Semantics-Preserving Encryption for Computer Networking Related Data Types

Gergő Ládi
Laboratory of Cryptography and System Security
Department of Networked Systems and Services
Budapest University of Technology and Economics
Budapest, Hungary
me@gergoladi.me

Abstract — **Semantics-preserving encryption methods are encryption methods that not only preserve the format (data structure) of the input, but also a set of additional properties that are desired to be preserved (for example, transforming an IP address into another from the same subnet). Such methods may be used to anonymize logs or otherwise hide potentially sensitive information from third parties, while preserving characteristics that are essential for a given purpose. This paper presents tuneable semantics-preserving encryption methods that may be applied to common computer networking related data types such as IPv4, IPv6, and MAC addresses.**

Keywords – semantics-preserving encryption; format-preserving encryption; networking; data type; MAC address; IPv4 address; IPv6 address; TCP port; UDP port; privacy; log anonymization;

I. Introduction

Format-preserving encryption methods, i.e. methods that – using a secret key – reversibly map an element of a set to another element of the same set, are most often used when data is to be shared with untrusted third parties that perform format checks on the input. In these cases, the output of a traditional encryption function would not be accepted by the third party, as the format checks would fail. In recent years, format-preserving encryption methods have been proposed for simple data types, such as integers [1] or *DateTime*s [2], as well as more complex ones, such as JPEG or PNG images ([3], [4]).

Computer networking related data types (e.g. IPv4 and IPv6 addresses) are usually treated as binary data, which works fine as long as only basic (length) checks are in place. However, several addresses and address types have special or additional meanings, therefore, an address encrypted into one of these may fail more sophisticated checks. This is where semantics-preserving encryption methods enter the picture.

Using semantics-preserving encryption, it can be ensured that alongside encrypting in a format-preserving manner, a set of desired properties are also transferred to the ciphertext. This property set should be tailored to each use case to contain the least amount of information to make validation checks pass, without revealing unnecessarily much. In addition, semantics-preserving encryption can also be used to anonymize logs containing computer network related data types. Then, these logs may be shared with third parties, without having to worry about leaking sensitive information such as the network topology or internal addressing practices. Since this is encryption, the data owner may reverse the encryption using the key if needed.

This paper enumerates the most frequently used computer networking related data types, providing a semantics-preserving encryption method for each one.

II. Related Works

There are currently no known papers that address the topic of encrypting MAC addresses in a semantics-preserving manner. For IPv4 addresses, there exists *Tcpdpriv* [5], an open source anonymizer that may be configured to preserve the first N bits of the address, but it does not consider special addresses and address spaces. Xu et al. [6] show that it also has the drawbacks of not being consistent between different traces, and that it has a rather large memory footprint (320 MBs of memory for a trace with 10 million unique IPv4 addresses). *Tcpdpriv* did not originally support IPv6 addresses, but was extended by Cho et al. [7] to do so. There exist additional proposals ([8], [9], [10]) for transforming IPv4/IPv6 addresses, but none of them goes further than preserving prefixes, and some of these methods are not even reversible.

III. Mathematical Background

This section explains the mathematical background that is needed to understand how format-preserving encryption, the basis of semantics-preserving encryption works.

All of the proposed algorithms herein rely on the existence of a function $\mathcal{F}$, where $\mathcal{F}$ is a format-preserving transformation (encryption) as defined by Black and Rogaway [11], where $\mathcal{K}$ is the encryption key, and $\mathcal{M}$ is the message space:

$$\mathcal{F}: \mathcal{K} \times \mathcal{M} \to \mathcal{M} \tag{1}$$

For $\mathcal{F}$, an inverse function $\mathcal{F}^{-1}$ must also exist such that if (2) holds true, then (3) is also true:

$$\forall k \in \mathcal{K};\ \forall\ m_1, m_2 \in \mathcal{M};\ \mathcal{F}(k, m_1) = m_2 \tag{2}$$

$$\mathcal{F}^{-1}(k, m_2) = m_1 \tag{3}$$

In their paper, *Format Preserving Encryption*, Bellare et al. [12] construct functions that work in this manner, but $\mathcal{M}$ can only be a set of integers of arbitrary width (often referred to as

