

IPv6 maintenance Working Group (6man)  
Internet-Draft  
Intended status: Standards Track  
Expires: March 7, 2014

F. Gont  
SI6 Networks / UTN-FRH  
September 3, 2013

**A Method for Generating Semantically Opaque Interface Identifiers with  
IPv6 Stateless Address Autoconfiguration (SLAAC)  
draft-ietf-6man-stable-privacy-addresses-13**

**Abstract**

This document specifies a method for generating IPv6 Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), such that addresses configured using this method are stable within each subnet, but the Interface Identifier changes when hosts move from one network to another. This method is meant to be an alternative to generating Interface Identifiers based on hardware address (e.g., using IEEE identifiers), such that the benefits of stable addresses can be achieved without sacrificing the privacy of users. The method specified in this document applies to all prefixes a host may be employing, including link-local, global, and unique-local addresses.

**Status of this Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 7, 2014.

**Copyright Notice**

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Design goals . . . . .</a>	<a href="#">7</a>
<a href="#">3.</a>	<a href="#">Algorithm specification . . . . .</a>	<a href="#">9</a>
<a href="#">4.</a>	<a href="#">Resolving Duplicate Address Detection (DAD) conflicts . . . .</a>	<a href="#">14</a>
<a href="#">5.</a>	<a href="#">Specified Constants . . . . .</a>	<a href="#">15</a>
<a href="#">6.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">16</a>
<a href="#">7.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">17</a>
<a href="#">8.</a>	<a href="#">Acknowledgements . . . . .</a>	<a href="#">19</a>
<a href="#">9.</a>	<a href="#">References . . . . .</a>	<a href="#">20</a>
<a href="#">9.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">20</a>
<a href="#">9.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">21</a>
<a href="#">Appendix A.</a>	<a href="#">Possible sources for the Net_Iface parameter . . . . .</a>	<a href="#">23</a>
<a href="#">A.1.</a>	<a href="#">Interface Index . . . . .</a>	<a href="#">23</a>
<a href="#">A.2.</a>	<a href="#">Interface Name . . . . .</a>	<a href="#">23</a>
<a href="#">A.3.</a>	<a href="#">Link-layer Addresses . . . . .</a>	<a href="#">23</a>
<a href="#">A.4.</a>	<a href="#">Logical Network Service Identity . . . . .</a>	<a href="#">24</a>
	<a href="#">Author's Address . . . . .</a>	<a href="#">25</a>



## 1. Introduction

[RFC4862] specifies Stateless Address Autoconfiguration (SLAAC) for IPv6 [RFC2460], which typically results in hosts configuring one or more "stable" addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) that typically embeds a hardware address (e.g., using IEEE identifiers) [RFC4291].

Cryptographically Generated Addresses (CGAs) [RFC3972] are yet another method for generating Interface Identifiers, which bind a public signature key to an IPv6 address in the SEcure Neighbor Discovery (SEND) [RFC3971] protocol.

Generally, the traditional SLAAC addresses are thought to simplify network management, since they simplify Access Control Lists (ACLs) and logging. However, they have a number of drawbacks:

- o since the resulting Interface Identifiers do not vary over time, they allow correlation of node activities within the same network, thus negatively affecting the privacy of users.
- o since the resulting Interface Identifiers are constant across networks, the resulting IPv6 addresses can be leveraged to track and correlate the activity of a node across multiple networks (e.g. track and correlate the activities of a typical client connecting to the public Internet from different locations), thus negatively affecting the privacy of users.
- o since embedding the underlying link-layer address in the Interface Identifier will result in specific address patterns, such patterns may be leveraged by attackers to reduce the search space when performing address scanning attacks. For example, the IPv6 addresses of all nodes manufactured by the same vendor (at a given time frame) will likely contain the same IEEE Organizationally Unique Identifier (OUI) in the Interface Identifier.
- o embedding the underlying link-layer address in the Interface Identifier leaks device-specific information that could be leveraged to launch device-specific attacks.
- o embedding the underlying link-layer address in the Interface Identifier means that replacement of the underlying interface hardware will result in a change of the IPv6 address(es) assigned to that interface.

