Article

Not peer-reviewed version

Securing IPv6 Neighbor Discovery Address Resolution with Voucher-Based Addressing

Zachary T Puhl and Jinhua Guo *

Posted Date: 24 June 2024

doi: 10.20944/preprints202406.1561.v1

Keywords: IPv6; security; networking; NDP; neighbor discovery; privacy; voucher-based addressing

Article

Securing IPv6 Neighbor Discovery Address Resolution with Voucher-Based Addressing

Zachary T. Puhl and Jinhua Guo

University of Michigan-Dearborn; Dearborn, MI, United State; zpuhl@umich.edu
* Correspondence: jinhua@umich.edu

Abstract: The majority of local IPv6 networks continue to remain insecure and vulnerable to neighbor spoofing attacks. The Secure Neighbor Discovery (SEND) standard and its concomitant Cryptographically Generated Addressing (CGA) scheme were accepted by large standards bodies to codify practical mitigations. SEND and CGA have never seen widespread adoption due to their complexities, obscurity, costs, compatibility issues, and continued lack of mature implementations. In light of their poor adoption, research since their standardization has continued to find new perspectives and proffer new ideas. The orthodox solutions for securing Neighbor Discovery have historically struggled to successfully harmonize three core ideals: simplicity, flexibility, and privacy preservation. This research introduces Voucher-Based Addressing, a low-configuration, low-cost, and high-impact alternative to IPv6 address generation methods. It secures the Neighbor Discovery address resolution process while remaining simple, highly adaptable, indistinguishable, and privacy-focused. Applying a unique concoction of cryptographic key derivation functions, link-layer address binding, and neighbor consensus on the parameters of address generation, resolved address bindings are verifiable without the need for complex techniques that have hindered the adoption of canonical specifications.

Keywords: IPv6; security; networking; NDP; neighbor discovery; spoofing; privacy; voucher-based addressing

1. Introduction

The Neighbor Discovery Protocol (NDP) for Internet Protocol version 6 (IPv6) was first introduced in 1996 by [1], revised in 1998 by [2], and published in its current version in 2007 as [3]. It is a protocol extension of ICMPv6 [4] used by neighboring nodes in a local network to discover each others' presence, to detect routers, to self-determine addresses, to resolve each others' link-layer addresses, and to maintain details about the reachability of known, active neighbors. A relevant and focal weakness of NDP is the potential for malicious neighbor spoofing attacks, whereby a malevolent neighbor can insert itself into the path of traffic between two other neighbors by intercepting and falsifying Neighbor Discovery Address Resolution (NDAR) messages. The category of threat vectors exploited by these spoofing-related traffic redirections is labeled "on-path" attacks (historically: Man In The Middle, or MITM, attacks). On-path attacks are a critical concern for network administrators because confidentiality, integrity, and availability may all be compromised by interception of traffic at a local scope, where the least amount of network traffic is likely to be encrypted or otherwise secured.

The Secure Neighbor Discovery (SEND; RFC 3971 [5]) and Cryptographically Generated Addresses (CGA; RFC 3972 [6]) specifications were conceived to address the neighbor spoofing problem and other NDP security concerns. Mitigation of on-path attacks with SEND and CGAs has been well established for some time now, but these mitigations have never received widespread adoption in practice. Considering the arcaneness and obscurity of IPv6 outside of academia, networking enthusiast communities, or IETF circles, SEND and CGA descend into an even greater shroud of obscurity that unfortunately keeps them beyond the purview of the mainstream. Furthermore, any existing implementations of SEND and CGA are immature, having little to no

 doi:10.20944/preprints202406.1561.v1

backing or validation [7], and each implementation brings wildly varying effects on network performance across different systems [8].

Keeping in mind the mistakes of the past and the paths already paved, a new solution should be proposed for preventing NDP-driven on-path attacks. This solution should do so while remaining viable and efficient for all adopting devices, with a conception that necessitates the balance of three key aspects: privacy, flexibility, and simplicity. This research presents the Voucher-Based Addressing specification (VBA; also Voucher-Based Addresses, VBAs) to satisfy this requirement.

VBA is a low-complexity, high-impact, privacy-conscious, optional proposal that secures local networks from NDP-driven on-path threats via an alternative address generation technique. Interface link-layer addresses are bound to sets of deterministic IP addresses, in a process which can be repeated by local neighbors to perform address verification. By coupling link-layer identifiers and network-layer identifiers together, the common NDAR procedure for resolving this type of binding can guarantee that only legitimate bindings are used and statefully maintained. All generated addresses appear random to off-link nodes and potential trackers, which ensures node privacy, especially for mobile nodes. Locally distributed vouchers play a key role in controlling, obfuscating, and adjusting the parameters used to create and verify these resultant VBAs.

A key purpose of this research is to produce an alternative to the widely accepted SEND and CGA standards–rather than augment, replace, or survey them–in order to foster new perspectives or to renew interest in solving these longstanding IPv6 security issues. Starting with Section II, this work provides more background about the problem it attempts to solve and collates some other research which has been done to solve relevant NDP issues. A conceptual overview for VBAs is given in Section III. Relevant benchmarks and other experimental, implementation-based results are then described in Section IV. Section V includes a subsequent discussion of some of the miscellaneous practicalities of this work. Section VI then concludes by outlining some potential future research and summarizing the implications of this work.

2. Background & Motivation

Like many early and formative internet protocols, security unfortunately became an afterthought in the design of NDP: presumably sacrificed based on the need for protocol optimizations that matched the performance of more limited hardware constraints of the era. The widespread adoption of NDP occurred before its threat models were formally cataloged in RFC 3756 [9] and before the specification of Secure Neighbor Discovery in RFC 3971 [5] with its complementary Cryptographically Generated Addresses in RFC 3972 [6]. Arkko et al. in 2002 [10] first detailed the various vulnerabilities of Neighbor Discovery: neighbor spoofing, Router Advertisement spoofing, bogus prefix denial-of-service attacks, Duplicate Address Detection attacks, and attacks based on falsely advertised configurations. Many of these remain trivial to execute in normative networks as shown by Anbar et al. as recently as 2016 [11], almost a decade after the final revision of the NDP specification. These problems linger in modern IPv6 deployments, which unfortunately do not receive measurable security attention and considerations as compared to their IPv4 counterparts.

Among its list of insecurities, NDP harbors a dangerous capability for a neighbor spoofing threats, leading to trivial on-path attacks. By successfully advertising a spoofed link-layer address binding with a victim IP address, an attacker can redirect frames from neighbors to itself, allowing it to inspect and capture private packet data on the forwarding path before then forwarding it to its original unicast target. This initial spoof and redirection is sometimes termed “cache poisoning” to express that each misled neighbor falsely caches a binding (i.e., association) between a neighbor IP address and a malicious link-layer address, rather than a genuine target’s link-layer address. If an attacker successfully spoofs link-layer address bindings for both target IP addresses in a local exchange, then it can insert itself on-path between the two communicating neighbors and transparently observe passing network traffic. Any upper-layer protocol without encryption then becomes susceptible to this attack, having atrocious privacy and security implications at the local network scope where traffic has the highest chance to be unencrypted and insecure.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

2.1. Classic Neighbor Spoofing Attacks

The 'classic' neighbor spoofing attack targets two nodes who have established existing and Reachable-state cache entries between them. In its simplest form, it overhears an Address Resolution transaction and follows the completed exchange with an overriding advertisement packet to the target of the attack. RFC 4861 [3] makes a note about nodes receiving unsolicited Neighbor Advertisements: "The Override flag MAY be set to either zero or one. In either case, neighboring nodes will immediately change the state of their Neighbor Cache entries for the Target Address to STALE, prompting them to verify the path for reachability. If the Override flag is set to one, neighboring nodes will install the new link-layer address in their caches. Otherwise, they will ignore the new link-layer address, choosing instead to probe the cached address."

Such a mechanism is put in place to allow the target of Neighbor Discovery proxying to assert its own link-layer address as being reachable on-link directly, rather than letting traffic slowly meander to it through the proxy indirectly. Due to this tradeoff made by the NDP protocol specification, this advertised override forces the target node to update the link-layer address binding for the victim to the spoofed address.

In step 1 of Figure 1, Node A is the solicitor who asks for the link-layer to IP address binding from the address fe80::b. Since Node A does not know the target's link-layer address yet (and thus where to forward frames), a solicited-node multicast address is used instead which utilizes the last 24 bits of the target IP address. Any node can be subscribed to the solicited-node multicast group without authorization, so in the Figure both Node B (the legitimate target) and Node C (the listening threat actor) receive the multicast NS packet. Notice that in step 1 Node A also includes an "SLLAO" NDP option with its NS in order to let receivers know the reverse path on the link-layer to find fe80::a (i.e., the MAC address 11-22-33-44-55-aa).

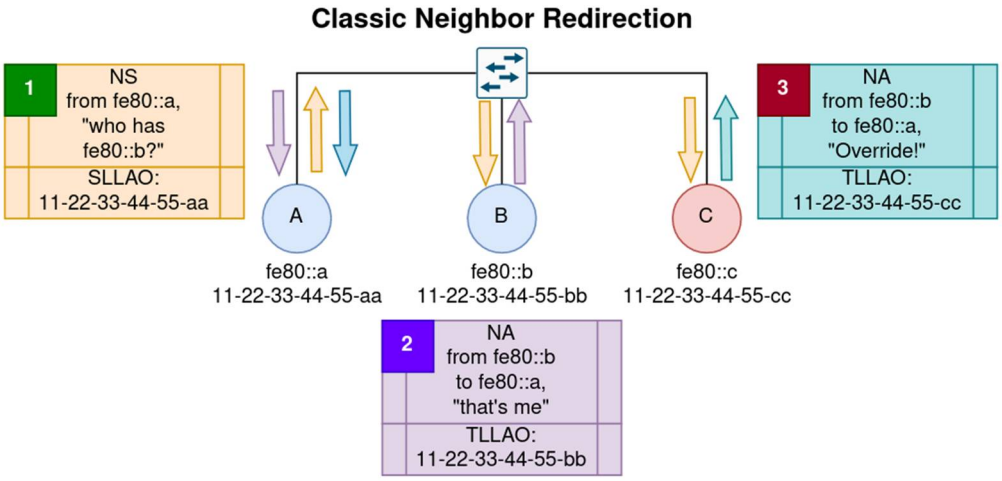


Figure 1. A classic Neighbor Discovery neighbor spoofing (traffic redirection; on-path) attack. After the normative address resolution process completes in steps 1 and 2, the listening malicious Node C sends a spoofed Neighbor Advertisement in step 3 to override the Link-Layer Address value in Node A's Neighbor Cache. Node A now unknowingly harbors a "poisoned" cache entry.

In step 2, Node B receives the NS, pre-caches the link-layer binding from the SLLAO, and responds with its legitimate MAC address in a unicast advertisement packet to Node A at fe80::a. After some brief delay, Node C sends a spoofed unicast Override NA in step 3 with its own link-layer address (11-22-33-44-55-cc) as the reported binding for fe80::b. Note that Node C does not need to await this NDAR process in order to send an Override NA if it already knows the link-layer address of Node A.

Node A has now been poisoned: it subsequently updates its Neighbor Cache for fe80::b to the spoofed link-layer address because of the Override, for which there is no authentication requirement. Packets destined for fe80::b will now be sent to Node C, who can read and interact with the data

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

before forwarding it down the path to Node B, where it will be received without knowledge that Node C had any interaction with it. If Node C repeats this process oppositely with Node B's traffic en route to Node A, then a completely transparent and two-way on-path attack is successfully set up. Node C would achieve a complete arbitration of the traffic stream between Nodes A and B..

2.2. Eager Neighbor Spoofing Attacks

A less well-known and much more powerful cache poisoning attack opportunity exists in an optimization related to Neighbor Solicitations. The Source Link-Layer Address Option (SLLAO) stub can be included with NS messages to indicate the intended link-layer address binding of the IP source address on the NS packet. This is done so that receiving nodes will not be required to reverse-probe the sender's IP address for a link-layer address binding during NDAR transactions.