BigIntegers). These functions can be used as the basis of a rank-then-encipher approach that may be used to construct others that work on sets of arbitrary data types. Rank-then-encipher methods have three distinct steps:

- Ranking: consider a set $\mathcal{M}$, where $\mathcal{M}$ can now contain elements of arbitrary data types (not just integers). Define a bijective mapping G (eq. 4):

  $$G: \mathcal{M} \rightarrow \{0, \ldots, |\mathcal{M}|\text{-}1\} \subset \mathbb{N} \qquad (4)$$

  Apply G to the value m to be encrypted:

  $$rank = G(m \in \mathcal{M}) \qquad (5)$$

  Note: In practice, G can be a simple function that maps each element of $\mathcal{M}$ to its index (position) in the set (i.e. the first element maps to 0, the second one to 1, and the last one to $|\mathcal{M}|$-1).

- Encipherment: use one of Bellare's functions (with any key of our choice) to encrypt the rank from the previous step:

  $$enc\_index = FE2\_Enc(k \in \mathcal{K}, rank, |\mathcal{M}|\text{-}1) \qquad (6)$$

- Unranking: since G is bijective, it has an inverse G^{-1}. Applying G^{-1} to *enc_index* yields an element of $\mathcal{M}$, thus we have constructed an $\mathcal{M} \rightarrow \mathcal{M}$ function.

  Note: If the above-mentioned simple function was used for ranking, then G^{-1} essentially just returns the *enc_index*th element of $\mathcal{M}$.

Decryption works similarly:

- Ranking: apply G to the value c to be decrypted, just as in eq. (5).

- Decipherment: use the inverse of Bellare's functions with the same key k as in eq. (6) to decrypt the *rank* from the previous step:

  $$dec\_index = FE2\_Dec(k, rank, |\mathcal{M}|\text{-}1) \qquad (7)$$

- Unranking: applying G^{-1} to *dec_index* returns the original m.

Making use of the rank-then-encipher method, it is now possible to construct algorithms that in addition to preserving the format of the input, they also preserve semantics.

For the rest of this paper, unless otherwise mentioned, encryption will refer to encrypting in a format-preserving manner, while decryption will refer to reversing said encryption.

IV. Preserving Semantics for Computer Networking Related Data Types

In order to design semantics-preserving methods, it is imperative to understand how each kind of address is structured and which addresses or address ranges have special meanings (this is what needs to be preserved). This section aims to describe the most commonly used networking-related data types, then proposes algorithms that may be used to encrypt said data types in a semantics-preserving manner.

A. MAC Addresses

Media Access Control (MAC) addresses are 48-bit physical addresses that are mostly used by network devices running IEEE 802-based network technologies, such as Ethernet, WiFi, or Bluetooth. The 6-byte address comprises of two 3-byte parts (see Figure 1.): the Organizationally Unique Identifier (OUI), and the Network Interface Controller specific identifier [13].

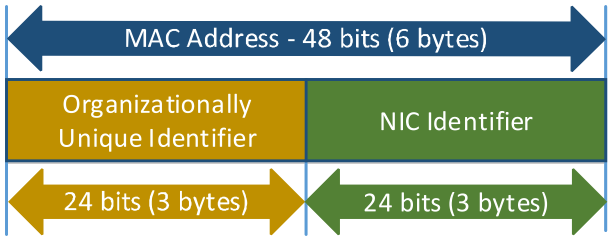
MAC Address - 48 bits (6 bytes)
Organizationally Unique Identifier
NIC Identifier
24 bits (3 bytes)
24 bits (3 bytes)

Figure 1. Structure of a MAC address

Generally, OUIs are allocated by the IEEE Registration Authority, and are used to uniquely identify the company that manufactured the network interface card, while NIC identifiers are assigned by the manufacturers under the requirement that every single NIC ever produced should be assigned a unique identifier [13]. The first and second least significant bits of the most significant byte of the OUI have special meanings (Figure 2.): the least significant bit (**I**ndividual/**G**roup bit) denotes whether the MAC address is a unicast (value of 0) or multicast/broadcast (value of 1) address, and the second least significant bit (**G**lobal/**L**ocal bit) denotes whether the address is locally administered (overridden by the system administrator) (value of 1) or not (value of 0).

