Privacy Enhanced Interface Identifiers in IPv6

Stephen Odero
School of Science, Engineering and Environment
University of Salford
Manchester, UK
s.o.odero@edu.salford.ac.uk

Tooska Dargahi
School of Science, Engineering and Environment
University of Salford
Manchester, UK
t.dargahi@salford.ac.uk

Haifa Takruri
School of Science, Engineering and Environment
University of Salford
Manchester, UK
h.takruri-rizk@salford.ac.uk

Abstract— The Internet Protocol Version 6 (IPV6) proposed to replace IPV4 to solve scalability challenges and improve quality of service and security. Current implementation of IPv6 uses static value that is determined from the Media Access Control (MAC) address as the Interface Identifier (IID). This results in a deterministic IID for each user that is the same regardless of any network changes. This provides an eavesdropper with the ability to easily track the physical location of the communicating nodes using simple tools, such as ping and traceroute. Moreover, this address generation method provides a means to correlate network traffic with a specific user which can be achieved by filtering the IID and traffic analysis. These serious privacy breaches need to be addressed before widespread deployment of IPv6. In this paper we propose a privacy-enhanced method for generating IID which combines different network parameters. The proposed method generates non-deterministic IIDs that is resistance against correlation attack. We validate our approach using Wireshark, ping and traceroute tools and show that our proposed approach achieves better privacy compared to the existing IID generation methods.

I. INTRODUCTION

The explosive growth of the number of Internet users and Internet-connected devices has overwhelmed IPv4 and has driven the development of the next generation Internet protocol, IPv6 [1]. IPv4 32-bit address space provides a maximum of 2^{32} address which is approximately 4.29 billion addresses. With the current world population of over 7 billion people, even if it were possible to use 100% of the IPv4 address space, we would still not be able to provide IP address for everyone. IPv6 on the other hand uses a 128-bit address, which means that we have a maximum of 2^{128} addresses available [1] [2] [3].

The 32-bit IPv4 addresses are generated manually or using Dynamic Host Configuration Protocol (DHCP). However, 128-bit IPv6 address is too long for manual configuration. In IPv6, address generation is divided into stateless address autoconfiguration (SLAAC) and DHCPv6. DHCPv6 is similar to DHCPv4 and all the addresses are managed centrally by a DHCP server [3] [4]. SLAAC uses modified Extended Unique Identifier (EUI) algorithm to generate the Interface Identifier (IID) portion of the IPv6 address (this has been explained in RFC4941) [5]. EUI-64 uses 48-bit MAC address of an interface to generate 64-bit modified EUI-64 for the IID. As indicated in RFC4941 [6], this address generation method could allow an adversary to analyze packet content, packet size and packet timing, however the privacy implications that arise from SLAAC have not been addressed in that document.

The use of MAC address in IID generation has serious privacy concern. Considering that MAC is the unique identifier of each network-connected device, having access to the MAC address of the users an adversary will be able to track the users. IIDs should be unique, producing an IPv6 address that is unique within all its applicable scope (local and global) [7]. This type of privacy attack is not possible in IPv4 addresses that is restricted to local subnet. The location of a node in IPv4 is hidden due to the usage of Dynamic Host Configuration Protocol (DHCP), which generates addresses based upon availability. Furthermore, the use of network address translation (NAT) unintentionally protects a host identity by hiding it within a private address space that is not viewable externally [4] .

A user's privacy in IPv6 implementation can be breached through traffic analysis by correlating traffic captured from a specific IID. This kind of analysis is also possible in IPv4, but only within the lifespan of an address, since DHCP addresses change. IPv6 on the other hand permits correlation over multiple sessions due to the deterministic address that binds users to each of their packets [4]. The usage of temporary IIDs (RFC4941 [6]) will not completely solve the problem because an adversary can still actively probe multiple subnets for a certain IID. A node does not change its IID when moving to a new network prefix. Thus, tracking a host remains possible within the lifetime of a temporary identifier (in 24 hours). RFC7217 has proposed the inclusion of different network parameters when generating IIDs, a solution which is more secure although it has not been implemented by manufacturers as reported in RFC7217 [8]

A. Problem Statement

Figure 1 shows an example of IID generated from MAC address as reported in [5] . As it can be seen, an IPv6 IID can be generated for a device with a MAC address of *00:26:08:e6:24:e6* through following steps:

Step 1: Splitting the MAC address in half
002608 | e624e6
Step 2: Inserting a constant *ff:fe* in the middle
002608 ff fee624e6
Step 3: Changing the format to use a colon delimiter
0026:08ff:fee6:24e6
Step 4 Converting the first eight bits to binary
00 → 00000000
Step 5 Flipping the seventh bit 00000000 → 00000010

Step 6 converting these eight bits back to hexadecimals

00000010 → 02

Hence the modified EUI-64 address is *a00:27ff:fe36:daeb*

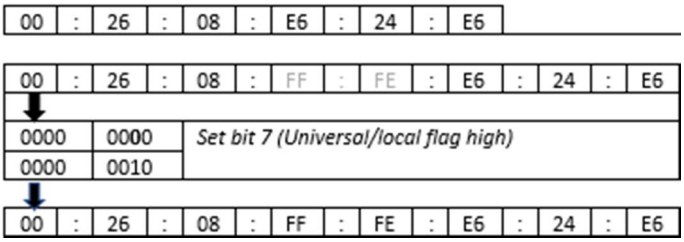

Figure 1. How EUI-64 addresses are generated from MAC Addresses [5].

Many researchers have shown that using the EUI-64 in generating IIDs in IPv6 could compromise users privacy as discussed in RFC4941 [6]. Figure 2 shows a Wireshark packet capture that has an IID that has been generated using EUI-64.

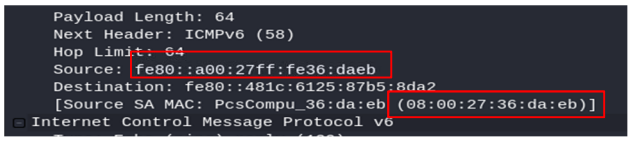

Figure 2. Showing MAC address of the communicating node

A non-changing IID would allow an eavesdropper to correlate unrelated bits of information with a node. According to RFC4882 [9] location privacy is of concern for any roaming device. Any compromise to location privacy could lead to a more targeted profiling of the user activity. An IID that remains unchanged across the network could suggest related activity [10]. While other researchers like [11] [12] have analyzed how different IID generation methods are used against correlation attacks. The most recent works on IID generation (RFC7217 [13] and RFC8064 [14]) propose a new algorithm to generate IID by mixing various packet fields in a pseudorandom function.

In this paper, we first discuss and show that deterministic addresses have serious privacy implications, and an adversary can determine a user's MAC address from the IPv6. We then propose a modification to RFC7217 to include Network_ID as a mandatory filed in the calculation of the IID. We then hash it using SHA256 algorithm (which takes an input of arbitrary length and generates an almost unique 256-bits (32-byte) signature of the text [9] [15]). A large hash makes it more difficult to invert the function and it ensures that the function is collision free [16]. We evaluate the efficiency of the proposed method through experimental analysis.

The rest of the paper is organized as follows. Section 2 provides background and literature review and explains the existing privacy issue. Section 3 explains the proposed approach. Section 4 presents the experimental analysis results, and Section 6 concludes the paper and discusses future work.

II. BACKGROUND AND RELATED WORK

In this section we provide background information on IPv6 address generation and then discuss the existing privacy issue.

The contents of address fields in IPv6 may contain values or end with zero-valued fields. Zeros and ones are all valid values for any address field as indicated in RFC4291 [17] [18]. IPv6 addresses of all types are assigned to interfaces, not nodes/devices. There are three conventional forms for representing IPv6 addresses as text strings. According to [18], the preferred form is X:X:X:X:X:X:X:X, where the 'X's are one to four hexadecimal digits of the eight 16-bit pieces of the address. For example:

ABCD:EF01:2345:6789:ABCD:EF01:2345:6789 or
2001:DB8:0:0:8:800:200C:417A

It is not necessary to write all the leading zeros in an individual field, but there must be one numerical in every field, except for cases where addresses contain long strings of zeros and in order to make writing the zeros easier, a syntax is available to compress the zeros using '**::**'.
For example, the following addresses;

2001:DB8:0:0:8:800:200C:417A *(Unicast Address)*
FF01:0:0:0:0:0:0:101 *(Multicast Address)*
0:0:0:0:0:0:0:1 *(Loopback Address)*
0:0:0:0:0:0:0:0 *(Unsigned Address)*

Can be represented as
2001:DB8:0:0:8:800:200C:417A
FF01::101
::1
::

IPv6 address is divided into two parts, network prefix and host IID as shown in Figure 3.