The NDP specification in RFC 4861 [3] reads: "If the [NS] is being sent to a solicited-node multicast address, the sender MUST include its link-layer address (if it has one) as a Source Link-Layer Address option. Otherwise, the sender SHOULD include its link-layer address (if it has one) as a Source Link-Layer Address option. Including the source link-layer address in a multicast [NS] is required to give the target an address to which it can send the [NA]." This optimization to NDP allows NDAR transactions to occur much faster, but weakens the protocol due to its blind trust of senders and subsequent automatic caching. Since NS packets are generally the first contact in an NDP transaction, there is no straightforward way to recognize and mitigate this 'eager' attack unless the NS occurs for an IP address that already has a binding within a target's Neighbor Cache.

Figure 2 demonstrates the simplicity and potency of a more eager neighbor spoofing attack: only two steps are required and the poisoning can begin from the very start of an apparently legitimate and innocuous NDP transaction. In step 1, malicious Node C creates a solicited-node multicast Neighbor Solicitation with a spoofed IP Source Address of fe80::b (Node B's address). The target address of address resolution is not important, but the attacker will need to aim the NS at the multicast group for which it knows the target node (Node A) is a member, while trying to avoid sending the NS to the victim (Node B). By random chance, this will almost always be the case anyway for a 24-bit address suffix used to derive a solicited-node multicast address. When Node A receives the NS packet, it will preemptively cache the link-layer address found in the SLLAO and bind it to the IP Source Address of the packet; so Node A creates the entry [fe80::b → 11-22-33-44-55-cc].

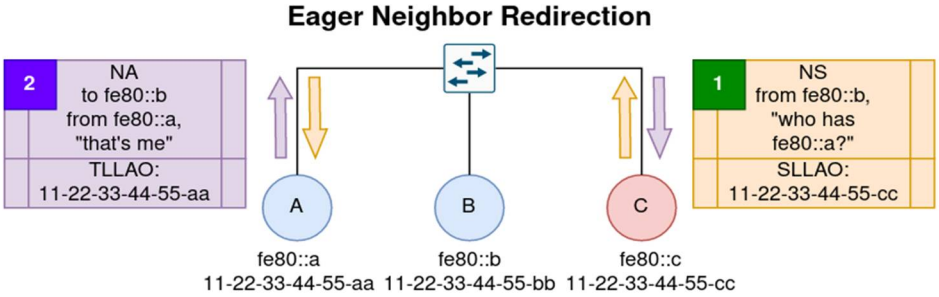


Figure 2. A more subversive, 'eager' approach to preemptively poisoning a target's Neighbor Cache without requiring any prior interaction. Nodes receiving SLLAO stubs on certain NDP messages are dictated by the protocol specification to accept them as-is for the sake of protocol optimization. This optimization is performed so that NS receivers can respond with unicast messages to the sending link-layer address without needing reverse address resolution and without a multicasted response.

In step 2, Node A happily advertises itself through a unicast NA message as fe80::a with its legitimate MAC address 11-22-33-44-55-aa, as part of an expected and normal NDAR response. Node C is now free to receive data frames originally intended for Node B, transparently forwarding them, blocking them, or modifying them in-transit. It is also important to note that this attack can be effective in NDP augmentations requiring a Nonce field to be attached to NS/NA packets, because this eager poisoning can dishonestly forge any Nonce value as a virtue of being the NDAR initiator (i.e., solicitor).

 doi:10.20944/preprints202406.1561.v1

2.3. Motivating Attack Commonalities

There exists a glaring commonality between all of these various NDP attack surfaces presenting an opportunity for on-path attacks: caching. Regardless of how the cached value is updated or exposed to poisoning, the efficacy of the attack relies solely on making a 'bad' update to the target's Neighbor Cache. Attacks successfully poisoning a target's Neighbor Cache often then only need to maintain the malicious entries through the normative NUD process. This observation reveals that by merely guarding the cache at the target node, through some form of sender validation or challenge-response authentication, attacks resulting in false updates to (or creations of) Neighbor Cache entries can be mitigated altogether.

This research proposes an alternative to securing NDP against neighbor spoofing attacks that relies on this idea. It identifies and applies what should be considered three crucial aspects for the adoption of any amendment to Neighbor Discovery: simplicity, flexibility, and privacy preservation. Voucher-Based Addressing is thus introduced as a possible candidate to mitigate the neighbor spoofing threat in local networks.

3. Voucher-Based Addressing

3.1. Terminology

A glossary of terms and acronyms related to Voucher-Based Addressing is necessary to index, organize, and comprehend the many different aspects of its proposal. To acquire more prerequisite context, please see Section 2.1 of the NDP specification in RFC 4861 [3] for definitions of the following terms: neighbor, node, interface, link, address, router, host, on-link, off-link, IP, ICMP, packet, target, Neighbor Unreachability Detection (NUD), Neighbor Cache (NC), and all NDP ICMP packet types (Redirect, RS, RA, NS, and NA).

Table 1. Related and necessary terminology as a reference for this research.

SEND	SEcure Neighbor Discovery (RFC 3971 [5]).
CGA	Cryptographically Generated Address (RFC 3972 [6]).
NDAR	Neighbor Discovery Address Resolution (see Section 7.2 of RFC 4861 [3]).
LLID	A shorthand representation for the terms "Link Layer Address" or "Link Layer Identifier". Both terms are synonymous and describe any individual link-layer identifier for a network interface. This work generally uses this term synonymously with MAC addresses.
SLLAO	Source Link-Layer Address Option. An NDP option indicating an LLID, typically of the NDAR initiator.
TLLAO	Target Link-Layer Address Option. An NDP option indicating an LLID, typically of the NDAR target.
SLAAC	Stateless Address Autoconfiguration (RFC 4862 [12]).
VBA	Could mean one of two things depending on context: Voucher-Based Addressing (such as "the VBA-enabled subnet" or "VBA mandates this").

 doi:10.20944/preprints202406.1561.v1

	Voucher-Based Address (such as "a VBA" or "using VBAs"). A unicast IPv6 address generated by a mixture of Link Voucher details, network interface details, and subnet details. The term "VBA" might be used in lieu of "IP address", but an IP address may also be a VBA. There is no special value contained within an IP address to indicate that it is a VBA.
LV	An NDP Link Voucher option. A data payload distributed by a responsible neighbor. Its details are statefully maintained on receiving neighbors and are used in both generating and verifying VBAs.
VB	The Voucher Bearer, a neighbor that is solely responsible for dissemination of the current voucher through NDP LV options. This node is authorized by any potential guards and infrastructure to transmit Router Advertisements or Redirect messages with LV options attached.
IEM	Interface Enforcement Mode. An interface-level, mutable operating mode which controls interface VBA generation and verification behaviors.
Binding	Used primarily to describe a coupling between two types of addresses on different layers of the OSI Reference Model [13]. In the case of VBA, it is usually used in reference to link-layer identifiers as bound to network-layer identifiers.
LL2IP	Used to shorten the phrase "link-layer-address-to-IP-address" when discussing address bindings.
KDF	Key Derivation Function, as defined in Section 3 of RFC 8018 [14].
Hextet	A 16-bit aggregation; data that is 16 bits in size.

3.2. Threat Model

In the projected threat model for the local network, threat actors are only interested in stealthy on-path attacks resulting from neighbor spoofing exploitation. Modeled threat actors are not concerned with network disruptions or denial of service attacks; they would prefer to remain quiet and unseen. For the most part, the success of an on-path attack arbitrating and examining unicast messages is dependent upon the threat actor remaining undiscovered on the path between two victim nodes in the first place. This model assumes that no two LLIDs within the target broadcast domain can be the same value or be spoofed in the network without obvious disruptions to network activity. It also simultaneously assumes an insecure link layer.

For external networks, the threat model includes risks to the privacy of an interface communicating off-link. Nodes can be remotely tracked, targeted, and even exploited through their unique, global unicast addresses if they are not sufficiently rotated. If an address generation mechanism incorporates link-layer information and does not obscure it in some way, then attacks can be launched against addresses based on what might be revealed from link-layer information. Lastly, address assignment schemes which do not encourage or permit regular primary address rotations are subjected to these threats and can be a valuable attack vector for targeting victims.

3.3. Design Goals & Overview

MAC spoofing attacks as presented in Section 3.B of [15] are a driving factor of why Voucher-Based Addressing functions in the first place. When two nodes share the same link-layer identifier in

 doi:10.20944/preprints202406.1561.v1

a switched network, frames will unreliably be forwarded to one of them based on who most recently communicated through the switch. A flip-flopping of frame delivery at the link layer causes a confusion of higher-level protocols and will most likely result in a denial of service attack on the legitimate node owning the MAC, thus disrupting the network. With this context, the principle of MAC address uniqueness per broadcast domain can be established.

During the address resolution process, the goal is to associate a target IP address with its underlying link-layer address to which frames can be forwarded and switched. When LLIDs become determining factors for higher-layer abstractions, such as IP addresses in the case of VBA, then 'bindings' are created between the LLIDs and the abstractions. Since VBA generation depends on both these bindings and the principle of MAC address uniqueness at the per-interface scope, VBA verifications are considered valid proofs of MAC address ownership so long as the interface remains present on-link. The verification process is employed at specific times during NDAR transactions, independently mirroring the VBA generation process.

Verifications are parameterized by (1) various inputs which identify the target node during NDAR (the IP and MAC addresses from NDP packets), and (2) inputs which lie beyond the control of the target node. Such external information resides within locally disseminated Link Voucher details agreed upon by all neighbors. Due to the utilization of LL2IP bindings in both generating and verifying VBAs, it is impossible to forge an association of an IP address to an LLID that cannot be bound to it. Enforcing binding verification secures the NDAR process against issues of LL2IP binding impersonation and thus against neighbor spoofing threats.

Creating an IP address which incorporates verbatim the direct value of the LLID would suffice if the goals of bindings in VBA only included validating address ownership. For example, EUI-64 is a long-established address assignment methodology that employs direct implantation of an LLID in assigned addresses. This would create an easily verifiable LL2IP binding for each interface, but it would not incorporate a focus on privacy because the LLID of the interface is publicly exposed and the address remains fixed. Additionally, EUI-64 is a one-to-one LL2IP binding, whereas VBA seeks to derive many IP addresses for each LLID.

To meet the need for privacy, simplicity, and flexibility in generating new, obscure IP addresses during SLAAC self-addressing, VBA employs a trivial hashing process. Using techniques based on hash functions ensures that any LLID of an arbitrary length can be reliably bound to an irreversible address suffix that is a desirable and fixed length. However, simply hashing an LLID will still only manifest a one-to-one binding. Many formalized IPv6 address generation schemes already offer ways to derive many privacy-focused addresses on a single interface (e.g., Section 5 and Appendix A.3 of RFC 7217 [16]). Addresses generated through these schemes are intentionally obfuscated to preserve the privacy of the node, unless reversing parties are aware of all input parameters used by the deterministic address generation function. VBA solves this one-to-one addressing problem by employing a variant of salted hashing and key derivation functions.

VBA strikes a careful balance of (1) keeping off-link nodes unaware of local voucher information, and (2) ensuring on-link nodes are aware of all parameters used to generate any neighbor's IP address(es). Off-link parties cannot derive a target's bound LLID because they cannot receive NDP messages from the target's broadcast domain, nor can they determine the binding from any patterns within the target's address itself. VBAs will always appear random as a consequence of utilizing the outputs of deterministic hash functions.

3.4. Address Generation & Verification

To gain a further advantage, this work elevates simple hashing techniques to the use of key derivation functions (KDFs), enabling a set of one-to-many LL2IP bindings and also enforcing a minimum address computation time. KDFs accept input iteration values specifying how many times the driving pseudo-random function (or underlying hash function) must be iterated before producing a final result. In VBA generation, various inputs that specifically identify the target node, as well as the chosen iterations count, are computed by a known KDF into a hash value. The input iterations count is then obscured and planted into the generated VBA adjacent to a slice of the resultant hash, such that the following three components are an inherent value exchanged by any NDAR transaction: (1) the target's reported LLID and IP address (i.e., VBA itself); (2) a portion of the KDF's hash output

 doi:10.20944/preprints202406.1561.v1

(embedded within the VBA); and (3) the iterations count that was supplied when computing the KDF hash (also embedded within the VBA).