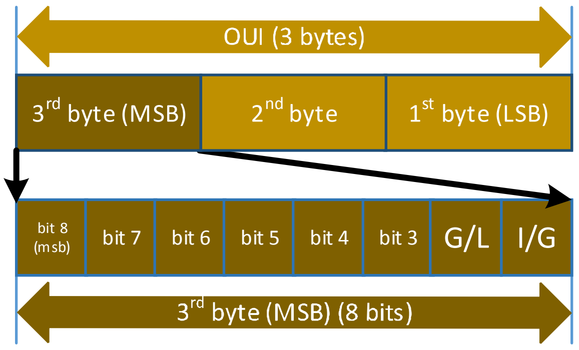

Figure 2. Position of the G/L and I/G bits in the OUI

To preserve semantics, the value of these two special bits should be preserved. For locally administered addresses, the remaining 46 bits of the address may be encrypted without restrictions, whereas for unique addresses, two different strategies may be used:

- Leave the OUI as is, and only encrypt the NIC identifier. This ensures that the encrypted address will appear to be one from the same vendor as the original.

TABLE I. SPECIAL-USE MAC ADDRESS BLOCKS

Block start	Block end	Defined in	Purpose
00:00:5E:00:00:00	00:00:5E:00:00:FF	RFC 7042	Reserved, ratification required for assignment
00:00:5E:00:01:00	00:00:5E:00:01:FF	RFC 5798	Virtual Router Redundancy Protocol (VRRP) for IPv4 networks
00:00:5E:00:02:00	00:00:5E:00:02:FF	RFC 5798	Virtual Router Redundancy Protocol (VRRP) for IPv6 networks
00:00:5E:00:52:00	00:00:5E:00:52:FF	RFC 7042	Reserved for small allocations (3 addresses currently allocated [14])
00:00:5E:00:53:00	00:00:5E:00:53:FF	RFC 7042	For use in documentation only (unicast)
01:00:5E:00:00:00	01:00:5E:7F:FF:FF	RFC 1112	IPv4 multicasting
01:00:5E:80:00:00	01:00:5E:8F:FF:FF	RFC 5332	Multi-Protocol Label Switching (multicast)
01:00:5E:90:00:00	01:00:5E:90:00:FF	RFC 7042	Reserved for small allocations (4 addresses currently allocated [14])
01:00:5E:90:10:00	01:00:5E:90:10:FF	RFC 7042	For use in documentation only (multicast)
01:80:C2:00:00:00	01:80:C2:FF:FF:FF	802.1D	Reserved for use in the 802.1D IEEE standard for MAC bridges
33:33:00:00:00:00	33:33:FF:FF:FF:FF	RFC 2464	IPv6 multicasting

- Make a list of valid OUIs[1], then use a rank-then-encipher function to encrypt the OUI. Encrypt the NIC identifier separately. This ensures that the encrypted address will appear to be from a valid manufacturer.

Some address blocks, instead of being assigned to vendors, are reserved for use in specific protocols (see Table I.). In order to encrypt addresses from any of the reserved blocks, it is important to understand how those addresses are allocated and used. For the VRRP MAC ranges, the last octet comes from the VRID[2], which also appears in upper-layer packets, therefore care should be taken to encrypt it consistently everywhere where it appears. Small allocation ranges at present have only single address allocations with a specific purpose [14], these should not be changed during encryption. The ranges reserved for documentation should never be used in a non-hypothetical network, therefore these need not be encrypted (but if desired, the last octet may be encrypted freely). For the MAC range used by MPLS, the last 20 bits of the address are calculated from MPLS labels[3], meaning that these should only be encrypted along with the labels to preserve consistency. The address range reserved for 802.1D consists of single allocations with a specific purpose [15], these addresses should be left intact. As for the IPv4 and IPv6 multicast MAC ranges, the lowest 23 and 32 bits (respectively) are derived from the IP addresses, therefore these should not be directly encrypted, but instead, be recalculated when the IP addresses are encrypted.

