RSSAC0XX: Harmonizing the Anonymization of Queries to the Root

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

DD February 2018

Preface

This is an Advisory to the Internet Corporation for Assigned Names and Numbers (ICANN) Board of Directors and the Internet community more broadly 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.

1 Introduction

DNS operators, in particular Root Server Operators (RSOs), are periodically requested to collect query data and submit it to a central storage 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 querying servers. 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 the different organizations. 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, still 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, but not allowing direct comparison of anonymized addresses.

1.1 Scope 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 was expected that the work would 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 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 by the Domain Name System Operations Analysis and Research Center (DNS-OARC) that are often supplied by root server operators.

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 data. In the case of anonymizing DNS requests, the part of the data that is anonymized is the source IP address.

For DNS query data such as DITL data, there is usually a timestamp. 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.

One common example of the desire for anonymization is for the DITL data, which is 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.

2.1 Benefits and Drawbacks of Harmonization of Anonymization

If each Root Server Operator (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. 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. Even if RSOs use the same form of anonymization but use different random keys, datasets cannot be compared across RSOs.

A drawback of harmonized anonymization is that all effective methods that allow correlation across RSOs require the RSOs to share a secret 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

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

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 keys across RSOs. If there was a standard way of anonymizing IP addresses and keys, researchers would benefit because they would have much better insight into how resolvers use the entire root server system. Although it is impossible to estimate the value of this increased insight, it is generally assumed that better research for the root server system 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 Methods for Anonymizing IP Addresses

We propose four primary ways to anonymize the IP addresses in a sequence of queries:

1. Cryptographically mix each new address seen with AES-128 and a random 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 random value)
3. Cryptographically encrypt each new IPv4 address seen with ipcrypt and a random value, and encrypt each new IPv6 address with AES-128
4. Assign each new address seen to an existing list of mapped addresses

These methods 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 due to a Denial of Service (DoS) attack where the source addresses are randomized. For example, in the 2017 DITL collection, even in the largest dataset, there were fewer than 5 million unique IP addresses. Thus, the address set is relatively small and each address repeats roughly a thousand times per address.

Only the first three of these methods are considered for deployment by RSOs. The fourth method, described in Appendix C, works poorly with data that has randomized source addresses, and such addresses are common when faced with a Distributed Denial of Service (DDoS) attack.

If long-term anonymization is desired, the random key must be kept secret. In the current Internet, the vast majority of DNS requests seen at root servers are over IPv4, so the anonymized data would be an IPv4 address. Anonymizing data that only has a small number of possible values using cryptographic functions is impossible without adding randomness to the data. 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 random number 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 mixing are encrypting with a random key, encrypting with a random 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.

Both mixing full addresses and mixing bit-by-bit have two significant disadvantages:

- Unless the random value that is mixed into the address is shared across datasets, addresses in each dataset will result in different outputs, so harmonization across such datasets is impossible.
- Efficient use of either method relies on looking up already-seen values, and thus can become inefficient during a DoS attack where the source addresses are randomized and so often very numerous. If the cache of already-seen values gets too large for efficient searching, the cache can be ignored but then each address must be mixed.

Appendix A contains pseudocode for a system that embodies the three mechanisms described in this section.

3.1 Mixing Full Addresses with Truncation

Addresses can be anonymized by encrypting the address using AES-128 with a random key. The same key is used for an entire dataset.

When a cryptographic function has a result longer than the input, the result must be truncated if the outputs are to be the same size as the inputs. For mixing full addresses, 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. (Because of this, the IPv6 addresses can be decrypted using the same key.)

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, with C=*n* (1 - (1 -

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

$1/N)^{(n-1)}$) for a mean of C collisions from n unique inputs into a space of N addresses.[6] Thus, if 4,000,000 IPv4 addresses (about the number seen at a root letter per day) are mapped to 2^{32} possible values, on average there will be about 3,700 collisions, and so a statistic like "number of unique sources" will be very slightly underestimated if computed using anonymized data.

This method has the following advantages:

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

3.2 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: 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 advantages:

- It has been used by multiple organizations for over a decade.
- The one-to-one correlation may be more desirable than the collisions that come with truncating IPv4 addresses after mixing full addresses.
- Prefix preservation may be useful to some observers of the output.

3.3 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 random key. The same key is used for an entire dataset. The resulting addresses can be decrypted with the same key.

---

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

[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 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.

This method has the following advantage:

- The description of the method is relatively simple.

4 Saving Additional Information When Anonymizing

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. A side effect of anonymization is that it destroys any possibility of establishing the origin AS 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.

5 Recommendations

Recommendation 1: RSSAC should consider the advantages and disadvantages of harmonization of anonymization for DITL Data.
RSSAC has to decide whether to pursue harmonization of anonymization data that comes from multiple operators, particularly the DITL 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 random value that will invalidate the anonymization if it is later revealed.

Even if RSSAC decides not to harmonize with sharing of secret salt or keys, 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: If harmonization of anonymization is desired, choose a specific anonymization strategy.
Any of the three proposals given in Section 3 of this document can be used as the anonymization specification.

Recommendation 3: Autonomous System (AS) numbers of original addresses should be made available with the anonymized data.

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.

6 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.

6.1 Acknowledgments

RSSAC Caucus Members

Jaap Akkerhuis
John Heidemann
Paul Hoffman
Geoff Huston
Lars-Johan Liman

ICANN Support Staff

Andrew McConachie (editor)
Kathy Schnitt

6.2 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

6.3 Dissents

There were no dissents.

6.4 Withdrawals

There were no withdrawals.

6.5 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 earlier in the document. The input is a stream of IPv4 and IPv6 addresses in wire format.

```
Startup
- 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[11]
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 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:[12]
            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 3.1)
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 3.2)
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):[13]
        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

Mixing function for full addresses with ipcrypt or AES-128 (Section 3.3)
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)
```

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

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[14], and multiple implementations of it exist. As of 2017, 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 (1) deterministic, depending only on the input address and salt, (2) 1:1, so every input address has exactly one output address, with no collisions, in the same number of bits, (3) prefix preserving (so two input addresses with the same prefix of *i* bits have the same *i* bits in the output.

These properties are convenient, since (1) memory requirements are around tens of bytes if one uses a cache of translations, it is only to improve performance, not for correctness, (2) eliminating collisions reduces concerns that the statistics on the anonymized addresses will be different than those on clear addresses, (3) one can detect spoofed addresses from the same subnet in anonymized addresses.

Conceptually:

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

Where C'_i represents the *i*th anonymized bit of the output, $C_{[0..i]}$ 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. Note that implementations do not directly implement this formula.

A C implementation with test inputs is at the Dnsanon project[15].

[14] *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

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

Appendix C: Anonymization by Assigning New Addresses to a List

RSSAC considered other methods but did not think they would work under expected conditions in the DNS. One method that might seem sensible, but fails under a common scenario, is assigning each new address seen to an existing list of mapped addresses. This is the easiest method for anonymization, but fails 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 advantages:

- The result will always be a one-to-one match with the source.
- The description of the method is extremely simple.

This method has the following disadvantage:

- This method relies on storing and looking up already seen values, and thus will become inefficient in the face of a DoS 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.