Interfaces using the VBA generation procedure during SLAAC therefore enforce that all three aforementioned items are bound together and conveyed to neighbors, alongside on-link voucher details, to produce the same output VBA during verification. Each increment or decrement of the iterations count value produces an entirely new, seemingly random address with no correlation to another VBA produced from a different iterations count value. Nodes falsifying any information during the NDAR process will be rejected in communication by neighbors who cannot successfully verify the illegitimate bindings.

VBAs are composed of three key components in order from most-significant to least-significant byte: the 64-bit subnet prefix, a 16-bit Z value indicating the encoded iterations (L) value used to compute the address, and a 48-bit hash-derived address suffix (H). If the subnet is less than 64 bits in length, then the remaining gap between the end of the subnet prefix and the beginning of Z is always populated with pseudo-random noise by the generating node. VBA generation is not compatible with networks whose subnet prefixes exceed 64 bits in length. A VBA contains only a partial conveyance of the information required for neighbors to reconstruct and therefore verify the address binding. Figure 3 provides a simple visual representation of how an IPv6 address is partitioned.

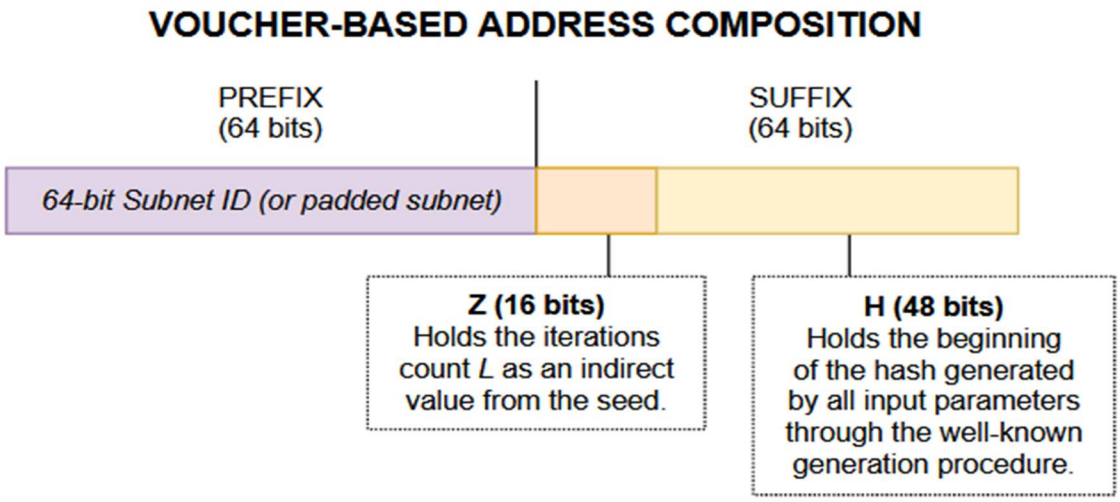


Figure 3. The overall structure of a Voucher-Based Address.

The interface identifier for all VBAs, also called the Suffix, embeds two important details for verification. A 16-bit Z value is calculated as a bitwise complement of the XOR of the 16-bit L value (the iterations count used in the KDF function K) and the first hextet of the current voucher seed value. The Z value is used to ensure the same input iterations count value, L, will be unique across different voucher seeds to provide enhanced address privacy.

The L value is a significant member of the generated VBA: this parameter controls how many times the KDF function specified in the voucher is iterated to produce the resulting hash value from which H is derived. Increasing this value increases both the work required to verify the VBA and the work necessary to discover potential collisions with H. H is the other value embedded in the VBA which consists of 48 bits acquired from computing the resulting KDF function with L iterations. The first 8 bytes of the resultant KDF hash are used in formulating the Suffix of the VBA, where its first hextet (bytes 1 and 2) is replaced with the Z value as shown in Figure 4.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

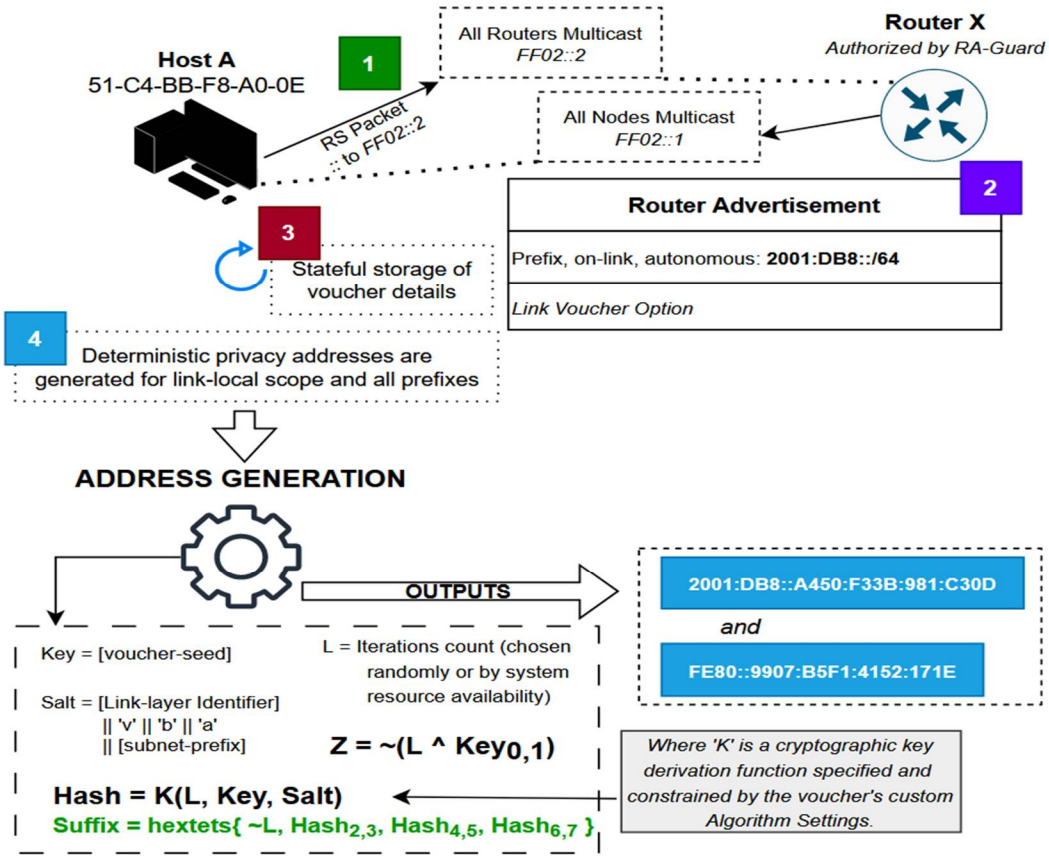


Figure 4. The Voucher-Based Address generation procedure is used via SLAAC to generate all initial interface addresses from the interface on Host A. Router X is a Voucher Bearer authorized by local administrative policy to delegate Link Voucher information to its neighbors.

The address generation algorithm is detailed procedurally as follows:

1. A node connects to a network and discovers link VBA compatibility from a Link Voucher option obtained upon router solicitation.
2. The local L value is chosen based on (1) node preference, (2) intended VBA difficulty, or (3) random selection. The Link Voucher contains instructions for KDF parameters and algorithm selection as well as the 128-bit seed value to use.
3. The KDF salt is created as a variable-length concatenation of a few different inputs, in the order specified by the list below. The adjective 'raw' dictates specifically binary values, not hexadecimal string notations of said values.

a. The raw LLID of the network interface on which VBAs are being generated, in network byte order. Since the salt value has a varying length, this is not required to be an IEEE 802 MAC address. It must only represent the LLID to which the VBA(s) is to be bound and which will be provided to verifying nodes during NDAR transactions.

b. The string "vba".

c. The raw Prefix (subnet prefix) value, in network byte order. This must match the prefix that will be prepended to the final VBA given during the NDAR processes.

2. The final address Suffix is computed:

a. The first 16 bits are the bitwise complement of an XOR between the node-selected iterations count L and the first hextet of the known voucher seed.

b. The least significant 48 bits are 6 sequential bytes from the computed KDF hash H, skipping its first hextet (two bytes).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

When enabled and enforced by a receiving interface's Interface Enforcement Mode, the verification process uses the information embedded within the IP address that is provided during an NDAR exchange. VBA verification simply mirrors the generation of the reported VBA locally; Figure 5 illustrates this process. If the reconstructed address of the target node does not match the IP address reported in the NDAR transaction, then the VBA is invalid and communication with the node is denied according to the verifier's IEM setting. The Link Voucher ('LV' in the Figure) is always retrieved from the state preserved on the verifying interface, and never from an external source that is not the acting Voucher Bearer. If the verification procedure fails due to a voucher mismatch between nodes A and B, then there is most likely either (1) a synchronization problem, (2) malicious activity, or (3) an issue with multiple vouchers being distributed simultaneously.

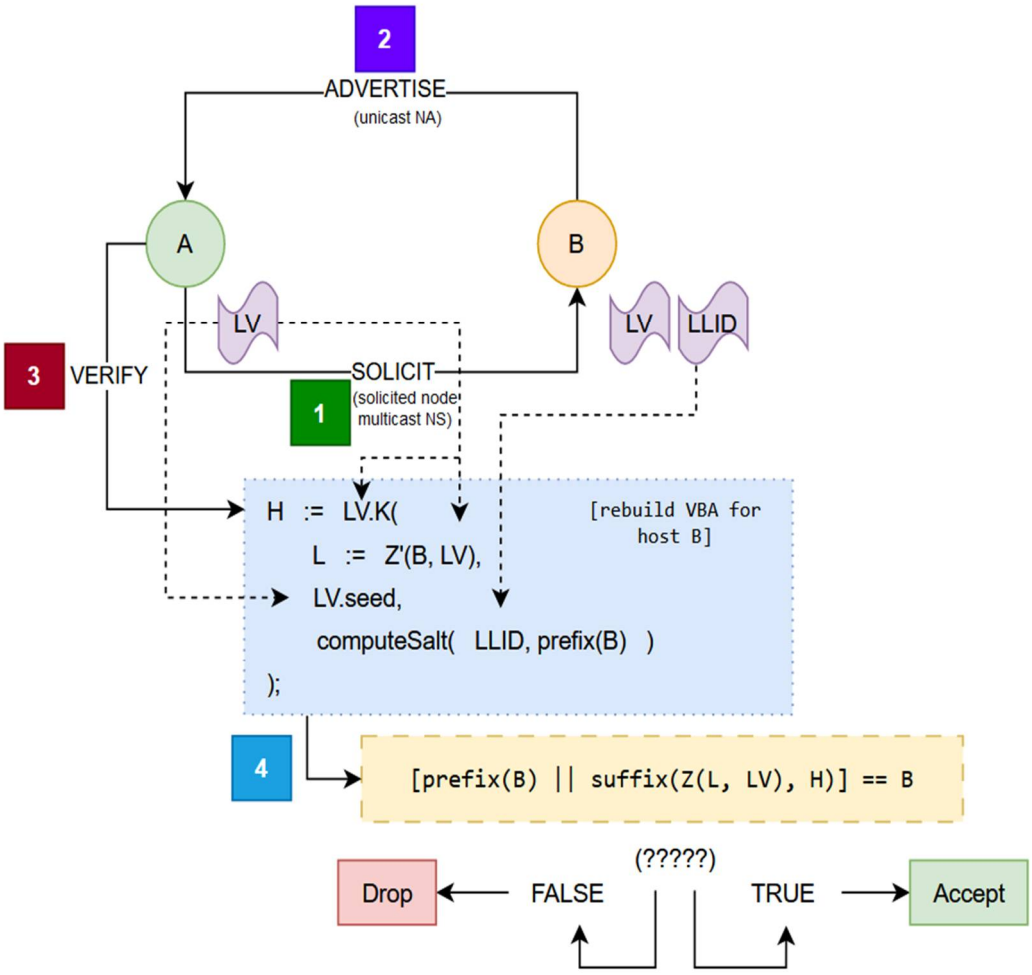


Figure 5. The VBA verification procedure. The generation process is repeated at the verifier using various values known about the neighbor and the current voucher parameters.