There are also two MAC addresses with a special meaning: the so-called *unspecified* address 00:00:00:00:00:00 (with all the bits set to 0), and the broadcast address ff:ff:ff:ff:ff:ff (with all the bits set to 1). These should not be changed during encryption.

[1] The list changes frequently, and is too long to be included here. The latest version may be acquired from https://standards-oui.ieee.org/oui/oui.txt.

[2] Virtual Router Identifier, a number used by the VRRP protocol.

[3] A 20-bit field used for routing packets in MPLS networks.

Considering the previous observations, the following algorithm can be given to encrypt MAC addresses in a semantics-preserving manner:

1) If the input is the unspecified address or the broadcast address, return the input unchanged.
2) If the input is in one of the special-use blocks, encrypt the address (and possibly other fields in the packet, as needed) based on the previous considerations.
3) If the address has the locally administered bit set, consider the address without the I/G and G/L bits as a 46-bit integer, then encrypt it. In the resulting ciphertext, re-insert the original I/G and G/L bits in the appropriate positions. Return this 48-bit number as a MAC address.
4) Otherwise, use one of the OUI encryption strategies mentioned earlier, then encrypt the NIC identifier. Construct a new MAC address from these two ciphertexts.

The decryption process can also be described in the same fashion:

1) If the input is the unspecified address or the broadcast address, return the input unchanged.
2) If the input is in one of the special-use blocks, decrypt the address and any other fields that were changed during encryption.
3) If the locally administered bit is set, consider the address without the I/G and G/L bits as a 46-bit integer, then decrypt it. In the resulting plaintext, re-insert the original I/G and G/L bits in the appropriate position. Return this 48-bit number as a MAC address.
4) Otherwise, we have to know which strategy was used for encrypting the OUI. If the OUI was

TABLE II. SPECIAL-USE IPV4 ADDRESS BLOCKS

Block	Defined in	Purpose	Block	Defined in	Purpose
0.0.0.0/8	RFC 1122	*Unspecified* address	192.0.2.0/24	RFC 5737	Documentation (TEST-NET-1)
10.0.0.0/8	RFC 1918	Private use	192.168.0.0/16	RFC 1918	Private use
100.64.0.0/10	RFC 6598	Carrier-grade NAT	198.18.0.0/15	RFC 2544	Benchmarking
127.0.0.0/8	RFC 1122	Loopback addresses	198.51.100.0/24	RFC 5737	Documentation (TEST-NET-2)
169.254.0.0/16	RFC 3927	Link-local addresses	203.0.113.0/24	RFC 5737	Documentation (TEST-NET-3)
172.16.0.0/12	RFC 1918	Private use	240.0.0.0/4	RFC 1112	Reserved
192.0.0.0/24	RFC 6890	IETF assignments	255.255.255.255/32	RFC 1122	Limited broadcast

preserved, keep the OUI and decrypt the NIC identifier. If the OUI was not preserved, reverse the rank-then-encipher method on the OUI over the same list that was used during encryption, then decrypt the NIC identifier.

B. *IPv4 Addresses*

Internet Protocol version 4 (IPv4) addresses are 32-bit logical addresses that are used by devices to communicate over IPv4 networks. In order to encrypt IP addresses in a semantics-preserving manner, it should be understood how IP addresses are allocated, formed, and used. The addresses are assigned in blocks by the Internet Assigned Numbers Authority (IANA), either to be leased to customers through other organizations, or to be used for special purposes [16].

IPv4 addresses are usually written in the form A.B.C.D, where A, B, C, and D are integers between 0 and 255. Each address is made up of network bits and host bits – the former identify the network, the latter a host. The number of network bits may be given in the Classless Inter-Domain Routing (CIDR) notation, where it is appended to the IP address following a forward slash (e.g. 127.0.0.1/8), or it may be given in the form of a subnet mask (e.g. 255.0.0.0), where each 1-bit in the mask means that the corresponding bit in the IP address identifies the network. The first and last addresses of each subnet are special: the first one is the *network address*, while the last one is the *broadcast address* for that subnet – these may not be assigned to hosts. Networks usually have a *default gateway* – a host to which other hosts in the same network can send packets that are addressed to hosts on different networks. While there is no standard for this, default gateways are usually assigned the first or last assignable address of the subnet.

