Information When Anonymizing

A side effect of anonymization is that it destroys any possibility of establishing the origin AS of the querier. In order to help researchers looking at anonymized data, the process of anonymization can preserve some information relating to the originating Autonomous System (AS) number of the querier. If the origin AS is sufficiently general that it does not unnecessarily expose data that should have been anonymized, it is useful to also publish the AS numbers of the anonymized addresses. This can most easily be done with a table that maps the anonymized addresses to their original AS numbers.

6. Recommendations

Recommendation 1: Root Server Operators should consider the advantages and disadvantages of harmonization of anonymization for DITL Data.
RSOs need to decide whether to pursue harmonization of anonymization data that comes from multiple operators, particularly the DITL data. That decision needs to include consideration of the advantages and disadvantages from the standpoint of the RSO, of the users of the RSS, and of researchers looking at the anonymized data.

Harmonization using mixing full addresses or bit-by-bit will help the research community correlate sources of DNS queries across datasets that are collected from different RSOs. However, full harmonization inherently relies on sharing a secret value that will invalidate the anonymization if it is later revealed.

Even if the RSOs decide not to harmonize with sharing of secret values, harmonizing the method used can help RSOs choose an anonymization strategy, and simplify understanding the properties of the data for those who use data from multiple RSOs.

Recommendation 2: Each RSO should consider the anonymization procedures in this document individually.
Any of the proposals given in Section 4 of this document can be used as the anonymization specification for IP addresses, depending on the policy of the party doing the anonymizing.

Recommendation 3: Autonomous System (AS) numbers of original addresses should be made available with the anonymized data if the origin AS is sufficiently general that it does not unnecessarily expose data that should have been anonymized.
It should be possible for an operator to publish a machine-readable table that maps the anonymized addresses to the AS of the original data. Such a table should have a timestamp for when the mapping was made due to AS values changing over time.

7. Acknowledgements, Disclosures of Interest, Dissents, and Withdrawals

In the interest of transparency, these sections provide the reader with information about four aspects of the RSSAC process. The Acknowledgments section lists the RSSAC Caucus members, outside experts, and ICANN staff who contributed directly to this particular document. The Statement of Interest section points to the biographies of all RSSAC Caucus members. The Dissents section provides a place for individual members to describe any disagreement that they may have with the content of this document or the process for preparing it. The Withdrawals section identifies individual members who have recused themselves from discussion of the topic with which this Advisory is concerned. Except for members listed in the Dissents and Withdrawals sections, this document has the consensus approval of the RSSAC.

7.1 Acknowledgments

RSSAC Caucus Members
Jaap Akkerhuis
Ray Bellis
John Bond
Joao Damas
Brian Dickson
John Heidemann
Paul Hoffman
Geoff Huston
Akira Kato
Warren Kumari
Lars-Johan Liman
Robert Story
Brad Verd
Paul Vixie
Duane Wessels

ICANN Support Staff
Andrew McConachie (editor)
Kathy Schnitt

7.2 Statements of Interests

RSSAC caucus member biographical information and Statements of Interests are available at: https://community.icann.org/display/RSI/RSSAC+Caucus+Statements+of+Interest

7.3 Dissents

There were no dissents.

7.4 Withdrawals

There were no withdrawals.

7.5 Revision History

This is the first version.

Appendix A: Pseudocode for Mixing Addresses in Datasets

This code shows how to mix addresses using the methods described in Section 4 of the document. The input is a stream of IPv4 and IPv6 addresses in wire format. For simplicity, this pseudocode ignores the padding necessary for AES-128.

```
Startup for 4.1, 4.2, and 4.3 (no startup is needed for 4.4)
- dataset_random is a 128-bit value used in the mixing functions
- address_map is used to map input IP addresses to anonymized addresses
- address_map_max_length is the maximum size of the map[13]
if this run is being added to an earlier dataset:
    set dataset_random to dataset_random from the earlier dataset
    set address_map to address_map from the earlier dataset
else:
    initialize dataset_random with a newly-chosen 128-bit random value
    initialize address_map to be an empty map

Convert a stream of input addresses to anonymized addresses for 4.1, 4.2, and 4.3
for each input_address in stream:
    if input_address is not already a key in the address_map:
       if length(address_map) >= address_map_max_length:[14]
            remove a random entry from the address_map
        set address_map[input_address] to mixing_function(input_address)
    return address_map[input_address]
save updated address_map

Mixing function for full addresses with AES-128 (Section 4.1)
define mixing_function(input_address):
    set output_holder to aes-128(msg=input_address, key=dataset_random)
    if length(input_address) is 4 bytes:
       set truncated_value to leftmost 4 bytes of output_holder
       return truncated_value
    else:
       return output_holder

Mixing function for Cryptopan (Section 4.2)
define mixing_function(input_address):
    set output_holder to length of input_address (32 or 128 bits)
    for each bit_position in length(input_address):[15]
       set this_message to input_address[0:bit_position] | dataset_random
       set this_bit to highest bit returned from aes-128(msg=this_message, key=0)
       set output_holder[bit_position] to this_bit
    return output_holder
```

[13] Determined by the amount of memory allocated for holding the table in memory.
[14] This makes the address map be more useful after a DDoS attack has finished.
[15] This code block gets executed either 32 or 128 times for each run of the function.

Mixing function for full addresses with ipcrypt or AES-128 (Section 4.3)

```
define mixing_function(input_address):
    if length(input_address) is 4 bytes:
       return ipcrypt(msg=input_address, key=dataset_random)
    else:
       return aes-128(msg=input_address, key=dataset_random)
```

Convert a stream of input addresses to anonymized addresses for 4.4

```
for each input_address in stream:
    if length(input_address) is 4 bytes:
       return input_address with last byte zeroed
    else:
       return input_address with last 8 bytes zeroed
```

Appendix B: Technical Description of Cryptopan

Since Cryptopan is much less widely known than simple mixing full addresses, we provide a brief technical description of the algorithm here. Cryptopan has been peer-reviewed,[16] and multiple implementations of it exist. As of 2018, interoperability between implementations has not been a goal.

Conceptually, Cryptopan uses a cryptographic function to mix each bit of the address as a function of the more significant bits concatenated with the salt. This concept has the properties that it is:

- deterministic, depending only on the input address and salt,
- 1:1, so every input address has exactly one output address, with no collisions, in the same number of bits,
- prefix preserving, so two input addresses with the same prefix of *i* bits have the same randomized prefix of *i* bits in the output.

Eliminating collisions reduces concerns that the statistics on the anonymized addresses will be different than those on clear addresses. Also, one can detect spoofed addresses from the same subnet in anonymized addresses.

Conceptually:

$$C'_i = \text{mix}(C_{[0..i]}\ \textbf{concat}\ \textit{salt})$$

Where C'_i represents the *i*th anonymized bit of the output (starting with the least significant bit), $C_{[0..i]}$ represents the first i+1 bits of the input starting from the least significant bit, **concat** means bitwise concatenation, and *salt* is some bitstring kept secret by the anonymizer to prevent brute-force attacks over a small input domain.[17]

A C++ implementation with test inputs is at the Dnsanon project.[18]

[16] See Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme. In Proceedings of the 10th IEEE International Conference on Network Protocols, pp. 280-289. Washington, DC, USA, IEEE. November, 2002. http://www.cc.gatech.edu/computing/Networking/projects/cryptopan/icnp02.ps

[17] Note that implementations do not directly implement this formula, and this formula does not specify the padding needed for the mixing function.

[18] See https://ant.isi.edu/software/dnsanon/