During Moment 1 from the Figure, Node A can choose to attach a Source Link-Layer Address Option (SLLAO) to its solicitation, which will cause Node B to verify its binding with the IP Sender Address from the NS packet. The Z' function returns the L value (iterations count) embedded within Node B's address. This function is the opposite of Z from the address generation process: it uses an input address to determine L rather than using an input L to determine an encoded hextet. Despite the different inputs, the naming alludes to the opposite purposes for each function.

VBA does little to modify Neighbor Discovery, instead opting to change the behavior and decision-making processes of its underlying software implementations. Figure 6 expresses the application of these changes in a practical scenario, where both nodes are required to engage their VBA verification processes between certain NDAR events. In summary, VBA requires modifications to the following NDP behaviors on VBA-enabled interfaces:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

- Any new LLIDs on neighbors have an impossibly low chance of organically producing the same VBA as one already cached by verifying neighbors. Such an unlikelihood implies that any Neighbor Advertisement (NA) packets targeting an already-cached IP address which is not in the INCOMPLETE state must be ignored if an included Target Link Layer Address Option (TLLAO) attempts to update the cache entry state or change its binding to a different LLID. This will prevent threat actors from maliciously triggering costly VBA verification processes where they are not necessary.
- The value of an LLID within a Neighbor Solicitation (NS) packet must likewise never update any existing cache entry for the given IP Source Address. This is because it is a statistical improbability for any known VBA to have been genuinely formed from a different LLID when the voucher has not changed.
- Any supposed urgent updates about underlying details for a known VBA are unnecessary. The Override flag in received NAs must not be able to freely update the underlying LLID or state of any cache entry. Some devices might wish to support a more lax AGVL IEM which allows compatibility with static unicast addresses on-link. In the case where the IEM is set to AGVL, the Override flag in NAs should not be ignored, in order to let static addresses immediately notify neighbors of a change in their interface LLIDs.

VBA verification is a 'shim' software process–a small functionality added as a process between two existing procedures–that prescreens incoming requests to insert or update cached bindings according to the normative Neighbor Discovery process. In Figure 6, Host B calls upon the verification shim of a verifying neighbor, Host A, in Moment 4. If the shim rejects the binding from entering or updating the Neighbor Cache, then Host A will be denied from properly forwarding data frames to the advertising Host B. This is because a cache entry in the REACHABLE state does not exist and will not be created. Prefiltering in such a manner immediately eliminates any threat actor's opportunity to forge NDAR messages or to redirect traffic maliciously.

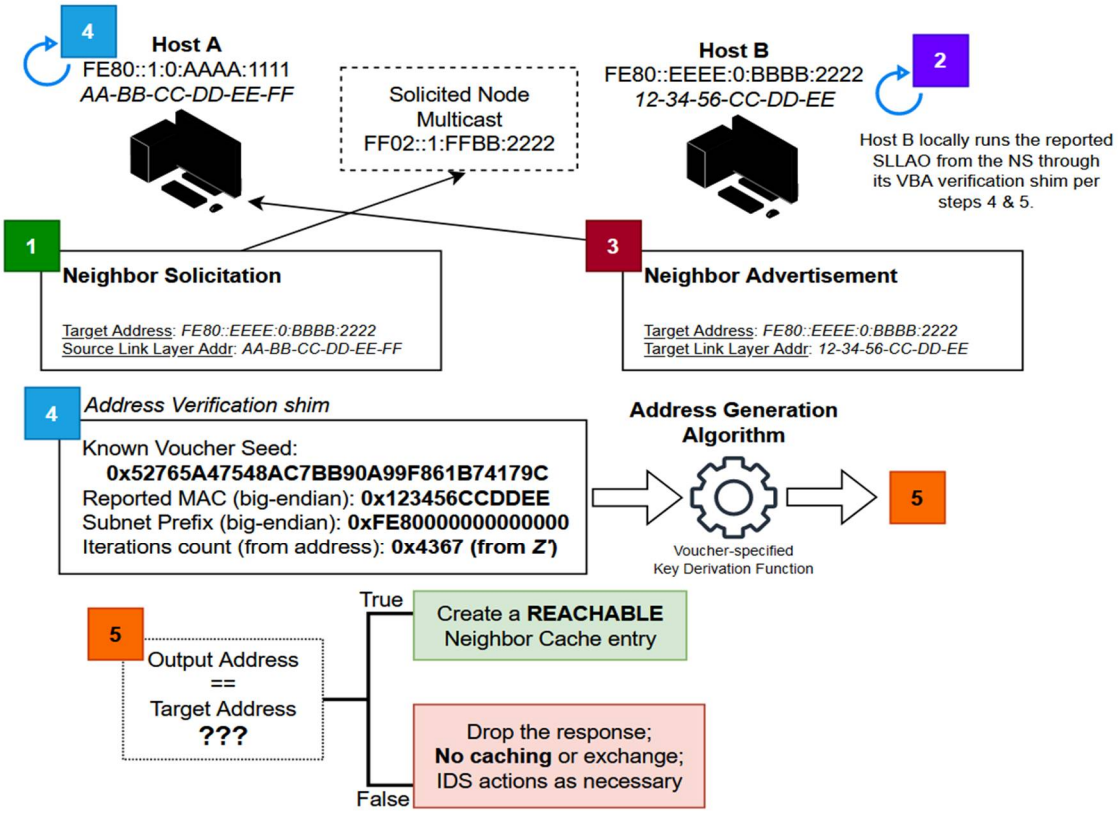


Figure 6. VBA processes do not modify the typical Neighbor Discovery process or exchange. Instead, software local to each interface, if equipped, will act to verify received IP addresses during NDAR transactions based on the interface's selected Enforcement Mode.

 doi:10.20944/preprints202406.1561.v1

Employing the verification shim results in repeated KDF computations that could impact performance significantly for systems with minimal resources, so the shim must be optimized and called as seldom as possible. For the sake of optimization, NUD exchanges must not call upon the shim when none of the NDAR parameters (i.e., the IP address or the LLID) are being changed. Incoming NDAR messages failing VBA verification must be immediately ignored, and NC entries must not be created or updated as a result of their receipt. Nodes likewise must not respond to any packets failing the verification process.

There are a few situations when address resolution packets cannot be optimized and must explicitly pass through the VBA verification shim for approval:

- An NS, RS, or RA packet is received with an SLLAO attached and an NC entry for the IP Source Address is not already present.
- An NA or Redirect packet is received for a Target Address whose NC entry is in the INCOMPLETE state.
- An NA packet is received and the Override flag is set, and the receiving interface is using the AGVL IEM.

The above list is perhaps not all-inclusive and does not consider other extensions to NDP which may allow certain messages to modify or insert cache entries. Except for forward progress indications through NUD, NDAR transactions of any type seeking to update or create any active NC entry must be pipelined through the verification shim first, depending on the current IEM.

3.5. Interface Enforcement Modes

Each interface employing VBA must have the option at all times to set a single local Interface Enforcement Mode (IEM) which determines its handling of NDP messages and VBA. IEMs are a flexibility feature allowing a granular, per-interface setting that adjusts the behaviors of each interface in real-time. They are designed to be changeable at any time and for any reason, no matter the operating state of the interface. The IEMs, in order of increasing strictness, are:

- Address Awareness Disabled (AAD): The interface must disable any generation or verification of addresses during NDAR transactions. It must completely withdraw from any activity related to VBA, with the exception that it can still listen for and capture the current voucher state.
- Address Generation Only (AGO): The VBA generation and process is followed during SLAAC self-assignment, but the interface's address verification shim must be disabled for all NDAR transactions.
- Address Generation and Verification with Levels (AGVL): VBA generation and verification is performed, but any failure to verify a neighbor must not be strictly enforced. Purported LL2IP bindings which fail to verify are instead tagged in the Neighbor Cache as "Unsecured" entries, and those which do successfully verify are tagged as "Secured". Secured responses are always preferred over Unsecured ones, which permits verified bindings to receive sole connection priority without denying communication to neighbors whose VBAs do not successfully verify (and for which a Secured response does not exist).
- Address Generation and Verification (AGV): Any purported binding which does not verify must be dropped immediately from the Neighbor Cache. Interfaces opting to use this mode will guarantee themselves protection against neighbor spoofing threats, because they will only ever be able to receive IPv6 packets from valid VBAs. Flexibility with neighbors in mixed networks where some nodes do not have VBA capabilities is consequently revoked.

3.6. Link Voucher Structure & Rules

The Link Voucher is an optional attachment to NDP Router Advertisement and Redirect messages from a responsible Voucher Bearer. Link Vouchers dictate the parameters used by the local network's neighbors to generate and verify VBAs. By agreeing on these shared, distributed parameters during address generation, all neighbors are able to independently incorporate the same information to verify each other's addresses. Establishing a link-local baseline for VBA generation parameters in the voucher enhances node privacy, because off-link nodes will not know all of the address inputs.

 doi:10.20944/preprints202406.1561.v1

Figure 7 shows the binary structure of the Link Voucher NDP option and all its descriptive field names. Each field is completed, either statically or by the Voucher Bearer, as such:

Type

The unique NDP Option Type identifier for Link Vouchers is 63.

Length

The total length of the LV from the Type through its end, inclusive, in units of 8 octets.

Expiration

A 16-bit big-endian value storing the amount of time in seconds that the Link Voucher option should be considered legitimate when an update has not been received from the VB. This value is recommended to be set between 3,600 (1 hour) and 43,200 (12 hours) seconds. Setting the value any lower or higher results in issues with over-rotations and under-rotations, respectively; two situations which can easily cause denial of service attacks when abused.

Reserved

Reserved for future use. This field is set to zero by senders and ignored by receivers.

Timestamp

A 64-bit value representing the local system time of the sender at the moment the Link Voucher option is constructed.

VoucherID

A pseudo-random 32-bit value which uniquely identifies a persistent Link Voucher instance. This must never change between distributions of the same unique voucher.

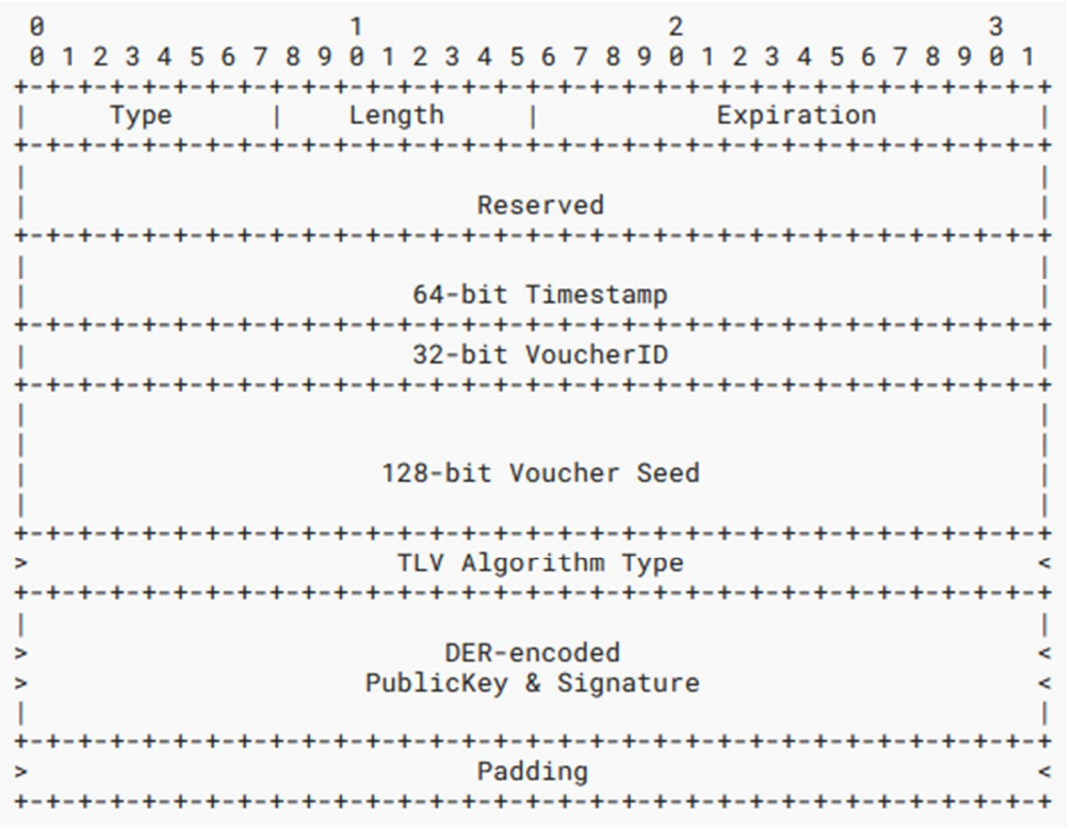
Seed

A 128-bit pseudo-random value used as an input in local VBA generation. This value must be the same for each voucher instance identified by a VoucherID, and it cannot be equal across different VoucherID values.