[I-D.cooper-6man-ipv6-address-generation-privacy] provides additional details regarding how these vulnerabilities could be exploited, and the extent to which the method discussed in this document mitigates



them.

The "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" [[RFC4941](#)] (henceforth referred to as "temporary addresses") were introduced to complicate the task of eavesdroppers and other information collectors (e.g. IPv6 addresses in web server logs or email headers, etc.) to correlate the activities of a node, and basically result in temporary (and random) Interface Identifiers. These temporary addresses are generated in addition to the traditional IPv6 addresses based on IEEE identifiers, with the "temporary addresses" being employed for "outgoing communications", and the traditional SLAAC addresses being employed for "server" functions (i.e., receiving incoming connections).

It should be noted that temporary addresses can be challenging in a number of areas. For example, from a network-management point of view, they tend to increase the complexity of event logging, trouble-shooting, enforcement of access controls and quality of service, etc. As a result, some organizations disable the use of temporary addresses even at the expense of reduced privacy [[Broersma](#)]. Temporary addresses may also result in increased implementation complexity, which might not be possible or desirable in some implementations (e.g., some embedded devices).

In scenarios in which temporary addresses are deliberately not used (possibly for any of the aforementioned reasons), all a host is left with is the stable addresses that have been generated using e.g. IEEE identifiers. In such scenarios, it may still be desirable to have addresses that mitigate address scanning attacks, and that at the very least do not reveal the node's identity when roaming from one network to another -- without complicating the operation of the corresponding networks.

However, even with "temporary addresses" in place, a number of issues remain to be mitigated. Namely,

- o since "temporary addresses" [[RFC4941](#)] do not eliminate the use of fixed identifiers for server-like functions, they only partially mitigate host-tracking and activity correlation across networks (see [[I-D.cooper-6man-ipv6-address-generation-privacy](#)] for some example attacks that are still possible with temporary addresses).
- o since "temporary addresses" [[RFC4941](#)] do not replace the traditional SLAAC addresses, an attacker can still leverage patterns in those addresses to greatly reduce the search space for "alive" nodes [[Gont-DEEPSEC2011](#)] [[CPNI-IPv6](#)] [[I-D.ietf-opsec-ipv6-host-scanning](#)].



Hence, there is a motivation to improve the properties of "stable" addresses regardless of whether temporary addresses are employed or not.

We note that attackers can employ a plethora of probing techniques [[I-D.ietf-opsec-ipv6-host-scanning](#)] to exploit the aforementioned issues. Some of them (such as the use of ICMPv6 Echo Request and ICMPv6 Echo Response packets) could be mitigated by a personal firewall at the target host. For other vectors, such as listening to ICMPv6 "Destination Unreachable, Address Unreachable" (Type 1, Code 3) error messages referring to the target addresses [[I-D.ietf-opsec-ipv6-host-scanning](#)], there is nothing a host can do (e.g., a personal firewall at the target host would not be able to mitigate this probing technique).

This document specifies a method to generate Interface Identifiers that are stable/constant for each network interface within each subnet, but that change as hosts move from one network to another, thus keeping the "stability" properties of the Interface Identifiers specified in [[RFC4291](#)], while still mitigating address-scanning attacks and preventing correlation of the activities of a node as it moves from one network to another.

The method specified in this document is orthogonal to the use of "temporary" addresses [[RFC4941](#)], since it is meant to improve the security and privacy properties of the stable addresses that are employed along with the aforementioned "temporary" addresses. In scenarios in which "temporary addresses" are employed, implementation of the mechanism described in this document (in replacement of stable addresses based on e.g. IEEE identifiers) would mitigate address-scanning attacks and also mitigate the remaining vectors for correlating host activities based on the node's constant (i.e. stable across networks) Interface Identifiers. On the other hand, for nodes that currently disable "temporary addresses" [[RFC4941](#)] for some of the reasons described earlier in this document, implementation of this mechanism will result in stable privacy-enhanced addresses which address some of the concerns related to addresses that embed IEEE identifiers [[RFC4291](#)], and which mitigate IPv6 address-scanning attacks.