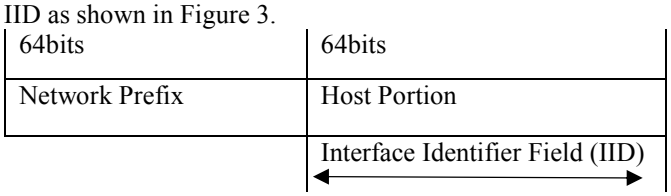

Figure 3. IPv6 Address format. Each group is expressed as four hexadecimal digits separated by colons.

The first 64-bit IPv6 address (i.e., network prefix) is being broadcasted by the router, while the second 64-bit (i.e., IID) is generated through SLAAC for each host. SLAAC allows the network administrator to configure the network and sub network bits of the address while each host automatically configures the IID.

A. IID GENERATION

In this section we explore various means of generating the IPv6 IIDs that are proposed in the literature.

Presently, there are four methods for generating IID [13]: i) EUI-64 ii) Encryption iii) Random iv) Stable.

According to Internet Engineering Task Force (IETF), IPv6 addresses should not be assigned manually, but rather through SLAAC or DHCP. However, how to choose between SLAAC and DHCP and how to validate the uniqueness of the IPv6 is the main challenge [6]. SLAAC and DHCPv6 can be used simultaneously. Hosts use SLAAC to obtain IPv6 address and DHCPv6 to obtain configuration information such as DNS. If a host's address is assigned by DHCPv6, the whole IPv6 address is assigned by the DHCP server address pool.

If the IPv6 address is assigned by the SLAAC, according to [6], hosts configure one or more "stable" addresses composed of network prefix that is advertised by a local router and an IID that embeds a MAC address. The default mechanism used to generate IIDs is modified EUI-64 [13] . This IID is formed by extending the 48-bit MAC address to a 64-bit hexadecimal [4]. SLAAC is meant to be an alternative to modified EUI-64 address generation [9]. Modified EUI-64 method is simple and does not consume too much computing resources of a node; however, this approach exposes the MAC address of the IPv6.
Encryption method on the other hand generates IIDs by hashing the public key and other parameters [13] The network prefix is given as a input of a hash function and a node's cryptographically generated address (CGA) changes from one network to another network to prevent geographic correlation. However, this method is computationally expensive, and it has neither been widely implemented nor deployed [19].
Temporary addresses, with a short lifetime, were introduced to complicate the task of an eavesdropper (RFC4941 [6]). These addresses have a few challenges [6]:

- From the network management point of view, they increase the complexity of event logging, troubleshooting, enforcing Access Control Lists (ACLs)
- They complicate implementation, making it impossible to be implemented in embedded systems.

Where temporary addresses are not used, all that a host is left with is the stable addresses that have been generated from MAC address [13]. Addresses that remain stable for the lifetime of a host's connection to a single subnet are viewed as desirable [14] .These types of addresses may be viewed as beneficial for network management, event logging, enforcement of the (ACLs) and provision of quality services [20] [14]. These kind of addresses as opposed to temporary addresses in RFC4941 [5] allow for long lived TCP connections and are more desirable when performing server like functions.
According to [14], nodes should not employ IPv6 address generation schemes that embed a stable link layer address in the IID. Particularly, nodes should not generate IIDs with schemes specified in RFC2464, RFC2467, RFC2470, RFC2491 etc., rather, follow the recommendations of RFC7217. A new algorithm is therefore proposed by (RFC7217 [13]) for generating "*stable"* IIDs using the following formula.

RID = F(prefix, Net_Iface, Network_ID, DAD_Counter, Secret_Key)
Where:
RID: is a random (but stable) identifier
F(): a pseudorandom function (PRF) that MUST NOT be computable from outside (without the knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret key even when given the samples of the output of F() and knowledge of other input parameters. The F() should produce an output of at least 64 bits [9].
Prefix: this should be the prefix used for SLAAC as learned from ICMPv6 router advertisement message or a link local IPv6 unicast address
Net_Iface: This is an implementation-dependent stable identifier associated with the network interface for which the RID is being generated. The implementation "may" provide a configuration option to select the source of the identifier to be used for the *Net_Iface*
Network_ID: Some of the network specific data that identifies the subnet to which this interface is attached, e.g. IEEE 802.11 Service Set Identifier (SSID). In [21] some ways of generating Network_ID is explained. However, this parameter is *"optional"*. Unknown *Nework_ID* can help in mitigating attacks where a victim host connects to the same subnet as the attacker and the attacker tries to learn the IID used by the victim host for a remote attack [14].
DAD_Counter: This parameter resolves Duplicate Address Detection (DAD) conflicts. It must be initialized to 0 and incremented by 1 for each of the new tentative address configuration as a result of DAD conflict. Implementations that record DAD_Counter in non-volatile memory for {prefix, Net_Iface, Network_ID} must initialize the *DAD_Counter* to the recorded value if such an entry exists in non-volatile memory. On a switch, DAD configuration would be:

#IPv6 nd dad-attempts <0-255>

This command is issued at the global configuration level and configures the number of neighbor solicitations to send when performing duplicate address detection for a unicast address. The default setting of DAD is 3 and could vary from 0 to 255.
Secret_Key: A secret key of at least 128 bits that is not known to an attacker must be initialized to a pseudorandom number according to [22].
After all these processes, an IID is finally obtained by taking as many bits from the RID (as computed in the previous step) as necessary. This generated IID should be compared against reserved IIDs according to [19] and against those IIDs already employed in an address of the same network prefix. In the event that an unacceptable identifier has been generated, the case should be handled the same way as the case of a duplicate address [13].

B. Privacy attacks in the current IID generation methods

IP address plays an important role in the connectivity of a device to a network. On the one hand, it must correctly identify the sender and the receiver so that the message reaches the intended destination. On the other hand, address-based correlation enables attribution of different transactions to the same origin which allows an adversary to gain insight into a user's location and activity [12]. EUI-64 address generation method violates the privacy of devices by exposing their MAC address. The MAC address of the communicating devices will be revealed when devices send or respond to probes [23]. The exposure of MAC address introduces two potential vulnerabilities:

- The ability to identify the manufacturer and model of the device, thereby permitting targeted attacks
- The ability to track and correlate user activity

Geographic location discovery has created a security hole for adversaries to geolocate devices on the network [24]. An adversary can discover the location of a device using GPS by employing a rationalizer to examine the frequency changes caused in the signal strength transmitted by any sensor enabled device e.g. a computer. An adversary can verify the exact

location of a sensor enabled device using an in-built computation inside the device (MAC address) or cloud computing [25].

III. Proposed Approach

In this section, we first describe our considered system and attack models, and then present our modified IID generation algorithm.

A. System model

Our system model is composed of two entities: A user in a network and a remote server. The user wishes to connect to a remote server (e.g. web server) through Internet (as shown in Figure 4). To get an IPv6 address, the router advertises the network prefix and the host (i.e., user device) generates its own IID. This type of network can function with or without a DHCP server because hosts are able to generate their own IPv6 address after getting the network prefix from the router.

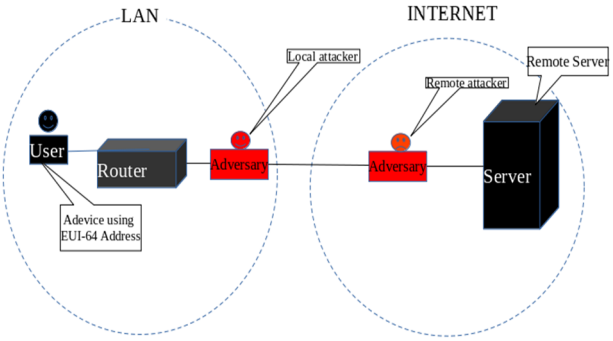

Figure 4. System model of a node communicating through EUI-64 IID

B. Attack Model

This paper focuses on correlation attack. We consider two types of adversaries: 1) an attacker who is inside the user's network (local adversary) and 2) an attacker who is outside the network (remote adversary). Both adversaries can launch packet analysis using Wireshark to analyze the packets from the user and determine the MAC address of the user if the address is generated using EUI method shown in Figure 1. MAC addresses contain both manufacturer and device identities, hence the attacker can discover some of the default vulnerabilities in certain devices. A global attacker on the other hand is able to conduct a correlation attack by first doing a ping sweep to determine the IPv6 addresses that are active in a certain network, after this, the adversary can correlate specific IPv6 addresses that are active and even geolocate them using GPS rationalizer [24] [25]. A global attacker can get a network prefix by doing a "*whois*" query on any of the network Uniform Resource Locator (URL).

