Voucher-Based Addressing & Sessions: A Simple, Flexible, Privacy-Preserving Recipe for Mitigating IPv6 Neighbor Discovery Redirection Attacks

by

Zachary T. Puhl

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Computer and Information Science)
in the University of Michigan-Dearborn
2024

Master's Thesis Committee:

Associate Professor Jinhua Guo, Chair
Professor Di Ma
Assistant Professor Birhanu Eshete

Zachary T. Puhl

zpuhl@umich.edu
zpuhl@xmit.xyz

ORCID iD: 0009-0001-6955-8104

Dedication

Dedicated to everyone in my life who each graciously lent me their time, support, attention, and insights.

I would like to distribute my many thanks to Dr. Jinhua Guo at the University of Michigan-Dearborn for his support, direction, oversight, and patience in trudging with me through this endeavor; to all committee members for their expertise and gracious participation in evaluating my work; to my family and friends for forgiving my long weeks away, my frenetic whiteboard sessions, and my lack of responsive communication during the time I wrote this paper; and to my wife Sandra, for always dearly supporting me, unhesitantly affording me her ear, and giving me the love I need to keep chasing my dreams.

Throughout all my ramblings, not without any one of you would this be complete.

Table of Contents

List of Figures

List of Appendices

Abstract

The vast majority of local IPv6 networks continue to remain insecure and vulnerable to neighbor spoofing attacks, a fearsomely practical attack vector formally described many years ago. The Secure Neighbor Discovery (SEND) standard and its concomitant Cryptographically Generated Addressing (CGA) scheme were introduced, revised, and adopted by large standards bodies to codify practical mitigations. Considering their poor adoption, much research since their acceptance has continued to find new perspectives and proffer new ideas. The orthodox solutions for securing Neighbor Discovery traffic have historically struggled to successfully harmonize three core ideals: simplicity, flexibility, and privacy preservation. This research introduces an alternative to IPv6 address generation methods that secures the Neighbor Discovery address resolution process while remaining highly adaptable, indistinguishable, and privacy-focused. Applying a unique concoction of cryptographic key derivation functions, hash chaining, link-layer address binding, and neighbor consensus on the parameters of address generation, local address ownership is verifiable without the need for techniques that have hindered the adoption of the canonical specifications. Voucher-Based Addressing and end-to-end Neighbor Discovery Sessions are presented as synergistic, low-configuration, low-cost, and high-impact specifications for securing local networks against neighbor spoofing attacks.

Chapter 1 Introduction

Questionable ownership or authentication of addresses at the link layer of the Open Systems Interconnect (OSI) reference model [1] is a problem with solutions both sought and offered by many research efforts. Lacking proof of ownership and identity at this foundational layer affects the integrity, confidentiality, and overall trust of protocols at higher layers of the network stack. The formation of IEEE 802 Medium Access Control (MAC) addresses–by far the most commonly used link-layer address specification–originally planned that each identifier should uniquely distinguish network interfaces at a global scope. Each address was to be "burned in" and unchangeable for each interface. As the conception of MAC addresses became more removed, progressive technologies permitted burned-in addresses to be arbitrarily modified by operating system software and set directly by end users.

Randomized and Changing MAC addresses (RCM) is a more recent MAC address assignment methodology supported by some software vendors to preserve device privacy by wholly and intentionally abandoning any assumptions of MAC address stability and consistency [2]. This is outlined well by the ongoing efforts of the MADINAS IETF Working Group to catalog the implications of RCM at a broad scope [3]. By introducing regular rotations of device MAC addresses and terminating any notions of global address uniqueness, validating device ownership of a purported address now known to be temporary is elevated to an even greater importance. The same importance applies for on-the-fly MAC address modifications by operating systems. All authentication constructs underpinned by MAC address stability must

therefore be reshaped and reconsidered due to this decoupling of interface identities from their link-layer identifiers.