Algorithm Type

A Type-Length-Value field specifying exactly which type of key derivation function to use in VBA generation and its corresponding baseline difficulty.

ECDSA PublicKey & Signature



 doi:10.20944/preprints202406.1561.v1

Figure 7. The binary structure and fields of the Link Voucher NDP option. This is only considered valid by receivers when attached to Router Advertisement and Redirect packets.

A variable-length field representing a unique public-key value held by the Voucher Bearer issuing the Link Voucher option. It is used to sign the LV option and to authenticate any updates to future LV option details. The exact mechanism of public key management is beyond the scope of this paper. More specifically, this field contains a DER-encoded ECDSA [17] public key of type *SubjectPublicKeyInfo* according to Section 2 of RFC 5480 [18] . The public key structure is followed immediately by an adjacent DER-encoded ECDSA signature, derived using the private key corresponding to *PublicKey*. The ECDSA signature is computed over a series of sequential octets, constructed in the following order:

4. The 16-bit *Expiration* value.
5. The 64-bit *Timestamp* value.
6. The 32-bit *VoucherID* value.
7. The 128-bit *Seed* value.
8. The variable-length contents of the *Algorithm Type* value, including its *Type* and *Length* values.

The algorithm used in signature computation is *ecdsa-with-SHA256*, as defined in Section 3.2 of RFC 5758 [19]. The signature must be a DER-encoded ASN.1 structure of the type *ECDSA-Sig-Value* (Section 2.2.3 of RFC 3279 [20]). The final field appears as the two adjacent DER structures from Figure 8.

Padding

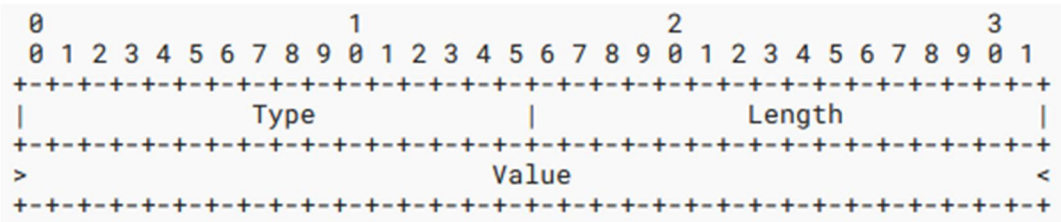
Any extra padding set on the datagram to round its total length to an even 8-octet boundary. This field is always set to zero and is ignored by receivers.