C. Proposed IID Generation Algorithm

We propose to make the optional "Network_ID" parameter in RFC 7217 to be a "mandatory" filed when generating an IID. Similar formula as explained in Section II-A will be used including Network prefix, Net_Iface, Network_ID, DAD_Counter and Secret_Key parameters as input of a SHA-256 function to produce a random hexadecimal digit. It should be noted that in RFC 7217 the *f* function has not been explicitly specified and it has been left to the implementor. However, as the devil is in the detail, we emphasize on the usage of SHA-256.

Our proposed method protects the user from the correlation attack, as the IID is generated using parameters that are not constant. Each time a device changes its location, the IID will be recalculated [13]. According to Common Weakness Enumeration (CWE) [26], the use of a small seed *"few parameters"* causes weakness during the implementation of an architectural security due to predictability. Security is determined entirely by the total length of the parameters [27]. Including Network_ID will make the total length of our parameters longer than the total length of RFC7217. As the number of parameters increases, the time required to analyze them will increase. Hence, our method will achieve higher security leading to increased privacy. However, different resources, such as time and space (parameters), should not be compared directly, therefore comparing efficiency of different algorithms depends on which efficiency measure is more important [28]. A 256-bit seed is a good starting point for producing a random enough number. For an adversary to correlate this newly generated IID, he/she will need to have access to all the victim's parameters to be able to compute the hash, which makes it difficult to succeed.

IV. Experimental Analysis

In this experiment, we describe the parameters used, the test environment and a detailed implementation of the approach. In order to perform the experiment with the new IID generation method we consider the following example value for each of the parameters.

Network Prefix: **fe80::**, Network Interface: **wlan0**, Network_ID: **iPhone**, DAD Counter: **100**, Secret Key: ***84a0 d5aa 52b0 4d35 k567 3aa6 7af5 474c***

The secret key is a 128-bit hexadecimal string that looks like an IPv6.

To produce an IID, we use these parameters in a SHA-256 function to produce a random hexadecimal.

f{*fe80:0000:0000:0000, wlan0, iPhone, 100, 84a0 d5aa 52b0 4d35 k567 3aa6 7af5 474c*}

The output of the function is:
531eb075a9b54885ea5ab75ee306fd85a17518325ab9a84568e f2692f19e73fa

Our IID is generated by taking the last 64 bits. Hence, our IID will be *08ef:2692:f19e:73fa/64*

A. Configuring the test environment

Our test environment is consisting of a Debian Linux machine running on a Linux 4.6 kernel. In this experiment, we use IPv6 tokens to test if the newly generated IID will diminish the user privacy. First, we must verify that the interface we want to configure with a token is plumbed using *#ifconfig -a*

The output should show a link local address that was automatically configured during installation. To configure an interface with a token, we use the following command:

#ip token set ::*8ef:2692:f19e:73fa/64 dev <interface>*

And repeat this in all the interfaces that will use the token.

B. Experimental Analysis And Results

The default method for generating node identifiers in IPv6 is modified EUI-64. An experiment has been conducted showing how this method reveals MAC address of the communicating nodes and how we can solve it using randomized IPv6 tokens. The use of MAC addresses as node identifiers has been discouraged by both RFC7217 and RFC8064 [13] [14], which shows how the device MAC address is used to generate the IIDs and how an adversary can take advantage of that by doing a reverse attack to determine the MAC address of the communicating device and to correlate its activities online.

Figure 5 is a ping test to show that the new IID is working correctly. This ping result shows that our new IID has no trace of MAC address.

In order to determine whether our address is working correctly, a loopback test is conducted where the communicating node sends a signal using the new IPv6 address and then returned (looped back) to it. The aim of this test is to determine whether our node will fail if it uses the new IPv6 address. From the results shown in Figure 5, the address worked successfully.

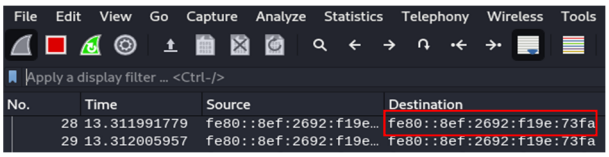

Figure 5. IPv6 ping results

Figure 6 is a Wireshark packet capture of how our new IID will appear in any traffic analysis. Comparing Figure 6 with Figure 2, it is evident that an adversary cannot conduct a reverse attack to determine the MAC address of the communicating node because the 64-bit IID is random.