We note that this method is incrementally deployable, since it does not pose any interoperability implications when deployed on networks where other nodes do not implement or employ it. Additionally, we note that this document does not update or modify IPv6 Stateless Address Auto-Configuration (SLAAC) [[RFC4862](#)] itself, but rather only specifies an alternative algorithm to generate Interface Identifiers. Therefore, the usual address lifetime properties (as specified in the corresponding Prefix Information Options) apply when IPv6 addresses



Gont

Expires March 7, 2014

[Page 5]

are generated as a result of employing the algorithm specified in this document with SLAAC [[RFC4862](#)]. Additionally, from the point of view of renumbering, we note that these addresses behave like the traditional IPv6 addresses (that embed a hardware address) resulting from SLAAC [[RFC4862](#)].

While the method specified in this document is meant to be used with SLAAC, this does not preclude the same algorithm from being used with other address configuration mechanisms, such as DHCPv6 [[RFC3315](#)] or manual address configuration.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].



## 2. Design goals

This document specifies a method for selecting Interface Identifiers to be used with IPv6 SLAAC, with the following goals:

- o The resulting Interface Identifiers remain constant/stable for each prefix used with SLAAC within each subnet. That is, the algorithm generates the same Interface Identifier when configuring an address (for the same interface) belonging to the same prefix within the same subnet.
- o The resulting Interface Identifiers do change when addresses are configured for different prefixes. That is, if different autoconfiguration prefixes are used to configure addresses for the same network interface card, the resulting Interface Identifiers must be (statistically) different. This means that, given two addresses produced by the method specified in this document, it must be difficult for an attacker tell whether the addresses have been generated/used by the same node.
- o It must be difficult for an outsider to predict the Interface Identifiers that will be generated by the algorithm, even with knowledge of the Interface Identifiers generated for configuring other addresses.
- o Depending on the specific implementation approach (see [Section 3](#) and [Appendix A](#)), the resulting Interface Identifiers may be independent of the underlying hardware (e.g. link-layer address). This means that e.g. replacing a Network Interface Card (NIC) will not have the (generally undesirable) effect of changing the IPv6 addresses used for that network interface.
- o The method specified in this document is meant to be an alternative to producing IPv6 addresses based on e.g. IEEE identifiers (as specified in [[RFC2464](#)]). It is meant to be employed for all of the stable (i.e. non-temporary) IPv6 addresses configured with SLAAC for a given interface, including global, link-local, and unique-local IPv6 addresses.

We note that of use of the algorithm specified in this document is (to a large extent) orthogonal to the use of "temporary addresses" [[RFC4941](#)]. When employed along with "temporary addresses", the method specified in this document will mitigate address-scanning attacks and correlation of node activities across networks (see [[I-D.cooper-6man-ipv6-address-generation-privacy](#)] and [[IAB-PRIVACY](#)]). On the other hand, hosts that do not implement/use "temporary addresses" but employ the method specified in this document would, at the very least, mitigate the host-tracking and address scanning



issues discussed in the previous section.

### 3. Algorithm specification

IPv6 implementations conforming to this specification MUST generate Interface Identifiers using the algorithm specified in this section in replacement of any other algorithms used for generating "stable" addresses with SLAAC (such as those specified in [\[RFC2464\]](#)). However, implementations conforming to this specification MAY employ the algorithm specified in [\[RFC4941\]](#) to generate temporary addresses in addition to the addresses generated with the algorithm specified in this document. The method specified in this document MUST be employed for generating the Interface Identifiers with SLAAC for all the stable addresses of a given interface, including IPv6 global, link-local, and unique-local addresses.