During encryption, care should be taken never to transform an IP address that is not a network address into one that is, to transform an address that is not a broadcast address into one that is, and vice versa. If the size of the network is known and is relevant (e.g. when anonymizing an internal network), it should be ensured that if any two hosts were in the same subnet before encryption, they should remain in the same subnet after encryption (but the subnet itself, of course, may be different). If there was a default gateway, and it had the first or the last address, it should have the first or last address in the anonymized (encrypted) network as well.

When encrypting IP addresses allocated for special purposes (see Table II.), to preserve semantics, it should be understood how the addresses in those blocks are used. Of the 0.0.0.0/8 block, usually only the address 0.0.0.0 is used – this should not be changed during encryption, while the rest of the addresses in the block should encrypt into other addresses in the block. The same applies for the loopback range – usually only the first address, 127.0.0.1 is used, this should not be transformed, but the rest of the addresses in the block should be. Addresses in the 100.64.0.0/10 block should be transformed into addresses in the same block. Private use blocks are freely assignable by network administrators, these addresses are used in internal networks and do not appear on the internet. Subnets in these blocks should only be transformed into other subnets in these blocks, but the subnets do not have to remain in the same private block (for example, transforming 192.168.10.0/24 into 172.16.20.0/24 is allowed). Addresses in the link-local block should be transformed into addresses in the same block, except for the first and last 256 addresses, which are reserved for later allocation [17], and should be left as is. Addresses in the IETF-reserved block should not be transformed. Addresses in the documentation blocks should never be used in actual networks. These may be transformed into addresses from the same block or may be left as is; in addition, each block as a whole may be transformed into one of the other blocks. Addresses in the benchmarking block may be left as is, or may be transformed into addresses from the same block. Addresses from the 240.0.0.0/4 (reserved) block should never appear anywhere, deciding how to deal with these is left for the implementor of the encryption algorithm. Finally, the limited broadcast address, 255.255.255.255 should be left intact.

There also exists a block, 224.0.0.0/4 that is used for point-to-multipoint (multicast) traffic [18]. The block contains several single-address allocations, range allocations, as well as ranges of reserved and unassigned addresses. These are not listed in this paper due to lack of space and frequency of updates to the allocations. When encrypting addresses from this block, single-address allocations should not be transformed, while addresses from range allocations should be encrypted into other addresses from the same range. Reserved addresses and address ranges should not appear in logs, but may be treated as single and range allocations, respectively. Unassigned addresses and ranges should be treated as single and range allocations, respectively.

TABLE III. SPECIAL-USE IPV6 ADDRESS BLOCKS

Block	Defined in	Purpose	Block	Defined in	Purpose
::/128	RFC 4291	*Unspecified* address	2001::/32	RFC 4380	TEREDO (4-to-6 transition technology)
::1/128	RFC 4291	Loopback address	2001:2::/48	RFC 5180	Benchmarking
::ffff:0:0/96	RFC 4291	IPv4-mapped addresses	2001:20::/28	RFC 7343	ORCHIDv2
64:ff9b::/96	RFC 6052	IPv4-IPv6 translation	2001:db8::/32	RFC 3849	Documentation
64:ff9b:1::/48	RFC 8215	IPv4-IPv6 translation	2002::/16	RFC 3056	6to4 (transition technology)
100::/64	RFC 6666	Discard addresses	fc00::/7	RFC 4193	Private use (unique local)
2001::/23	RFC 2928	IETF assignments	fe80::/10	RFC 4291	Link-local unicast

Transforming addresses into other addresses in the same block is done by leaving the network bits as is, then using a rank-then-encipher function on the host bits (section III.), modified to exclude the network and broadcast addresses from the mapping. Where addresses have to be transformed among a list of blocks, a rank-then-encipher function may be used on the list index for the network bits, and the host bits can be transformed as explained previously.