Figure 6. Privacy enhanced IPv6 IID

C. Performance Analysis

In our case we used AMD-64 Debian Linux. Our experiment used a SHA256 to generate the RID. SHA512 hashing will perform much faster when compared to SHA256 as shown in Figure 7 and Figure 8. However, this comes at a higher cost. From a security point of view, it would be pointless to generate random hashes using SHA512 because in practical terms, SHA256 is just as secure as SHA384 and SHA512 [29]. A collision cannot occur in any of them with the current foreseeable technology. Hence, our algorithm will be much cheaper to produce and secure.

When calculating processing power, we need to consider electricity cost. This is called efficiency of the processor. An increase in the difficulty of processing a hash increases the electricity cost. SHA512 makes 25% more rounds than SHA256 on a 64-bit processor. SHA512 performs 50% much faster than SHA256 for typical data sizes. However, for small messages (less than 448 bits) SHA512 will be approximately 1.25 times slower than SHA256 [29].

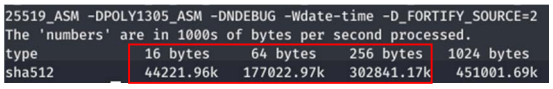

Figure 7. SHA512 hashing speed in amd-64 Linux

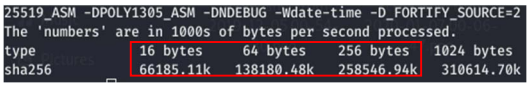

Figure 8 SHA256 hashing speed in amd-64 Linux

D. DISCUSSION

Privacy is one of the major challenges of IPv6 implementation. The potentiality of attacks that arise from using MAC address in RFC4291 and RFC4862 as part of the address generation prompted researchers to find ways of addressing the challenge. As discussed earlier, temporary address generation proposed by RFC4941 limits the capability of network management while stable IID generation proposed in RFC7217 enhances privacy but does not include all the potential parameters for generating IID to enhance privacy. Our method is a modification to RFC7217 to include Network_ID field which is optional in RFC7217. Based on the results of the experiment, our method produces IIDs that are non-deterministic. Additionally, it does not reveal user's MAC address. A comparison between different IID generation proposals is provided in Table 1.

Table 1. Features of various IID generation methods

Proposal	Usage of MAC	Correlation of device	Privacy
RFC4291	✓	For device lifetime	Low
RFC4862	✓	For device lifetime	Low
RFC4941	×	For temporary address lifetime	High
RFC7217	×	Only within a single IPv6 link	High
Our proposal	×	Only within a single IPv6 link	Higher

V. CONCLUSION AND FUTURE WORK

IPv6 has revolutionized Internet addressing by allowing more devices to connect to the internet. However, its privacy has come under attack by generating deterministic IIDs. This ability to predict a user's IPv6 address on any subnet has exposed the users to correlation attack. The ease of using deterministic IID does not pay back for the privacy that the user is forced to surrender. Solutions such as CGA and temporary IID have fallen short in addressing the cost of implementation and network management, respectively. A new solution that generates IIDs by hashing network prefix, network interface,

Network_ID, DAD counter and secret key has been proposed. This method produces nondeterministic 64-bit IID which does not contain the device MAC address. Hence, this method should be implemented on operating systems to protect the privacy of the users.

One possible future work could be identifying the best way to assign our new IID to nodes. Currently, we assigned the IID using IPv6 token which tends to be static. Static addresses are only recommended for devices that need addresses that do not change, for example, network servers. Hosts on the other hand need addresses that are dynamic to enhance security.

REFERENCES

[1] S. Hagen, "IPv6 Essentials. Integrating IPv6 into your IPv4 Network,," *O'Reily,* pp. 4-10, 2006.

[2] I. V. Beijnum, "Running IPv6: A practical guide to configuring IPv6 for windows XP, MacOS, FreeBSD, RedHat Linux, Cisco routers, DNS and BIND, Zebra and Apache2,," *Apress,* pp. 34-40, 2006.

[3] R. Beverly, D. Plonka, R. Durairajan and J. P. Rohrer, "In the IP of the beholder: Stages for active IPv6 Topology Discovery," in *Proceedings of the Internet Measurement Conference 2018*, Boston MA, 2018.