This means that this document does not formally obsolete or deprecate any of the existing algorithms to generate Interface Identifiers (e.g. such as that specified in [\[RFC2464\]](#)). However, those IPv6 implementations that employ this specification MUST generate all of their "stable" addresses as specified in this document.

Implementations conforming to this specification SHOULD provide the means for a system administrator to enable or disable the use of this algorithm for generating Interface Identifiers.

Unless otherwise noted, all of the parameters included in the expression below MUST be included when generating an Interface Identifier.

1. Compute a random (but stable) identifier with the expression:

$$\text{RID} = \text{F}(\text{Prefix}, \text{Net\_Iface}, \text{Network\_ID}, \text{DAD\_Counter}, \text{secret\_key})$$

Where:

RID:

Random (but stable) Interface Identifier

F():

A pseudorandom function (PRF) that is not computable from the outside (without knowledge of the secret key), which should produce an output of at least 64 bits. The PRF could be implemented as a cryptographic hash of the concatenation of each of the function parameters.





**Prefix:**

The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message, or the link-local IPv6 unicast prefix.

**Net\_Iface:**

An implementation-dependent stable identifier associated with the network interface for which the RID is being generated. An implementation MAY provide a configuration option to select the source of the identifier to be used for the Net\_Iface parameter. A discussion of possible sources for this value (along with the corresponding trade-offs) can be found in [Appendix A](#).

**Network\_ID:**

Some network specific data that identifies the subnet to which this interface is attached. For example the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. This parameter is OPTIONAL.

**DAD\_Counter:**

A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts. It MUST be initialized to 0, and incremented by 1 for each new tentative address that is configured as a result of a DAD conflict. Implementations that record DAD\_Counter in non-volatile memory for each {Prefix, Net\_Iface, Network\_ID} tuple MUST initialize DAD\_Counter to the recorded value if such an entry exists in non-volatile memory). See [Section 4](#) for additional details.

**secret\_key:**

A secret key that is not known by the attacker. The secret key MUST be initialized at operating system installation time to a pseudo-random number (see [[RFC4086](#)] for randomness requirements for security). An implementation MAY provide the means for the the system administrator to change or display the secret key.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit.

We note that [[RFC4291](#)] requires that, the Interface IDs of all unicast addresses (except those that start with the binary value 000) be 64-bit long. However, the method discussed in this document could be employed for generating Interface IDs of any arbitrary length, albeit at the expense of reduced

Gont

Expires March 7, 2014

[Page 10]

entropy (when employing Interface IDs smaller than 64 bits).

The resulting Interface Identifier should be compared against the Subnet-Router Anycast [[RFC4291](#)] and the Reserved Subnet Anycast Addresses [[RFC2526](#)], and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, this situation should be handled in the same way as the case of duplicate addresses (see [Section 4](#)).

This document does not require the use of any specific PRF for the function  $F()$  above, since the choice of such PRF is usually a trade-off between a number of properties (processing requirements, ease of implementation, possible intellectual property rights, etc.), and since the best possible choice for  $F()$  might be different for different types of devices (e.g. embedded systems vs. regular servers) and might possibly change over time.

Note that the result of  $F()$  in the algorithm above is no more secure than the secret key. If an attacker is aware of the PRF that is being used by the victim (which we should expect), and the attacker can obtain enough material (i.e. addresses configured by the victim), the attacker may simply search the entire secret-key space to find matches. To protect against this, the secret key should be of a reasonable length. Key lengths of at least 128 bits should be adequate. The secret key is initialized at system installation time to a pseudo-random number, thus allowing this mechanism to be enabled/used automatically, without user intervention.