To embrace this evolving climate, higher-level network protocols riding on an insecure link layer should opt to utilize some other verification means that can ensure fellow correspondents on the same link layer have not swapped or been impersonated. Attackers changing or intercepting data without authorization by spoofing link-layer identifiers presents a key and fundamental problem in Neighbor Discovery Protocol (NDP) for IPv6 networking implementations. Cryptographically Generated Addresses (CGAs) [4] have long stood as the solution that could guarantee authentication for an NDP that is by default insecure. These constructs leverage asymmetric cryptography and digital signatures to validate and affirm that an originator of a packet (i.e., the address owner) retains active knowledge of a secret cryptographic key. Proving knowledge of such a key authenticates a peer and voids any requirement of asserting link-layer address ownership. Asserting such signatures and proofs of identity with the CGA scheme is done during the NDP Address Resolution (NDAR) process, as augmented by the later Secure Neighbor Discovery (SEND) [5], a suite of augmentations to NDP aiming to protect the protocol from common attack vectors.

The SEND and CGA specifications were also formulated to address concerns of other NDP weaknesses, many of which the updated NDP protocol specification is self-aware of. One relevant and focal weakness of NDP is the potential for malicious redirection attacks, whereby a malevolent neighbor can insert itself into the path of traffic between two on-link neighbors by intercepting and falsifying NDAR responses and in some cases by successfully spoofing link-layer identifiers. The category of attack vectors exploited by NDP 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 the local scope, where the least amount of network traffic is likely to be encrypted or otherwise secured.

The capability to mitigate issues of link-layer address ownership and on-path attacks with SEND and CGAs has been well known for some time now, but these mitigations have never received widespread adoption, or even implementation, in practice [6]. The real reasons are ambiguous and unclear but are ostensibly associated with some combination of (1) insufficient awareness of the standards, (2) the sophistication of implementation, and (3) protocol inflexibility and disparate inefficiencies. Considering the arcaneness and obscurity of IPv6 outside of academia, networking enthusiast communities, or IETF circles, SEND and CGA reach another level of pinpointed knowledge that is not dampened whatsoever by their complexity and lack of adaptability with baseline NDP implementations. Furthermore, existing uses of SEND and CGA already represent wildly varying effects on networking performance across different systems [7], in implementations which have no sufficient backing or validation [8].

Keeping in mind the mistakes of the past and the paths already paved, a solution should be proposed for NDP that provides protection against on-path attackers while also remaining viable, flexible, and efficient for all adopting devices. Such a solution necessitates balancing three key aspects in its conception: privacy, flexibility, and simplicity. This research presents two new synergistic yet fully decoupled standards to fulfill this need: Neighbor Discovery Session Options (NDSOs) and Voucher-Based Addressing (VBA; also, Voucher-Based Addresses, VBAs). The adoption and transition to one proposed standard does not imply or require the implementation of the other, but is somewhat like SEND and CGA: their synergies are well defined in cases where they might be deployed together, as intended by their designs. Both

standards seek to create low-complexity, high-impact, optional protocol addenda to secure NDP via alternative authentication techniques; each aiming to resolve a different problem while maintaining a focus on user privacy. NDSOs validate and substantiate proof of identity at the link layer, while VBAs prevent subversive on-path attacks at the network layer.

Unlike much security research of the past, this work does not intend to recycle the applications of asymmetric cryptography or central registration authorities to NDP. A key purpose of this research is to theorize and produce alternative choices to the widely accepted SEND and CGA standards–rather than augment, replace, or survey them–to foster new perspectives or to renew interest in solving these longstanding IPv6 security issues in local networks. Starting with Chapter 2, 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. Conceptual overviews for VBAs and NDSOs are given in Chapters 3 and 4 respectively. Relevant benchmarks and other experimental, implementation-based results are then described in Chapter 5, with a subsequent discussion of some of the miscellaneous practicalities and observations of the outcomes of this research in Chapter 6. Chapter 7 then concludes the work by outlining some potential future research and summarizing the implications of this research's core ideas.

Chapter 2 Background & Related Works

2.1 The Broad Scope of Neighbor Discovery

The Neighbor Discovery Protocol (NDP) for Internet Protocol version 6 (IPv6) was first introduced in 1996 [9], revised in 1998 [10], and published in its current version in 2007 [11]. It is a protocol extension of ICMPv6 [12] used by neighboring nodes on a local area network to discover each other's presence, to detect routers, to self-determine addresses, to resolve each other's link-layer addresses, and to maintain details about the reachability of–and paths to–known, active neighbors. Introduced with it were many of the distinct, familiar terms and constructs often associated with IPv6 networks: nodes, hosts, interfaces, various caches, various built-in protocol options, et cetera. NDP is still ubiquitous in present day networks, providing what is the primary focus of this research: NDP Address Resolution (NDAR), the IPv6 analog of the IPv4 Address Resolution Protocol (ARP). NDAR is the essential functionality of resolving peer IP addresses to their corresponding on-link link-layer addresses (Section 7.2 of [11]), allowing link-layer frames to be forwarded directly to the appropriate destination on-link.