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
  algorithm         ::=  SEQUENCE {
    algorithm   OBJECT IDENTIFIER,
    parameters  ANY DEFINED BY algorithm OPTIONAL
  },
  subjectPublicKey  BIT STRING
}
ECDSA-Sig-Value  ::=  SEQUENCE  {
  r  INTEGER,
  s  INTEGER
}
```

Figure 8. The adjacent DER structure definitions for encoding the ECDSA PublicKey and Signature values in a Link Voucher option.

3.7. Algorithm Type Options

There are three default key derivation algorithms that must be included with all basic implementations of VBA. An Algorithm Type option is a value within LV options that is formatted as a Type-Length-Value (TLV) object, where Type is a numeric identifier uniquely representing a chosen KDF, Length is the width of the total Algorithm Type stub in units of 4 octets, and Value is a compact, binary data format that is zero-padded to the nearest 32-bit (4-octet) boundary. Future versions or extensions of VBA might wish to add and formalize new KDF algorithms and their corresponding Type identifiers.



 doi:10.20944/preprints202406.1561.v1

Figure 9. The basic binary structure of a TLV field that is used in the Algorithm Type field of a Link Voucher option. The Type and Length fields are a maximum width of 16 bits. The angle brackets to the sides of the Value field indicate a variable length field.

The list of the three default KDF Algorithm Type choices is given below:

PBKDF2_SHA256

The Password-Based Key Derivation Function (PBKDF2) is defined in Section 5.2 of RFC 8018 [14]. It is a CPU-bound KDF, use of which might result in significant computation speed disparities between embedded systems and high-performance hardware. It is included primarily for portability, universality, and ease of implementation.

Type: 1

Length: Always 2

Value:

ITERATIONS_FACTOR

A 16-bit integer representing the multiplier of an input KDF iterations count, specified in big-endian format. This value is required to be greater than zero, and receivers of zero values will simply assume '1' instead. This linear scaling factor can be used by a voucher to amplify the baseline cost of computing the PBKDF2 KDF across all neighbors.

Padding

16 bits (2 octets) of padding initialized to zero and ignored by receivers.

Argon2d

The Argon2 algorithm is specified in Section 3 of RFC 9106 [21]. It is a Memory-bound KDF providing significantly less disparate address computation speeds across varying hardware than CPU-bound algorithms like PBKDF2. VBA explicitly opts to use Argon2d instead of Argon2i or Argon2id because the generation of VBAs does not require any resistance to side-channel attacks. Implementations of VBA should always prefer to employ this Algorithm Type over others when there is no specific reason to opt for another Type, provided all participating neighbors have Argon2d support.

The iterations count value is used as the t input value for Argon2d computations. The Argon2 t parameter indicates the number of passes and is used to increase the algorithm's running time regardless of *MemorySize*. To give the parameters in the *Value* field more weight, t is reduced from the KDF's input L value as follows:

t := (L >> 8) + 1

The Argon2 parameters for Secret Value K and Associated Data X are neither used nor distributed by the LV for any reason, and the Tag Length T for Argon2d is set statically to a fixed value of 32. These predefined values assure the Argon2 computation will scale according to a specific projection as the input L value increases.

Type: 10

Length: Always 2

Value:

Parallelism

An 8-bit integer determining how many degrees of parallelism (i.e., lanes) are allowed to run during KDF computation. A value of zero is not acceptable and will instead default to one by receivers.

MemorySize

A 24-bit integer representing the number of kibibytes (KiB) used as space for the KDF computation. This value should be carefully controlled and if possible should take into consideration the computing resources across the link on which the voucher will be distributed. This value is required to be a minimum of 8 * Parallelism, and cannot be set to zero. Receivers will need to adjust the minimum *MemorySize* value accordingly if the value specified does not meet the minimum threshold for the actual degree of *Parallelism* being used.

Scrypt

The Scrypt KDF algorithm is specified in Section 6 of RFC 7914 [22]. It is a Memory-bound KDF that, similar to Argon2, provides less disparate address computation durations across varying hardware than CPU-bound KDF techniques. The iterations count L value is used in part for both the

 doi:10.20944/preprints202406.1561.v1

N and r inputs for Scrypt computations. The Scrypt N parameter indicates the resource cost of running the computation and is required to be a power of 2. The r Scrypt parameter indicates the desired block size. Actual values are computed through the following conversion:

r (Parallelism) := MAX{ 16, (L & 0xFF00) >> 4 } << SCALING_FACTOR

N (Cost) := MAX{ 2, 1 << (L & 0x00FF) } << SCALING_FACTOR

The Scrypt parameter dkLen (derived key length) is set to a fixed value of 32 and cannot differ between implementations. The parallelization parameter p is always set to one and also must not differ between implementations.

Type: 20

Length: Always 2

Value:

SCALING_FACTOR

An 8-bit integer setting the difficulty scaling of the Scrypt algorithm. This value must only be 0 through 5 inclusive. Receivers must always limit the maximum *SCALING_FACTOR* to 5 regardless of whether the received value exceeds 5.

Padding

its (3 octets) of padding initialized to zero and ignored by receivers.

4. Prototypes & Results

4.1. Minimum Difficulties

An average laptop was used to generate a set of VBAs for each VBA Algorithm Type option and to evaluate the time taken by each KDF algorithm for each input iterations count. The testing hardware is not important because all results are relative to one another; that is, since all tests are performed on the same device, their relationships will manifest similar ro how they would across other hardware. With a partial VBA generation reference implementation based on the work in Section III, it is possible to create an application which can spawn VBAs on-the-fly based on any set of dynamic or precompiled parameters.

Using the minimal key derivation function costs for each default algorithm, Figure 10 shows a mostly linear trajectory for the results of each test. As expected, an increase in the input iterations count value is linearly associated with the time taken to compute the VBA. Deviations in the graph are spurious events generated by an active CPU on the testing device switching between various scheduled tasks through the underlying operating system. Even though the results of the graph are not aggregate statistics, the relationships between the data and their interdependencies demonstrate how VBA might perform in practice.

 doi:10.20944/preprints202406.1561.v1

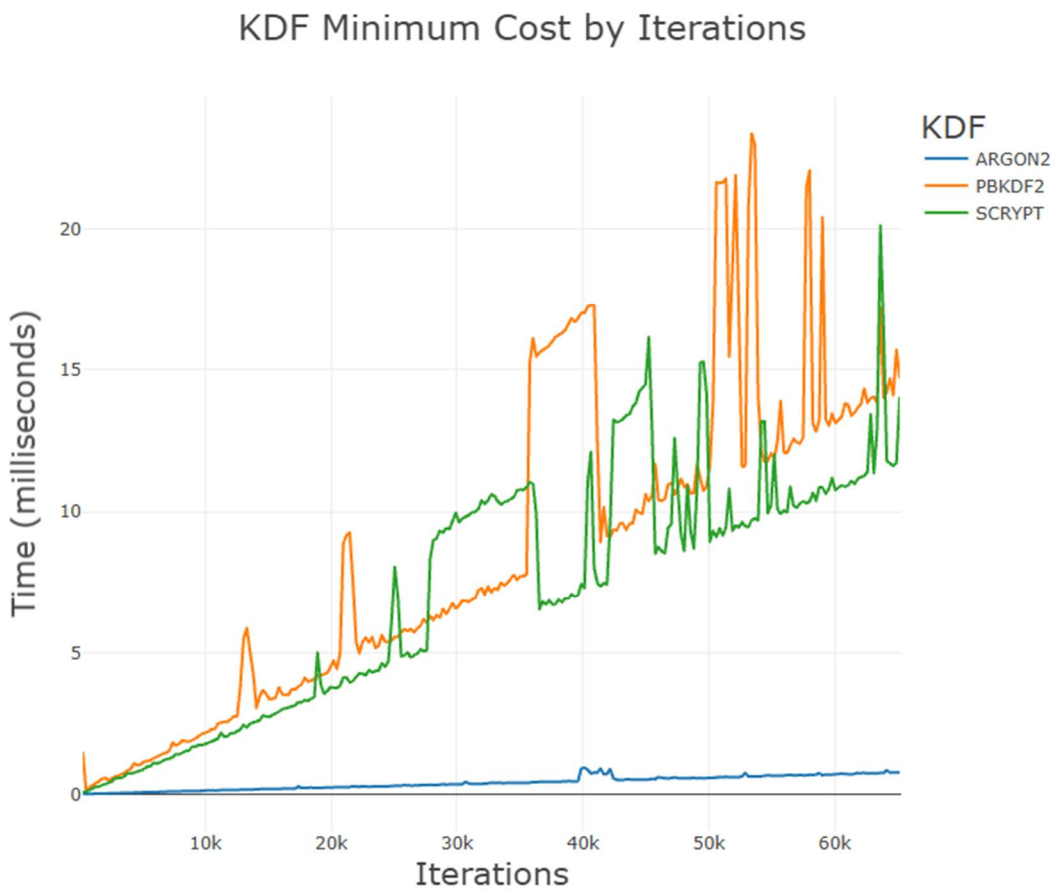


Figure 10. The three default key derivation functions are employed and benchmarked in VBA generation procedures with their minimum possible baseline difficulty settings. Each increase in the iteration count for each KDF expectedly shows mostly linear increases in address generation times. All outliers and deviations from the observable linear pattern are due to spurious slowness of the local processor on which these tests were run.

Argon2 key derivation is highly preferred as the default Algorithm Type for VBA Link Vouchers, because its minimal baseline settings allow for a memory-bound KDF to require relatively little time to compute addresses. This allows link Voucher Bearers to have a suitable origin to start from when determining the desired level of baseline computational difficulty of VBA generation on the local network. Both PBKDF2 and Scrypt share a similar baseline relationship with the iterations count used: about every 20-25 thousand additional required iterations results in 5 more milliseconds of computation time during address generation and verification.

These two KDFs intentionally follow a similar progression with their minimal difficulties because PBKDF2 is a CPU-bound KDF while Scrypt is a memory-bound KDF. Keeping these two minimal baselines close will allow implementations to choose from a similar baseline difficulty for each type (memory-bound or CPU-bound), and to make their further determinations from that pattern. It should be noted that the Scrypt KDF's linearity slightly tapers into a gentle curve at higher iterations count values. The reason for this fall-off is unclear but it does little to affect any projections of minimal baseline computation times.

Next, the same laptop was used to evaluate the individual relationships between iteration counts at an arbitrarily high difficulty for each KDF. Results from each Figure should not be used to compare one algorithm to another, as each algorithm's 'high' difficulty setting was chosen independently from the others. Instead, the results should serve to show how much time a considerably high cost for each algorithm might require during generation and verification processes, even on an average laptop, in relation to the individual, node-selected iterations counts used.

The input iterations count selected during each VBA generation can widely change the aggregate cost of securing an address on the network, whether advantageous or not. Importantly, selecting the iterations count is in the control of the VBA-generating node. Neighbors who wish to ensure the

 doi:10.20944/preprints202406.1561.v1

legitimacy of a received LL2IP binding will be expending the same, symmetric amount of time and energy to verify the binding.

4.2. Difficult PBKDF2_SHA256

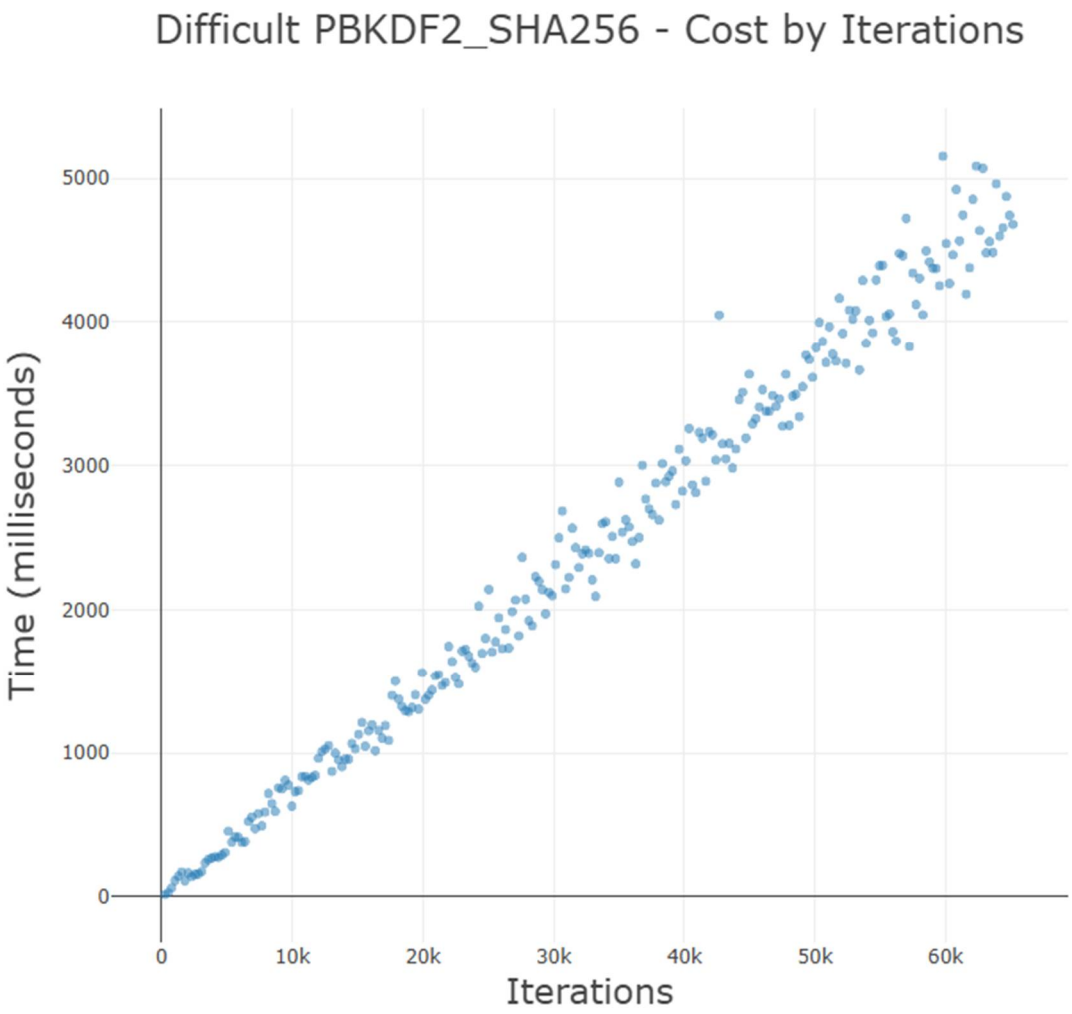


Figure 11. A high difficulty setting with the PBKDF2_SHA256 algorithm (an ITERATIONS_FACTOR of 256) shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increase.

4.3. Difficult Argon2

 doi:10.20944/preprints202406.1561.v1

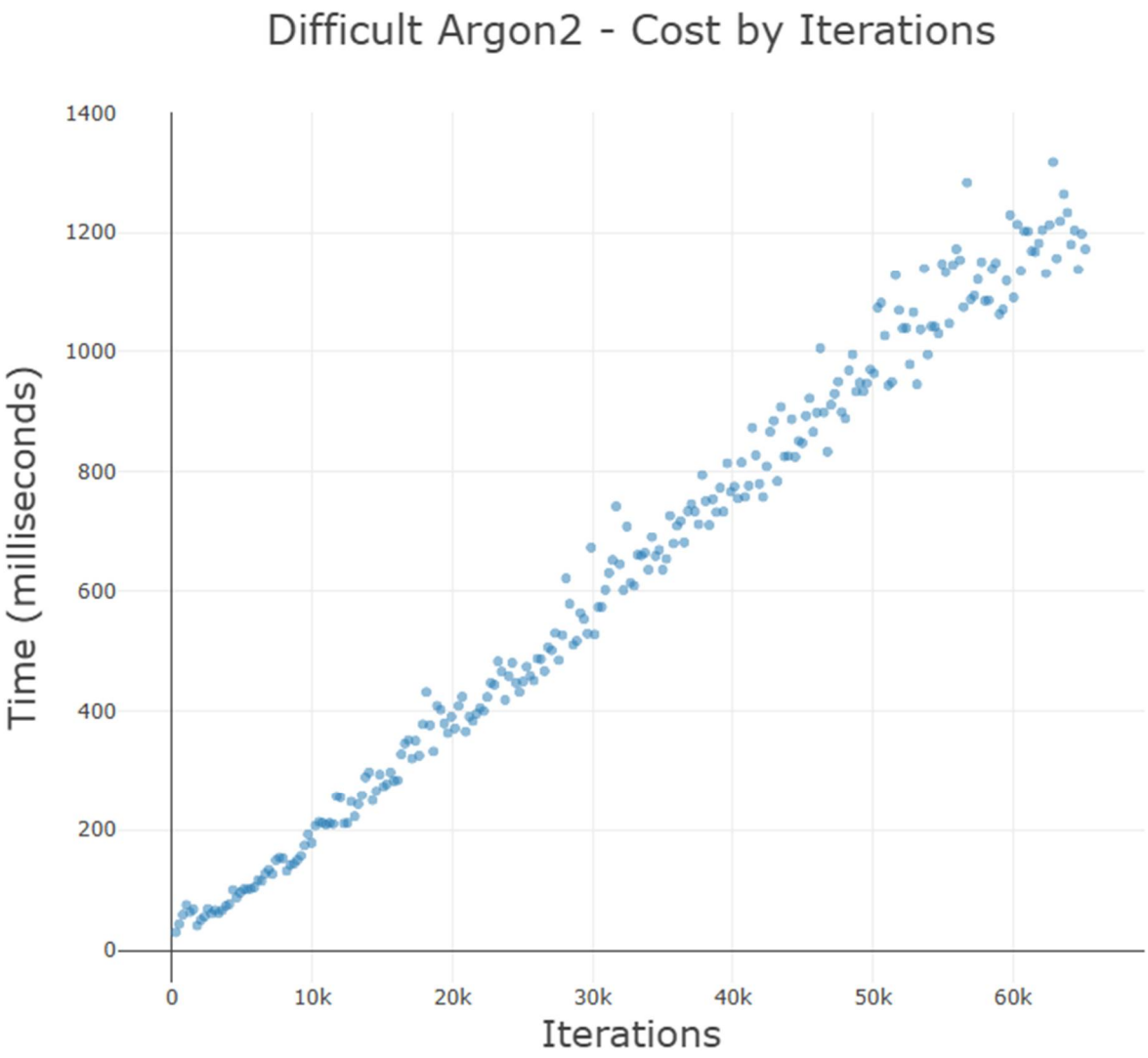


Figure 12. A high difficulty setting with the Argon2 KDF (more specifically, Argon2d; with a Parallelism of 32 and a MemorySize of 2,048) shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increase. The scale of the Time axis is much smaller than the other KDFs used in this experiment.

4.4. Difficult Scrypt

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1561.v1

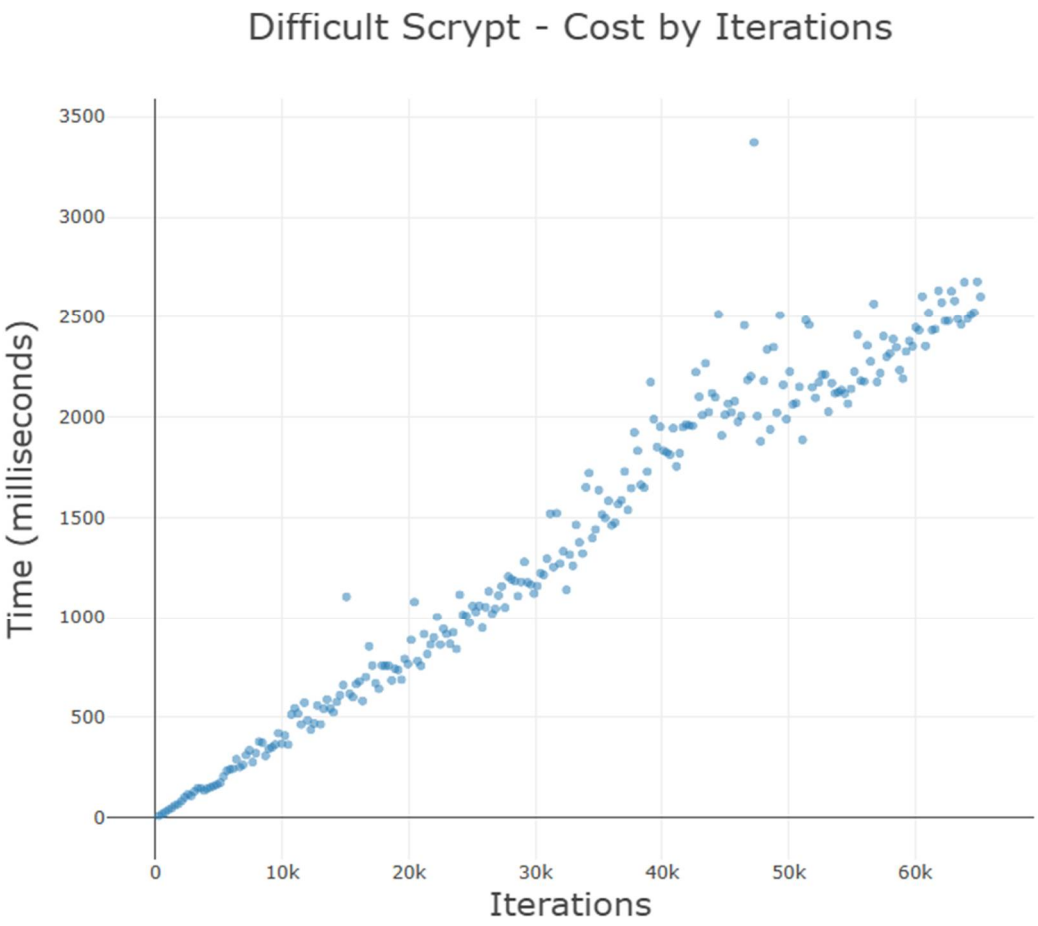


Figure 13. A high difficulty setting with the Scrypt KDF (a SCALING_FACTOR at its maximum of 5) shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increases and the linearity of the graph gently curves downwards.

5. Discussion

5.1. Precomputing Address Collisions

Malicious Voucher Bearers controlling address generation parameters are not able to further any goals aimed at neighbor spoofing attacks without infeasible time-memory tradeoffs. Control over network-wide KDF parameters and the voucher seed affords the opportunity to minimize the baseline difficulty of computing a hash collision, but only to an extent where the correct collision-producing link-layer identifiers must yet be discovered for neighbors with dynamic addresses.

If the role of Voucher Bearer is successfully hijacked by a malicious neighbor, then a static seed value with a minimal-cost KDF can be set according to the attacker's whims. The attacker is able to pre-compute a set of rainbow tables for all possible link-layer address bindings. However, selection of an iterations count is determined at each network node, and is not controllable by the threat actor. Therefore, indexing a set of predetermined results would require a repository of knowledge containing derivations from all possible link-layer identifiers (48 bits for MAC addresses), multiplied by the possible iteration values (a 16-bit value) for each generated network address.