Including the SLAAC prefix in the PRF computation causes the Interface Identifier to vary across each prefix (link-local, global, etc.) employed by the node and, as consequently, also across networks. This mitigates the correlation of activities of multi-homed nodes (since each of the corresponding addresses will employ a different Interface ID), host-tracking (since the network prefix will change as the node moves from one network to another), and any other attacks that benefit from predictable Interface Identifiers (such as address scanning attacks).

The `Net_Iface` is a value that identifies the network interface for which an IPv6 address is being generated. The following properties are required for the `Net_Iface` parameter:

- o it MUST be constant across system bootstrap sequences and other network events (e.g., bringing another interface up or down)



- o it MUST be different for each network interface simultaneously in use

Since the stability of the addresses generated with this method relies on the stability of all arguments of `F()`, it is key that the `Net_Iface` be constant across system bootstrap sequences and other network events. Additionally, the `Net_Iface` must uniquely identify an interface within the node, such that two interfaces connecting to the same network do not result in duplicate addresses. Different types of operating systems might benefit from different stability properties of the `Net_Iface` parameter. For example, a client-oriented operating system might want to employ `Net_Iface` identifiers that are attached to the underlying network interface card (NIC), such that a removable NIC always gets the same IPv6 address, irrespective of the system communications port to which it is attached. On the other hand, a server-oriented operating system might prefer `Net_Iface` identifiers that are attached to system slots/ports, such that replacement of a network interface card does not result in an IPv6 address change. [Appendix A](#) discusses possible sources for the `Net_Iface`, along with their pros and cons.

Including the optional `Network_ID` parameter when computing the RID value above would cause the algorithm to produce a different Interface Identifier when connecting to different networks, even when configuring addresses belonging to the same prefix. This means that a host would employ a different Interface Identifier as it moves from one network to another even for IPv6 link-local addresses or Unique Local Addresses (ULAs). In those scenarios where the `Network_ID` is unknown to the attacker, including this parameter might help mitigate attacks where a victim node connects to the same subnet as the attacker, and the attacker tries to learn the Interface Identifier used by the victim node for a remote network (see [Section 7](#) for further details).

The `DAD_Counter` parameter provides the means to intentionally cause this algorithm produce a different IPv6 addresses (all other parameters being the same). This could be necessary to resolve Duplicate Address Detection (DAD) conflicts, as discussed in detail in [Section 4](#).

Finally, we note that all of the bits in the resulting Interface IDs are treated as "opaque" bits. For example, the universal/local bit of Modified EUI-64 format identifiers is treated as any other bit of such identifier. In theory, this might result in Duplicate Address Detection (DAD) failures that would otherwise not be encountered. However, this is not deemed as a real issue, because of the following considerations:

Gont

Expires March 7, 2014

[Page 12]

- o The interface IDs of all addresses (except those of addresses that start with the binary value 000) are 64-bit long. Since the method specified in this document results in random Interface IDs, the probability of DAD failures is very small.
- o Real world data indicates that MAC address reuse is far more common than assumed [[HDMoore](#)]. This means that even IPv6 addresses that employ (allegedly) unique identifiers (such as IEEE identifiers) might result in DAD failures, and hence implementations should be prepared to gracefully handle such occurrences.
- o Since some popular and widely-deployed operating systems (such as Microsoft Windows) do not employ unique hardware identifiers for the Interface IDs of their stable addresses, reliance on such unique identifiers is more reduced in the deployed world (fewer deployed systems rely on them for the avoidance of address collisions).





#### **4. Resolving Duplicate Address Detection (DAD) conflicts**

If as a result of performing Duplicate Address Detection (DAD) [[RFC4862](#)] a host finds that the tentative address generated with the algorithm specified in [Section 3](#) is a duplicate address, it SHOULD resolve the address conflict by trying a new tentative address as follows:

- o DAD\_Counter is incremented by 1.
- o A new Interface Identifier is generated with the algorithm specified in [Section 3](#), using the incremented DAD\_Counter value.