Myriad functional requirements of all IPv6 networks are satisfied by Neighbor Discovery; it is an extensible and all-encompassing protocol gluing together link-layer traffic with network-layer addressing. Firstly, the NDAR process is accomplished through the use of Neighbor Solicitation (NS) and Neighbor Advertisement (NA) packets with various attached options. Next, once the NDAR process is completed and nodes have learned each other's resolved link-layer addresses for the two communicating IP addresses, cache entries are updated and maintained at each node. These stored bindings are ephemeral and must be regularly updated

by a process known as Neighbor Unreachability Detection (or NUD). This process occurs by means of either of two signals: any transaction occurring at upper-layer protocols which indicates forward connection progress (such as incrementing TCP sequence numbers), or a renewed exchange of NS and NA packets if no other higher-layer protocols are active between the two hosts.

With a focus on the node-to-node scope, each node must be able to self-assign its own address(es) before beginning to discover its neighbors and other local network terrain. Ideally, this process of self-assignment should occur independently of others on the local network, only checking to verify whether a chosen network-layer address is already being used. Stateless Address Autoconfiguration (SLAAC) is specified by RFC 4862 [13] to achieve this goal using NDP, requiring initializing nodes to automatically create their own link-local scope addresses [14] and check their uniqueness on the link with Duplicate Address Detection (DAD) [13]. Though there is an IPv6 analog to the more centralized Dynamic Host Configuration Protocol (DHCP), SLAAC is a mechanism unique to IPv6 (and thus Neighbor Discovery) environments and is often the preferred method of address assignment due to its independence from other communications.

Finally, NDP allows local nodes to discover routes to external networks and to learn a set of all active subnet prefixes on the link. Router Advertisement (RA) packets are either broadcast at an interval predetermined by the link routers, or directly in response to Router Solicitation (RS) packets from neighbors. Once traffic begins to flow to external networks, routers can also use Redirect messages to alert senders of a more suitable first-hop path to their particular destination address. RA messages might also carry prefix information that can allow listening nodes operating with SLAAC to self-assign more unicast addresses within these subnet prefixes,

and subsequently ensure the uniqueness of new addresses by use of the DAD process. Importantly, unicast addresses acquired by prefix delegation are typically globally routable in scope, while nodes themselves are free to use SLAAC and DAD to auto-configure link-local addresses as they see fit.

2.2 Security Concerns of Neighbor Discovery

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. Therefore, the widespread adoption of NDP occurred before its threat models were formally cataloged in RFC 3756 [15] and before the specification of Secure Neighbor Discovery (SEND) in RFC 3971 [16] with its complementary Cryptographically Generated Addresses in RFC 3972 [17]. Arkko et al. in 2002 [5] first detailed the various vulnerabilities of Neighbor Discovery: Neighbor spoofing, Router Advertisement spoofing, bogus prefix denial-of-service attacks, DAD attacks, and falsely advertised configurations. Many of these remain trivial to execute in normative networks as shown by Anbar et al. as recently as 2016 [18], 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 traffic redirection Man in The Middle (MITM) attack (also known as an ‘on-path’ attack), whereby a malicious neighbor can intercept traffic destined for another neighbor by falsifying NA or NS packet options. By successfully advertising a spoofed link-layer address binding with a victim IP

address, the attacker redirects frames from neighbors to itself, allowing it to inspect and capture private packet data on the forwarding path before forwarding it to its original target. This initial spoof and redirection is sometimes termed “cache poisoning” to express that each misled neighbor falsely caches the binding (i.e., association) between a neighbor IP address and a malicious link-layer address, rather than the genuine target’s link-layer address. If the attacker successfully spoofs the link-layer 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 is likely to be unencrypted the most often.