$$2^{48} * 2^{16} = 2^{64} = 18{,}446{,}744{,}073{,}709{,}552{,}000\ hashes$$

In the case specific to MAC addresses as link-layer identifiers: if each result stored only the necessary 48 bits extracted from the derived KDF hash value, then storage requirements per hash lookup in the rainbow table would total:

MAC Address (48 bits)
+ Iterations Count (16 bits)
+ Hash Result Slice (48 bits)
= 112 bits or 14 bytes

 doi:10.20944/preprints202406.1561.v1

For 2^{64} rainbow table entries at 14 bytes each, 258,254,417,031,933,722,624 bytes, or about 224 EiB (exbibytes), of storage space would be required. 224 EiB is roughly equivalent to 2 million 128-TiB enterprise-grade storage arrays in sequence: an incomprehensible amount of data. If each hash were to require only 1 microsecond to compute, 8183.589 millennia of compute time would be required to calculate all possible values, longer than any conceivable amount of time on a human scale.

For more perspective, in an evenly distributed workload, 100,000 nodes would need to operate at full throttle for almost 82 years to generate all of the desired information. Finally, this data would need to be readily cross-referenceable because finding a collision necessitates using a different input link-layer address than the one used by the legitimate node, by principle of link-layer address uniqueness on-link. All of these preemptive calculations assume the subnet prefix is also statically precomputed with the typical link-local prefix value of FE80::/64. Any change of the subnet prefix value requires an entirely new set of data at 224 EiB in size.

These attacks are clearly computationally infeasible. In the best interest of future-proofing VBA, and to avoid any absurdities related to these considerations, all deployments should strongly consider using Router Advertisement Guarding per [23] to assist in preventing these attacks. Guarding against rogue vouchers disallows any and all hijacking and abates these concerns, because the voucher seed will never be a fixed value.

5.2. Other Security Considerations

Networks requiring a mix of ephemeral addresses in parallel with static, stable addresses will encounter difficulties with VBA. Preserving the state of a voucher long-term will not be a feasible strategy to maintain stable addresses, since its preservation for any extended period grants more time for threat actors to compute collisions. Assigning static addresses to nodes in a VBA-enabled network can be accomplished using a couple approaches:

- Use the AGVL IEM on either all interfaces within the local network, or on interfaces known to interact with the target static address(es) directly. The AGVL IEM will permit per-implementation behaviors to strongly prefer Secured results of NDAR exchanges over Unsecured ones. This option will remove any guarantees of address ownership or on-path attack prevention from the static address(es), because a static address failing the VBA verification process will be tagged in the Neighbor Cache as an Unsecured entry, at the same level of preference and security as other addresses whose bindings fail to verify.
- If neighbors do not interact with the static address(es), then the only affected parties are the node(s) with static assignments and the subnet gateway, which will likely route traffic to and from the static address(es). If this is the case within the subnet, then only host-to-router NDAR transactions will fail verification, so a static entry in the NC of the router should correlate each LLID to each static IP address expected to use the gateway.

An additional concern specific to IPv6 networks is anycast addressing. Anycast addresses are allocated from the unicast address space and are thus indistinguishable to nodes establishing connections to them. NDAR exchanges with these targets may therefore respond with varying LLIDs and cause VBA verification to be unreliable. For this reason, it is not recommended to utilize anycast addresses, because the ownership of the address cannot be bound to any particular LLID. The IPv6 Addressing Architecture specification (RFC 4291 [24]) outlines a Required Anycast Address in Section 2.6.1. VBA maintains compatibility with this requirement by disabling address verification for per-prefix subnet anycast addresses. For example, a host using SLAAC to generate an address in the subnet 2001:db8:700::/64 will disable VBA verification for the address 2001:db8:700::.

Lastly, considering the constraints of VBA verification, it is possible for a sending node N_A to have verified the VBA for some neighbor N_B without the reverse being true, if N_A.had not provided an SLLAO during the initial NDAR exchange. N_A can send packets to an application or service on N_B without requiring any response traffic in return. Nodes receiving unsolicited packets from neighbors, for which no response is required or demanded by the sender, do not need to verify the sender's address binding. The receiving node may choose to verify the neighbor's IP address if enforced by a VBA implementation, but it is not required. VBA is specifically designed for the prevention of

 doi:10.20944/preprints202406.1561.v1

neighbor spoofing attacks and is not concerned with policing incoming traffic that does not require an NDAR exchange.

5.3. Built-in Transition Mechanisms

It is unrealistic to assume that VBA would be deployed simultaneously across all nodes in a local network, because not every active node will receive compatibility at the same time. It is safe to assume based on past adoption trends that most devices will never achieve operable VBA support. Therefore, VBA comes predefined with an ability to operate in an intermediate environment where its full support is lacking. The three factors driving this ability are (1) Interface Enforcement Mode options for each participating interface, (2) localized changes to NDP occurring mostly in the software logic and not to the protocol itself, and (3) processes that do not require complex interactions between neighbors.

A pure IPv6 local network using the AGV IEM across its nodes will simply not be able to communicate bidirectionally with any node(s) lacking VBA support. For example, bidirectional traffic between a non-VBA node with dynamic addresses and an AGV IEM network gateway will be dropped at the gateway due to its binding verification requirement. In the case of dual-stack local networks, IPv4 traffic can be used as an insecure failsafe protocol when connecting nodes are explicitly aware of a route in both protocol stacks, such as between a host and a gateway router. The Happy Eyeballs algorithm from RFC 8305 [25] specifies a connection methodology that simultaneously attempts IPv4 and IPv6 connections, preferring IPv6 communication where possible. For local networks using AGV mode, the IPv6 network will appear unavailable and broken to unsupporting node(s). Thus they might desirably fall back to using available IPv4 connections instead. This strategy will permit a degree of communication with non-VBA nodes wherever IPv4 traffic is allowed.

Local IEMs on nodes communicating directly with incompatible neighbors can be adjusted to better accommodate the lack of verifiable bindings. For example, a VBA-enabled node corresponding with a neighbor running an antiquated networking stack might opt to use the AGVL IEM. Doing so would allow the VBA node to strongly prefer Secured devices for the rest of the network, such as the default gateway, while still accepting Unsecured NDAR traffic that does not contain any superseding Secured responses. In the case of a subnet router in a network with mixed VBA support, using the AGVL IEM can again prove very advantageous for the sake of accommodation. Assuming most nodes use VBAs and a few cannot, only those few nodes will remain at risk of neighbor spoofing attacks.

 doi:10.20944/preprints202406.1561.v1

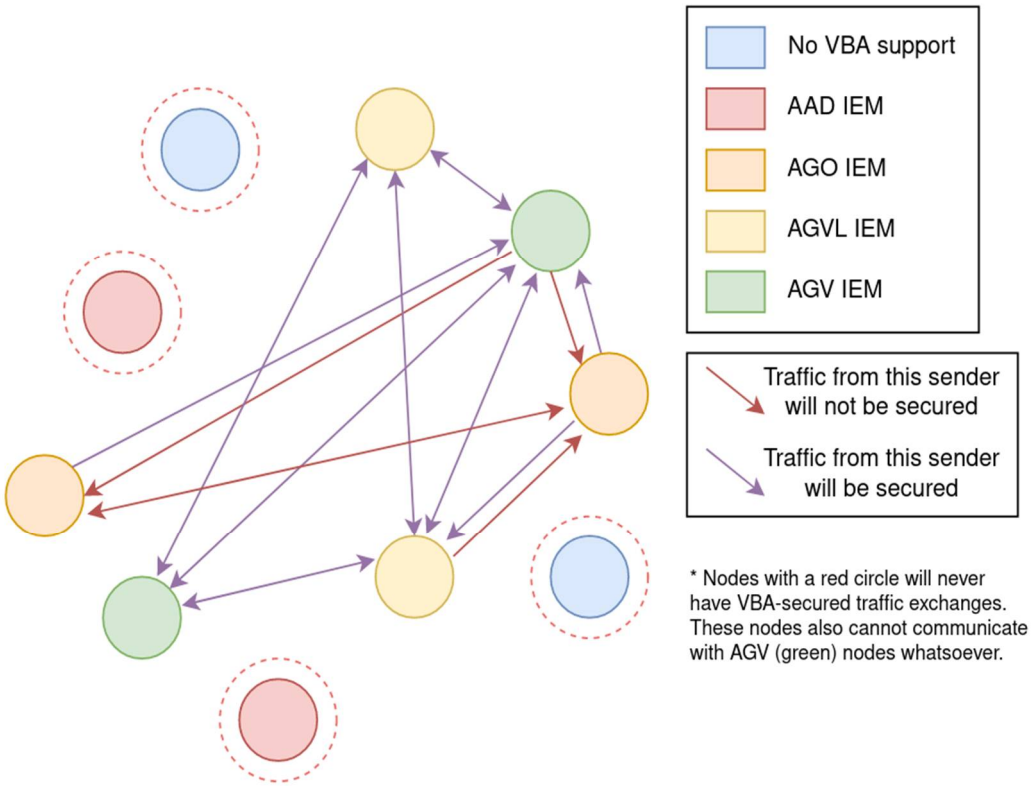


Figure 14. A mixed local network is shown where a single valid Link Voucher is delegated to VBA-capable neighbors. Neighbors without VBA capabilities are shown in blue, VBA-aware neighbors are shown in red, orange, yellow, and green for the IEMs AAD, AGO, AGVL, and AGV respectively. Different links are shown between some hosts to indicate their connectivity and security within this transitioning network. Traffic inbound to verifying nodes is generally considered secured.

5.4. Examining the Threat Model

The introduction of VBA satisfies the concerns listed in the original threat model from Section III. VBA relies on the principle of LLID uniqueness on the same broadcast domain, and thus threat actors cannot subversively spoof another node's legitimate LLID without introducing obvious network disruptions. Since all deterministic VBA generations depend upon the node's supposed LLID given during NDAR transactions, two nodes who cannot share the same LLID will never be successfully verified by neighbors for the same IP address. Whether or not the link layer is secure, VBAs are still necessary to validate bindings between IP and link-layer address components during address resolution.

Likewise, threats from external, off-link nodes are mitigated by VBA because IP addresses are the result of hashing algorithms, which generally produce pseudo-random outputs. External nodes will not be aware of the voucher details incorporated into the final address: the stored state required for address generation consists of local-only information, so the threat is abated. VBAs can also be rotated to an entirely new, valid address by changing the work factor value (i.e., iterations count) embedded within the address, even if no other parameters or voucher information has changed.

5.5. Simplicity, Privacy, & Flexibility

As the original goals of this research stated, the introduction of Voucher-Based Addressing aims to maintain three core ideals: simplicity, privacy, and flexibility. In this brief section, the work of this research will be fairly evaluated against these ideals, in order to determine its adoption potential. A justification is fairly provided to explain why simplicity, privacy, and flexibility are indeed goals achieved by VBA in practice.

Simplicity is the key to drawing audience attention to a protocol or idea; complexity can be understandably intimidating. Solutions of the past which have found themselves too complex or

 doi:10.20944/preprints202406.1561.v1

difficult to implement have been buried by the sands of time and relegated to a lifecycle filled with only academic citations and no concrete manifestations. VBA is a simple scheme because it does not require huge, mandatory impacts to NDP that cause disruptions in the network or at the node-to-node level. With a mixture of Interface Enforcement Mode selections, a broadcasted Link Voucher option, and already-present details about a node, addresses can be generated and verified using a procedure that is easy to follow and implement. The address verification shim process is a small snippet of software that carries the entire weight of VBA. It is modifiable with one simple per-interface setting (the IEM).

Privacy is important to establish properly and consciously in proposals that determine the unicast generation of interface IP addresses, whether local or global scope. It is also an afterthought of many proposals and research which might be elegant and practical otherwise. VBA manifests privacy in its ability to generate disposable, deterministic, outwardly pseudo-random addresses while still providing enough information for on-link neighbors to reconstruct and validate them. VBAs have no flags or other magic values which indicate that they are VBAs, adding to their obscurity.