This procedure may be repeated a number of times until the address conflict is resolved. Hosts SHOULD try at least IDGEN\_RETRIES (see [Section 5](#)) tentative addresses if DAD fails for successive generated addresses, in the hopes of resolving the address conflict. We also note that hosts MUST limit the number of tentative addresses that are tried (rather than indefinitely try a new tentative address until the conflict is resolved).

In those (unlikely) scenarios in which duplicate addresses are detected and in which the order in which the conflicting nodes configure their addresses may vary (e.g., because they may be bootstrapped in different order), the algorithm specified in this section for resolving DAD conflicts could lead to addresses that are not stable within the same subnet. In order to mitigate this potential problem, nodes MAY record the DAD\_Counter value employed for a specific {Prefix, Net\_Iface, Network\_ID} tuple in non-volatile memory, such that the same DAD\_Counter value is employed when configuring an address for the same Prefix and subnet at any other point in time.

In the event that a DAD conflict cannot be solved (possibly after trying a number of different addresses), address configuration would fail. In those scenarios, nodes MUST NOT automatically fall back to employing other algorithms for generating Interface Identifiers.



## **5. Specified Constants**

This document specifies the following constant:

IDGEN\_RETRIES:  
    defaults to 3.

## **6. IANA Considerations**

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

## 7. Security Considerations

This document specifies an algorithm for generating Interface Identifiers to be used with IPv6 Stateless Address Autoconfiguration (SLAAC), as an alternative to e.g. Interface Identifiers that embed IEEE identifiers (such as those specified in [[RFC2464](#)]). When compared to such identifiers, the identifiers specified in this document have a number of advantages:

- o They prevent trivial host-tracking, since when a host moves from one network to another the network prefix used for autoconfiguration and/or the Network ID (e.g., IEEE 802.11 SSID) will typically change, and hence the resulting Interface Identifier will also change (see [[I-D.cooper-6man-ipv6-address-generation-privacy](#)]).
- o They mitigate address-scanning techniques which leverage predictable Interface Identifiers (e.g., known Organizationally Unique Identifiers) [[I-D.ietf-opsec-ipv6-host-scanning](#)].
- o They may result in IPv6 addresses that are independent of the underlying hardware (i.e. the resulting IPv6 addresses do not change if a network interface card is replaced) if an appropriate source for Net\_Iface ([Section 3](#)) is employed.
- o They prevent the information leakage produced by embedding hardware addresses in the Interface Identifier (which could be exploited to launch device-specific attacks).
- o Since the method specified in this document will result in different Interface Identifiers for each configured address, knowledge/leakage of the Interface Identifier employed for one stable address will not negatively affect the security/privacy of other stable addresses configured for other prefixes (whether at the same time or at some other point in time).

In scenarios in which an attacker can connect to the same subnet as a victim node, the attacker might be able to learn the Interface Identifier employed by the victim node for an arbitrary prefix, by simply sending a forged Router Advertisement [[RFC4861](#)] for that prefix, and subsequently learning the corresponding address configured by the victim node (either listening to the Duplicate Address Detection packets, or to any other traffic that employs the newly configured address). We note that a number of factors might limit the ability of an attacker to successfully perform such an attack:



- o First-Hop security mechanisms such as RA-Guard [[RFC6105](#)] [[I-D.ietf-v6ops-ra-guard-implementation](#)] could prevent the forged Router Advertisement from reaching the victim node
- o If the victim implementation includes the (optional) Network\_ID parameter for computing F() (see [Section 3](#)), and the Network\_ID employed by the victim for a remote network is unknown to the attacker, the Interface Identifier learned by the attacker would differ from the one used by the victim when connecting to the legitimate network.

In any case, we note that at the point in which this kind of attack becomes a concern, a host should consider employing Secure Neighbor Discovery (SEND) [[RFC3971](#)] to prevent an attacker from illegitimately claiming authority for a network prefix.