Mohamed Sid Ahmed et al. [19] helpfully categorize ND spoofing attacks into routing-based and non-routing-based attack vectors. Most routing-based on-path attacks also occur when an attacker successfully spoofs Router Advertisements, directing neighbors to forward packets through itself or through a bogus prefix. In this event, the attacker transparently arbitrates traffic between the victim and the legitimate network gateway in order to harvest unencrypted traffic, meta information, or inject its own malicious data into the traffic stream. The same attack can occur through the easy use of spoofed NDP Redirect messages, expressing the malicious node to be a ‘better first-hop path’ to external networks than the legitimate local network gateway. In both scenarios, the threat actor can dangerously observe and interact with all packets in-transit to–and often from–external networks.

In their analysis of Neighbor Discovery security and attacks, Najjar et al. [6] specify a few non-routing-based tactics used between nodes to forcefully expose the NDAR process to redirection attacks. NS or NA flooding attacks seek to exhaust the system resources of a victim

node with bogus messages, causing it to dump previous or stale cache entries to make room for incoming cache data. This can cause the previous cache entry for a target IP address to be pruned from the victim's cache, thus opening the capability for the attacker to insert a new poisoned entry. A similar attack is possible by forcing a timeout of the Neighbor Unreachability Detection process, therefore triggering the NDAR process to repeat and expose the cache of the victim to a spoofed address binding. While many traditional attacks behave opportunistically in this manner, there also exist other simpler cache poisoning tactics that merely set an Override option on unsolicited Neighbor Advertisements or advertise false link-layer address options in Neighbor Solicitations.

The 'classic' neighbor redirection 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 redirection. RFC 4861 [11] 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 a Neighbor Discovery proxy 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.

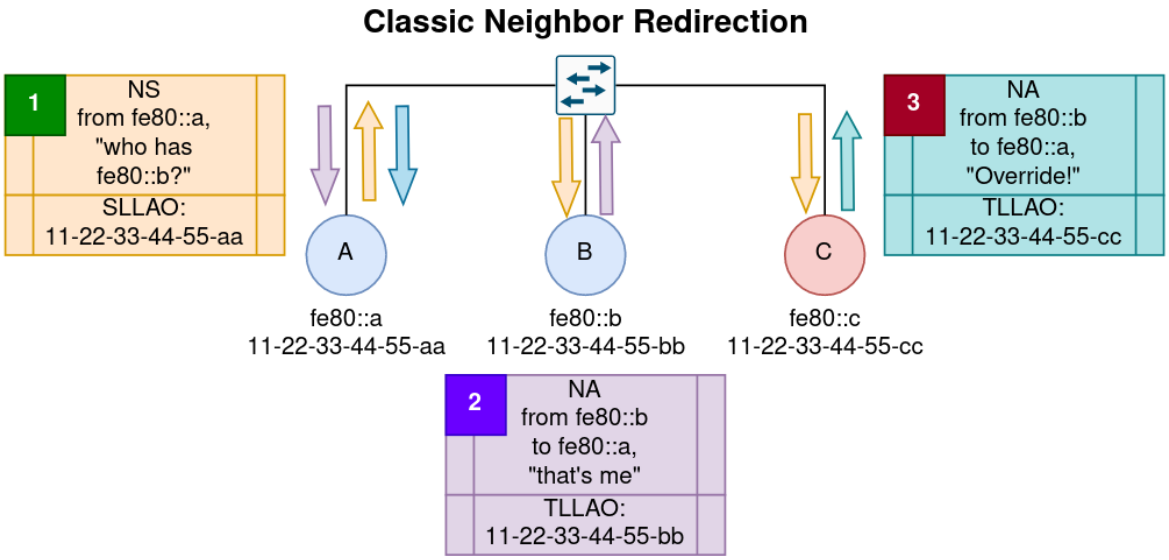


Figure 1. A classic Neighbor Discovery traffic redirection (on-path) attack. After the normal Address Resolution process is completed in steps 1 and 2, the listening malicious Node C sends a spoofed Neighbor Advertisement in step 3 to override the Link-Layer Address value in Node A's Neighbor Cache.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

In Figure 1, Node A is the solicitor who asks for the link-layer to IP address binding from the address fe80::b in step 1. Since Node A does not know the target's link-layer address yet (and thus where to forward frames), a solicited-node multicast group 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 Node A in step 1 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). In step 2, Node B receives the NS and 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. Node A is being 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 before forwarding it down the path to Node B where it will be received without knowledge that Node C had any interaction with it.