[4] S. Groat, M. Dunlop, R. Marchany and J. Tront, "The privacy implications of stateless IPv6 addressing," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, Blacksburg, 2010.

[5] T. Narten, R. Draves and S. Krishnan, "Privacy extension for stateless address autoconfiguration," *Network Working group,* vol. RFC4941, 2007.

[6] S. Thomas, T. Narten and T. Jinmei, "IPv6 Stateless Autoconfiguration," *Network Working Group,* vol. RFC4862, 2007.

[7] B. Carpenter and S. Jiang, "Significance of IPv6 Interface Identifiers," *RFC7136,* February 2014.

[8] F. Gont, "ask Ubuntu," Ubuntu, 7 Feb 2017. [Online]. Available: https://askubuntu.com/questions/880666/rfc7217-stable-privacy-addresses-implementation-in-ubuntu-16-10. [Accessed 28 May 2020].

[9] R. Koodli, "IP address location privacy and mobile IPv6," *Network Working Group,* vol. RFC4882, March 2007.

[10] N. Jagatheesan and S. El-Kadri, "Privacy and Security in IPv6," *ArXiv,* vol. abc, 2013.

[11] A. Cooper, F. Gont and D. Thaler, "Privacy considerations for IPv6 address generation mechanisms," *Network Working Goup,* no. Internet Draft, 15 July 2013.

[12] J. Weippl and E. Ullrich, "Privacy is not an option: Attacking the IPv6 privacy extension," in *International Symposium on Recent Advances in Intrusion Detection*, Vienna, 2015.

[13] F. Gont, "A method for generating semanticaly opaque interface identifiers with IPv6 stateless address autoconfiguration (SLAAC)," *IETF,* vol. RFC7217, April 2014.

[14] F. Gont, A. Cooper, D. Thaler and W. Liu, "Privacy extension for stateless address autoconfiguration in IPv6," *IETF,* vol. RFC8064, February 2017.

[15] D. Rachmawati, J. T. Targan and A. Ginting, "A comparative study of message digest 5 (MD5) and SHA256 algorithm," in *2nd International Conference on Computing and Applied Informatics*, Medan, Indonesia, 2018.

[16] M. Foster and J. C. Burnett, Hacking the code. ASP.NET web application security, Syngress, 2004.

[17] E. Kirdan, "Internet protocol version 6," Lehrstuhl fur Netzarchitekturen und Netzdienste Fakutat fur informatik, Munich, 2019.

[18] R. Deering and S. Hinden, "IP version 6 addressing architecture," *Network Working Group,* vol. RFC4291, 2006.

[19] S. Krishnan, "Reserved IPv6 Interface Identifiers," *IETF,* vol. RFC5453, February 2009.

[20] S. Krishnan, T. Narten, R. Draves and F. Gont, "privacy extension for stateless autoconfiguration in IPv6," *6 man Working Group,* no. Internet-Draft, 5 March 2018.

[21] S. Daley and G. Krishnan, "Simple procedures for detecting network attachment in IPv6," *Network Working Group,* vol. RFC6059, November 2010.

[22] D. Eastlakes, J. Schiller and J. Crockers, "Randomness requirement for security," *Network Working Group,* vol. RFC4086, June 2005.

[23] E. C. Rye, J. Martin and R. Beverly, "EUI-64 Considered Harmful," *Networking and Internet Architecture,* 24 Feb 2019.

[24] K. Dalya , O. K. Sheet, H. A. Abdul and N. H. Ahmed, "Location Information verification cum security using TBM in geocast routing," *ScienceDirect,* 2015.

[25] K. Sunil , S. Karan, K. Omprakash, C. Yue and Z. Huan, "Delimitated Antijammer scheme for internet of vehicle: Machine learning based security approach," vol. 7, 2019.

[26] CWE, "Generation of Predictable Numbers or Identifiers," MITRE, February 2020. [Online]. Available: https://cwe.mitre.org/data/definitions/340.html. [Accessed 28 May 2020].

[27] A. K. Lenstra, "Key Length. Contribution to The Handbook of Information Security," *Infoscience,* 2004.

[28] D. W. Penny, J. K. Friston, T. J. Ashburner, J. S. Kiebel and E. T. Nichols, time, Elsevier, 2011.

[29] H. Hanschuh and H. Gilbert, "Security analysis of SHA and sisters," in *International workshop on selected areas in cryptography*, 2003.