We note that this algorithm is meant to be an alternative to Interface Identifiers such as those specified in [[RFC2464](#)], but is not meant as an alternative to temporary Interface Identifiers (such as those specified in [[RFC4941](#)]). Clearly, temporary addresses may help to mitigate the correlation of activities of a node within the same network, and may also reduce the attack exposure window (since temporary addresses are short-lived when compared to the addresses generated with the method specified in this document). We note that implementation of this algorithm would still benefit those hosts employing "temporary addresses", since it would mitigate host-tracking vectors still present when such addresses are used (see [[I-D.cooper-6man-ipv6-address-generation-privacy](#)]), and would also mitigate address-scanning techniques that leverage patterns in IPv6 addresses that embed IEEE identifiers.

Finally, we note that the method described in this document addresses some of the privacy concerns arising from the use of IPv6 addresses that embed IEEE identifiers, without the use of temporary addresses, thus possibly offering an interesting trade-off for those scenarios in which the use of temporary addresses is not feasible.





## **8. Acknowledgements**

The algorithm specified in this document has been inspired by Steven Bellovin's work ([\[RFC1948\]](#)) in the area of TCP sequence numbers.

The author would like to thank (in alphabetical order) Mikael Abrahamsson, Ran Atkinson, Karl Auer, Steven Bellovin, Matthias Bethke, Ben Campbell, Brian Carpenter, Tassos Chatzithomaoglou, Tim Chown, Alissa Cooper, Dominik Elsbroek, Brian Haberman, Bob Hinden, Christian Huitema, Ray Hunter, Jouni Korhonen, Eliot Lear, Jong-Hyouk Lee, Andrew McGregor, Tom Petch, Michael Richardson, Mark Smith, Dave Thaler, Ole Troan, James Woodyatt, and He Xuan, for providing valuable comments on earlier versions of this document.

This document is based on the technical report "Security Assessment of the Internet Protocol version 6 (IPv6)" [\[CPNI-IPv6\]](#) authored by Fernando Gont on behalf of the UK Centre for the Protection of National Infrastructure (CPNI).

Fernando Gont would like to thank CPNI (<http://www.cpni.gov.uk>) for their continued support.



## **9. References**

### **9.1. Normative References**

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), December 1998.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", [RFC 2526](#), March 1999.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", [RFC 3315](#), July 2003.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", [RFC 3493](#), February 2003.
- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", [RFC 3542](#), May 2003.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", [RFC 3971](#), March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", [RFC 3972](#), March 2005.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), June 2005.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", [RFC 4122](#), July 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", [RFC 4291](#), February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", [RFC 4861](#), September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", [RFC 4862](#), September 2007.



- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", [RFC 4941](#), September 2007.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", [RFC 6105](#), February 2011.

## 9.2. Informative References

- [RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", [RFC 1948](#), May 1996.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", [RFC 2464](#), December 1998.
- [I-D.ietf-opsec-ipv6-host-scanning]  
Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", [draft-ietf-opsec-ipv6-host-scanning-02](#) (work in progress), July 2013.
- [I-D.ietf-v6ops-ra-guard-implementation]  
Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", [draft-ietf-v6ops-ra-guard-implementation-07](#) (work in progress), November 2012.
- [I-D.cooper-6man-ipv6-address-generation-privacy]  
Cooper, A., Gont, F., and D. Thaler, "Privacy Considerations for IPv6 Address Generation Mechanisms", [draft-cooper-6man-ipv6-address-generation-privacy-00](#) (work in progress), July 2013.
- [HDMoore] HD Moore, "The Wild West", Louisville, Kentucky, U.S.A. September 25-29, 2012, <<https://speakerdeck.com/hdm/derbycon-2012-the-wild-west>>.
- [Gont-DEEPSEC2011]  
Gont, "Results of a Security Assessment of the Internet Protocol version 6 (IPv6)", DEEPSEC 2011 Conference, Vienna, Austria, November 2011, <<http://www.si6networks.com/presentations/deepsec2011/fgont-deepsec2011-ipv6-security.pdf>>.
- [Gont-BRUCON2012]  
Gont, "Recent Advances in IPv6 Security", BRUCON 2012 Conference, Ghent, Belgium, September 2012, <<http://www.si6networks.com/presentations/brucon2012/>>.



fgont-brucon2012-recent-advances-in-ipv6-security.pdf>.