The VBAs assigned to a local interface stem from a one-to-many pairing of the interface LLID to its assigned output address(es) coupled with a set of node-selected iterations counts. This means a node is free at any time, even if the voucher parameters have not changed, to choose a new interface identifier which shares no correlation to another chosen iterations count, defeating any address tracking or activity correlation concerns between addresses. VBAs originating from the same input parameters with varying iterations counts cannot be used to determine the internal state of the VBA algorithm (and thus the details of the interface's link-layer address or active voucher), because a hashing algorithm generates the majority of the final address suffix. VBAs can be regenerated for each subnet prefix, thus location tracking is not possible. Additionally, by the irreversible nature of hashing functions, VBAs will not expose the underlying LLID of the generating interface, thus alleviating device-specific vulnerability exploitation concerns and other privacy concerns from leaked LLID values.

Finally, flexibility is a paramount concern for researchers seeking adoption of their proposals. Any specification that mandates an immediate, wholesale, strict adherence to itself is bound to fail. Networks are built to be dynamic, independent, and transitionable between various protocols and specifications, and any particular node may be at its own stage in deployment of a software or specification. Any proposal which ignores a simple transition capability will never be adopted because forced compatibility violates any notion of flexibility. VBA employs IEMs to fulfill this need. The various interface modes allow an operating neighbor to decide which policies to enforce based on local settings or autoconfiguration alone, enabling per-interface choices about which rules to adhere to.

6. Summary, Future Research, & Conclusion

There is no tangible velocity towards the adoption of a robust security solution for Neighbor Discovery's issues. It seems as though these security weaknesses have long been covered and patched with a cocktail of willful ignorance and idle hands. SEND and CGA have proven over the course of decades to be undesirable and too complicated or esoteric for most practical applications, in a realm such as IPv6 that remains fearsomely alien to laymen. Network monitoring solutions require consistent involvement and interference by administrators who ultimately 'just want it to work' so they can have peace of mind. Other proposals to fix NDP's weaknesses have either languished in obscurity, are too complex, or fail to focus specifically on their practicalities and acknowledge their own shortcomings.

The real fix to these weaknesses is to create and subsequently adopt effective NDP security solutions which provide a trinity of attributes: simplicity, privacy, and flexibility. A solution lacking one or more of these three properties will not be preserved in the long-term without painstaking effort, as time has already proven. Voucher-Based Addressing seeks to establish itself as a proposal touting these attributes and aiming to solve a specific problem, rather than ambitiously trying to remediate all complex issues of the protocol at once.

In summary, VBAs bind input link-layer addresses to sets of privacy-focused, deterministic output addresses that can be reconstructed by verifying neighbors to confirm a reported binding.

 doi:10.20944/preprints202406.1561.v1

Address generation and verification is constrained to some initial conditions set by a shared voucher on the local network. Generated addresses appear outwardly random to off-link nodes and are easily rotatable to other seemingly random values; yet on-link they provide all information required for voucher-aware neighbors to verify them. Neighbor consensus of voucher parameters acts as an enforceable handshake during the address resolution process. This process is not actively falsifiable without introducing obvious network disruptions, thus preventing threat actors from subversively intercepting or modifying traffic between neighbors in on-path attacks.

VBA is a simple, unique, transparent, privacy-conscious, and flexible standard for use in Neighbor Discovery Protocol transactions. The proposal presented in this research maintains a high level of abstraction and does not beget very much concrete and practical evidence of its practicality. There is still, however, a very pressing need for a low-configuration, low-complexity technology to fill the security void left by the absence of SEND and CGA adoption in the modern enterprise.

The intent of this research within the wider internet community is not to declare an ultimate solution to the neighbor spoofing issues of NDP, though it may perhaps be a suitable baseline. Rather, it is to draw attention to a longstanding security issue and to spur thought for future research to build upon. Future developments based upon concepts herein might continue to focus on how they could be refined or better implemented, which vulnerabilities they introduce, any gaps in their theory or practice, and how the cross-application of each technique could be applied otherwise. Researchers are encouraged to explore more modern, alternative approaches to securing the NDP Address Resolution process from neighbor spoofing threats.

To conclude, the broad goal of introducing VBA is to define an alternative to SEND, CGAs, and other monolithic approaches to solving the neighbor spoofing problem. This research aims to pragmatically harmonize three important attributes: privacy, flexibility, and simplicity. It has set out to molt the Neighbor Discovery security paradigm that has continued to depend upon sophisticated Public Key Infrastructure, limiting infrastructure-only protocols, superfluous new data constructs, unwieldy asymmetric cryptography, and centralized address registration authorities. This research has defined a decentralized and empirical approach that mostly evades the aforementioned tar pits and creates a new perspective. The problem of neighbor spoofing in IPv6 can indeed be solved in a way that is comprehensible, privacy-focused, wholly transparent to end users, and not at all disruptive to incompatible neighbors.

Supplementary Materials: The original implementation used in the study is openly available in a public repository at https://github.com/NotsoanoNimus/cpp-fiddles under the ‘vba-tests’ and ‘vba-collision-tests’ folders.

Author Contributions: Conceptualization, Z.P.; Investigation, Z.P.; Methodology, Z.P.; Project administration, J.G. and Z.P.; Software, Z.P.; Supervision, J.G. and Z.P.; Validation, J.G. and Z.P.; Writing - original draft, Z.P.; Writing - review, J.G. and Z.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not available as no new data were created as a result of this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. W. A. Simpson and E. Nordmark, “Neighbor Discovery for IP Version 6 (IPv6),” no. 1970. in Request for Comments. RFC Editor, Aug. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1970
2. D. T. Narten, W. A. Simpson, and E. Nordmark, “Neighbor Discovery for IP Version 6 (IPv6),” no. 2461. in Request for Comments. RFC Editor, Dec. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2461
3. W. A. Simpson, D. T. Narten, E. Nordmark, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” no. 4861. in Request for Comments. RFC Editor, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4861
4. M. Gupta and A. Conta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” no. 4443. in Request for Comments. RFC Editor, Mar. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4443
5. J. Kempf, J. Arkko, B. Zill, and P. Nikander, “SEcure Neighbor Discovery (SEND),” no. 3971. in Request for Comments. RFC Editor, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3971

 doi:10.20944/preprints202406.1561.v1

6. T. Aura, "Cryptographically Generated Addresses (CGA)," no. 3972. in Request for Comments. RFC Editor, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc3972
7. P. Sumathi, S. Patel, and A. Prabhakaran, "A Survey on IPv6 Secure Link Local Communication Models, Techniques and Tools," 2017.
8. P. Sumathi, S. Patel, and Prabhakaran, "Secure Neighbor Discovery (SEND) Protocol challenges and approaches," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6. doi: 10.1109/ISCO.2016.7726976.
9. J. Kempf, P. Nikander, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," no. 3756. in Request for Comments. RFC Editor, May 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3756
10. J. Arkko, T. Aura, J. Kempf, V.-M. Mäntylä, P. Nikander, and M. Roe, "Securing IPv6 neighbor and router discovery," *WiSE 02 Proc. 1st ACM Workshop Wirel. Secur.*, Sep. 2002, doi: 10.1145/570681.570690.
11. M. Anbar, R. Abdullah, R. M. A. Saad, E. Alomari, and S. Alsaleem, "Review of Security Vulnerabilities in the IPv6 Neighbor Discovery Protocol," in *Information Science and Applications (ICISA) 2016*, K. J. Kim and N. Joukov, Eds., Singapore: Springer Singapore, 2016, pp. 603–612.
12. D. T. Narten, T. Jinmei, and D. S. Thomson, "IPv6 Stateless Address Autoconfiguration," no. 4862. in Request for Comments. RFC Editor, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4862
13. J. D. Day and H. Zimmermann, "The OSI reference model," *Proc. IEEE*, vol. 71, no. 12, pp. 1334–1340, 1983, doi: 10.1109/PROC.1983.12775.
14. K. Moriarty, B. Kaliski, and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1," no. 8018. in Request for Comments. RFC Editor, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8018
15. T. Kiravuo, M. Sarela, and J. Manner, "A survey of Ethernet LAN Security," *IEEE Commun. Surv. TutorialsIEEE Commun. Surv. Tutor.*, vol. 15, no. 3, pp. 1477–1491, Jan. 2013, doi: 10.1109/surv.2012.121112.00190.
16. F. Gont, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)," no. 7217. in Request for Comments. RFC Editor, Apr. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7217
17. D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001, doi: 10.1007/s102070100002.
18. T. Polk, R. Housley, S. Turner, D. R. L. Brown, and K. Yiu, "Elliptic Curve Cryptography Subject Public Key Information," no. 5480. in Request for Comments. RFC Editor, Mar. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5480
19. D. R. L. Brown, T. Polk, S. Santesson, K. Moriarty, and Q. Dang, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA," no. 5758. in Request for Comments. RFC Editor, Jan. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5758
20. R. Housley, T. Polk, and L. E. B. III, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," no. 3279. in Request for Comments. RFC Editor, May 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3279
21. A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications," no. 9106. in Request for Comments. RFC Editor, Sep. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9106
22. C. Percival and S. Josefsson, "The scrypt Password-Based Key Derivation Function," no. 7914. in Request for Comments. RFC Editor, Aug. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7914
23. G. V. de Velde, J. Mohácsi, E. Levy-Abegnoli, and C. Popoviciu, "IPv6 Router Advertisement Guard," no. 6105. in Request for Comments. RFC Editor, Feb. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6105
24. D. S. E. Deering and B. Hinden, "IP Version 6 Addressing Architecture," no. 4291. in Request for Comments. RFC Editor, Feb. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4291
25. D. Schinazi and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency," no. 8305. in Request for Comments. RFC Editor, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8305