Considering all of the above, encrypting an IPv4 address in a semantics-preserving manner can be done using the following steps:

1) If the address must be left intact, leave it as is.
2) If the address is in one of the special-use blocks, encrypt it based on the previous considerations.
3) If the number of network bits is known, leave or encrypt the network bits as needed, then encrypt the host bits as explained previously.
4) Otherwise, the address is from a public, non-special range and nothing else is known about it, so make a mapping of addresses in non-special ranges, then use a rank-then-encipher function to encrypt the address.

Decryption is as follows:

1) If the address was to be left intact, do nothing.
2) If the address is in one of the special-use blocks, decrypt it based on the previous considerations.
3) If the number of network bits is known, decrypt the network bits if they were encrypted previously, then decrypt the host bits.
4) Otherwise, the address was from a public, non-special range and nothing was known about it, so use the same mapping as what was used for encryption, then use the rank-then-encipher function to decrypt the address.

C. *IPv6 Addresses*

Internet Protocol version 6 (IPv4) addresses are 128-bit logical addresses that are used by network devices to communicate over IPv6 networks. Addresses are written in the form of A:B:C:D:E:F:G:H/n, where A to H are 16-bit blocks represented as hexadecimal values (e.g. fe80) and *n* is the prefix length, having the same function as subnet masks in IPv4. Since the addressing works similarly to IPv4 addresses, the algorithms for encrypting and decrypting these addresses semantics-preservingly are also similar, with a few notable exceptions.

In IPv6, there is no broadcasting. As a result, there are also no broadcast addresses. This means that the last address of a prefix no longer has to be treated in a special way.

As with IPv4, some address blocks (see Table III.) are reserved for special use [19], these need to be treated differently. The unspecified and the loopback address ranges have been reduced to one address each, these should not be changed during encryption. The documentation, the benchmarking, the IETF-assigned range, and the private block should be treated as they were in IPv4. Addresses from the discard block and the link-local unicast block should be transformed into other addresses from the same block. ORCHIDv2 addresses are only to be used as non-routable overlay addresses, thus should not appear in regular traffic logs – in case they do, they need to be changed into a valid ORCHID identifier (this is not discussed further in this paper).

Addresses in the IPv4-mapped address block, the TEREDO block, the IPv4-IPv6 translation blocks, and the 6to4 block contain IPv4 addresses, either embedded in the IPv6 address, or in the payload. When encrypting addresses from these blocks, transform the embedded address instead of the IPv6 address.

In addition, IPv6 has a multicast address block, ff00::/8 [20]. This should be treated in the same way as addresses in the IPv4 multicast block.

Finally, some IPv6 addresses are generated from the interfaces' MAC address using the EUI-64 algorithm [21]. For cases when both the MAC and the IPv6 addresses are to be encrypted and the IPv6 address appears to have been calculated from the MAC address, encrypt the MAC address first, then recalculate the IPv6 address using the resulting ciphertext.

D. *TCP and UDP Port Numbers*

Port numbers are 2-byte numeric identifiers that are used to identify network services running on hosts. Port numbers belong to one of the three categories [22]:

1) Well-known ports (from 0 to 1023),
2) Registered ports (from 1024 to 49151),
3) Dynamic (also referred to as private) ports (from 49152 to 65535)

The list of well-known and registered ports is maintained by IANA. Ports in these ranges generally indicate a specific service running on a host (e.g. TCP port 25 identifies an SMTP service that can be used to send mail). However, this list is a recommendation, not a requirement, and any service may be set to listen on any port. It is also worth noting that some ports in these ranges are unassigned or reserved.

To encrypt port numbers, the following algorithm can be used:

1) If the port is a dynamic port, encrypt it into another from the dynamic range.
2) If it is a well-known or registered port that is not unassigned or unallocated, leave it as is.
3) Otherwise, make a list of unassigned and unregistered ports, then use the rank-then-encipher approach to encrypt it.

The process of decryption is not described due to the lack of space, but it may be easily inferred by looking at the algorithm used for encryption.

V. Further Considerations

When relying on the lists of reserved OUI blocks for MAC addresses and reserved IP blocks for IPv4/IPv6 addresses, one should check the IANA allocation tables ([14], [16], [18]-[20]) to ensure that the latest, most up-to-date list is used.