[Broersma]

Broersma, R., "IPv6 Everywhere: Living with a Fully IPv6-enabled environment", Australian IPv6 Summit 2010, Melbourne, VIC Australia, October 2010, <[http://www.ipv6.org.au/10ipv6summit/talks/Ron\\_Broersma.pdf](http://www.ipv6.org.au/10ipv6summit/talks/Ron_Broersma.pdf)>.

[IAB-PRIVACY]

IAB, "Privacy and IPv6 Addresses", July 2011, <<http://www.iab.org/wp-content/IAB-uploads/2011/07/IPv6-addresses-privacy-review.txt>>.

[CPNI-IPv6]

Gont, F., "Security Assessment of the Internet Protocol version 6 (IPv6)", UK Centre for the Protection of National Infrastructure, (available on request).





## **[Appendix A](#). Possible sources for the Net\_Iface parameter**

The following subsections describe a number of possible sources for the Net\_Iface parameter employed by the F() function in [Section 3](#). The choice of a specific source for this value represents a number of trade-offs, which may vary from one implementation to another.

### **[A.1](#). Interface Index**

The Interface Index [[RFC3493](#)] [[RFC3542](#)] of an interface uniquely identifies an interface within a node. However, these identifiers might or might not have the stability properties required for the Net\_Iface value employed by this method. For example, the Interface Index might change upon removal or installation of a network interface (typically one with a smaller value for the Interface Index, when such a naming scheme is used), or when network interfaces happen to be initialized in a different order. We note that some implementations are known to provide configuration knobs to set the Interface Index for a given interface. Such configuration knobs could be employed to prevent the Interface Index from changing (e.g. as a result of the removal of a network interface).

### **[A.2](#). Interface Name**

The Interface Name (e.g., "eth0", "em0", etc) tends to be more stable than the underlying Interface Index, since such stability is required/desired when interface names are employed in network configuration (firewall rules, etc.). The stability properties of Interface Names depend on implementation details, such as what is the namespace used for Interface Names. For example, "generic" interface names such as "eth0" or "wlan0" will generally be invariant with respect to network interface card replacements. On the other hand, vendor-dependent interface names such as "rtk0" or the like will generally change when a network interface card is replaced with one from a different vendor.

We note that Interface Names might still change when network interfaces are added or removed once the system has been bootstrapped (for example, consider Universal Serial Bus-based network interface cards which might be added or removed once the system has been bootstrapped).

### **[A.3](#). Link-layer Addresses**

Link-layer addresses typically provide for unique identifiers for network interfaces; although, for obvious reasons, they generally change when a network interface card is replaced. In scenarios where neither Interface Indexes nor Interface Names have the stability



properties specified in [Section 3](#) for Net\_Iface, an implementation might want to employ the link-layer address of the interface for the Net\_Iface parameter, albeit at the expense of making the corresponding IPv6 addresses dependent on the underlying network interface card (i.e., the corresponding IPv6 address would typically change upon replacement of the underlying network interface card).

#### **[A.4.](#) Logical Network Service Identity**

Host operating systems with a conception of logical network service identity, distinct from network interface identity or index, may keep a Universally Unique Identifier (UUID) [[RFC4122](#)] or similar identifier with the stability properties appropriate for use as the Net\_Iface parameter.



## Author's Address

Fernando Gont  
SI6 Networks / UTN-FRH  
Evaristo Carriego 2644  
Haedo, Provincia de Buenos Aires 1706  
Argentina

Phone: +54 11 4650 8472

Email: [fgont@si6networks.com](mailto:fgont@si6networks.com)

URI: <http://www.si6networks.com>