A less well-known and much more powerful cache poisoning attack opportunity exists in the optimizations for 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, so 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 [11] 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.

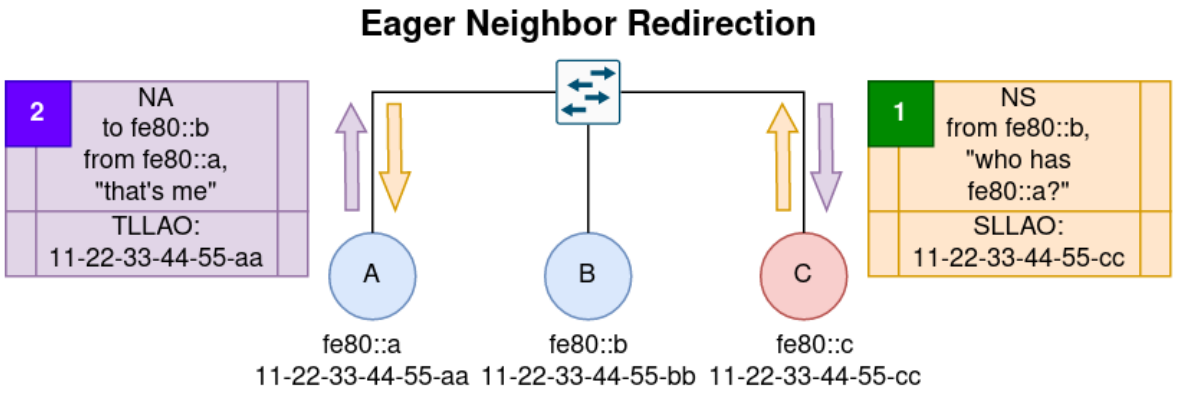


Figure 2. A more subversive approach to preemptively poisoning a neighbor's Neighbor Cache, without needing to wait for an NDAR transaction. Nodes receiving SLLAO stubs on Neighbor Solicitation packets are dictated by the NDP specification to accept them at face value for the sake of protocol optimization. So, Node A receiving the SLLAO can immediately respond with unicast messages to the apparent 'correct' link-layer address without needing reverse address resolution.

Figure 2 demonstrates the simplicity and potency of an Eager Neighbor Redirection 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. 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 for a 24-bit address suffix used to derive a solicited-node multicast address anyway. 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]. 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 by virtue of being the NDAR initiator (i.e., solicitor).

There exists a glaring commonality between all these various attack surfaces presenting an opportunity for on-path attacks with neighbor redirection: caching. Regardless of how the cached value is updated or exposed to poisoning, the simplicity of the attack relies solely on making a 'bad' update to the target's Neighbor Cache entries. 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. Much of the recent research regarding the Neighbor Redirection spoofing attack technique does not seem to place its aim at this common target.

2.3 SEND & CGAs: Promises & Problems

Secure Neighbor Discovery (SEND) was introduced by Arkko et. al in [5] as a conceptual security framework for ordinary NDP. It achieves much of its proposed security measures through the employment of Cryptographically Generated Addressing, a complementary idea that binds IPv6 addresses to public key values to assert an identity and to authenticate host-originating ND packets. After their inception, SEND and CGA were later formalized into two RFCs (RFC 3971 [16] and RFC 3972 [17], respectively) that are widely recognized as de facto, canonical solutions to the security pitfalls of the original Neighbor Discovery Protocol. SEND

and CGA have promised to deliver secure solutions to spoofing attacks, denial of service concerns, and abuses of the router discovery process; but, in the almost two decades since their formalization, have failed to come to fruition.