Conclusion

Semantics-preserving encryption is useful for a variety of use cases where data has to be encrypted or anonymized before being shared with third parties while still preserving certain properties of the source. Despite having practical uses, no elaborate solutions have been offered to date. The aim of this work was to demonstrate what semantical information the various computer networking related data types carry, then present algorithms that can be used to encrypt these data types while preserving this semantical information.

References

[1] M. Bellare, T. Ristenpart, P. Rogaway, T. Stegers, “Format-Preserving Encryption,” Cryptology ePrint Archive: Report 2009/251 (online: https://eprint.iacr.org/2009/251), 2009

[2] Z. Liu, C. Jia, J. Li, X. Cheng, “Format-preserving encryption for DateTime,” IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), 2010

[3] A. Unterweger, A. Uhl, “Length-preserving bit-stream-based JPEG encryption,” MM&Sec '12 Proceedings of the 14th ACM multimedia and security workshop, 2012, pp. 85-90.

[4] Z. Liu, M. Li, X.-Y. You, C. Jia, “Format-preserving encryption for PNG image,” Beijing Ligong Daxue Xuebao/Transaction of Beijing Institute of Technology, 2013

[5] G. Minshall, Tcpdpriv, http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html (online), 1996

[6] J. Xu, J. Fan, M. Ammar, S. B. Moon, “On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization”, IMW '01 Proceedings of the 1st ACM SIGCOMM Workshop on Internet measurement, 2001, pp. 263-266

[7] K. Cho, K. Mitsuya, A. Kato, “Traffic Data Repository at the Wide Project,” Proceedings of the FREENIX Track: 2000 USENIX Annual Technical Conference, 2000

[8] A. J. Slagell, Y. Li, K. Luo, “Sharing Network Logs for Computer Forensics: A New Tool for the Anonymization of NetFlow Records,” Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, 2005

[9] D. Plonka, A. Berger, “kIP: a Measured Approach to IPv6 Address Anonymization,” Measurement and Analysis for Protocols Research Group (MAPRG) meeting, 2017

[10] P. Zhang, X. Huang, M. Luo, C. Ning, Y. Ma, “Fast restorable prefix-preserving IP address anonymization for IPv4/IPv6,” The Journal of China Universities of Posts and Telecommunications, 2010, pp. 93-98

[11] J. Black, P. Rogaway, “Ciphers with arbitrary finite domains,” RSA-CT, 2002, p. 114.

[12] M. Bellare, T. Ristenpart, P. Rogaway, T. Stegers, “Format-Preserving Encryption,” Cryptology ePrint Archive: Report 2009/251 (online: https://eprint.iacr.org/2009/251), 2009

[13] IEEE Standards Association, “802-2014 – IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture,” ISBN 978-0-7381-9219-2, pp. 22-27

[14] Internet Assigned Numbers Authority, “Assignments, Ethernet Numbers,” (online), https://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml, retrieved: 2017/09/22.

[15] IEEE Standards Association, “802.1D – IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges,” ISBN 0-7381-3982-3 SS95213, p. 51.

[16] Internet Assigned Numbers Authority, “IANA IPv4 Special-Purpose Address Registry,” (online), https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml, retrieved: 2017/09/23.

[17] S. Chesire, B. Aboba, E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses,” (online), https://tools.ietf.org/html/rfc3927, Internet Engineering Task Force (IETF) RFC 3927, 2005, pp. 3, 24.

[18] Internet Assigned Numbers Authority, “IPv4 Multicast Address Space Registry,” (online), https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml, retrieved: 2017/09/23.

[19] Internet Assigned Numbers Authority, “IANA IPv6 Special-Purpose Address Registry,” (online), https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml, retrieved: 2017/09/24.

[20] Internet Assigned Numbers Authority, “Internet Protocol Version 6 Address Space,” (online), https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml, retrieved: 2017/09/24.

[21] R. Hinden, S. Deering, “IP Version 6 Addressing Architecture,” (online), https://tools.ietf.org/html/rfc2373, Internet Engineering Task Force (IETF) RFC 2373, 1998, p. 18.

[22] Internet Assigned Numbers Authority, “Service Name and Transport Protocol Port Number Registry,” (online), https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, retrieved: 2017/09/24.