One of the original CGA works by Aura in [4] introduces CGAs as a means by which asymmetric keys can be associated with known IPv6 address values, using the spacious and typical 64-bit interface identifier afforded to most IPv6 addresses. Public keys are used as well-known, public identifiers that are associated with IP addresses via cryptographic hashing, and their corresponding private keys are used to create digital signatures by the node owning the address in order to assert its proof of address ownership. A CGA is constructed by generating a public-private keypair, appending it to the end of a selected 128-bit Modifier, 64-bit Subnet Prefix, and 8-bit Collision Count value, and hashing it. A 3-bit "Sec" parameter embedded in the resultant IP address dictates how costly the generation of the CGA must be. Based on the difficulty set by Sec, a certain hash must be generated from the four aforementioned parameters as a necessary proof of work. This expense was instituted in order to prevent brute-force enumeration attacks seeking to discover a matching hash value from a different public-private keypair. One key aspect of CGAs is the explicitly stated lack of a requirement for Public Key Infrastructure in order for the addresses to properly generate, communicate, and verify between neighbors.

CGAs have seen both great criticisms and great praise as the accepted solution of the Internet Engineering Task Force (IETF) for IPv6 address ownership. Many proposals have sought to improve their efficacy over time by various means: the creation of CGA++ by [20], privacy extensions and denial-of-service mitigations by [21], and revisions of the hashing and asymmetric cryptography being used by [22]. CGAs are generally not private, despite having a

pseudo-random structure. Alsadeh, Rafiee, and Christoph in [21] note that the preferred long-term use of CGAs in a network–due to their excessive costs to generate with proof-of-work techniques–results in privacy violations whereby users can be tracked online. Additionally, the CGA specification has no mention or recommendation of regular address rotations likely because of their cost in the first place. CGAs are also not simplistic: the original proof-of-work algorithm based on the Sec value scales at a factor of 2^{16}, making CGAs quickly expensive to produce and thus infeasible for embedded or mobile devices. In their research [22], Shah and Parvez surveyed the average time taken to compute CGAs and found the performance to be intolerable unless the default RSA algorithm was swapped with ECC instead.

Secure Neighbor Discovery leverages CGA to achieve its two primary goals of proof of address ownership and validation of the router discovery process, with an auxiliary goal of denial-of-service protection where feasible. The same private key used to authenticate a CGA is used in a new RSA Signature option with a concomitant CGA option to sign multiple different types of NDP messages and prove their validity. The Timestamp and Nonce options are introduced to the protocol as well to prove message freshness, mitigate replay attacks, and assert a challenge-response pattern for NS/NA exchanges. Router validation is also possible through both the use of Public Key Infrastructure and two new SEND-introduced ICMP message types carrying certification path information to hosts. SEND is the accepted mitigation for a significant portion of the security shortcomings of plain NDP because it incorporates a layer-independent approach to achieve its goals. By tying an asymmetric keypair to a network-layer address and not incorporating any link-layer information into addresses, SEND relies only on itself and any necessary certification paths that are required to validate routers as appropriate authorities. The SEND specification [16] is self-aware of its susceptibility to some of the same issues that exist in

the original NDP, chiefly denial of service attacks, but also those which are the result of an insecure link layer.

Both CGA and SEND, despite their benefits and criticisms alike, ultimately have not seen widespread adoption in the vast majority of IPv6 local networks. While SEND has a capability for differentiating Secured from Unsecured neighbors, in an effort of flexibility for adopting networks, finding neighbors bothered to properly implement the specification is cumbersome because of SEND's lack of mature support. Part of the issue with the protocols has been their requisite setup requirements and their overhead for network administrators. Public Key Infrastructure has proven daunting and undesirable, and most implementations of the two specifications are not mature or supported enough to provide a measurable security benefit. Furthermore, SEND performance is classically inefficient and ill-suited for the majority of IPv6 local networks that have no need for its heavy armor, and to many the cost is not worth the added protection.

Najjar et al. outline the five primary causes for the lack of SEND adoption in [6] as issues with compatibility, complexity, arcaneness (or lack of awareness), cost, and limited supporting implementations. Though SEND and CGA provide formidable and proven protection against Neighbor Discovery attacks–most importantly Neighbor Redirection threats–it is a solution which is intended to be comprehensive and thus is quite complex and disruptive. Privacy matters notwithstanding, SEND and CGA lack simplicity in their approach; understandably so for a specification so ambitious as to simultaneously solve most of NDP's security concerns.

2.4 Leveraging Monitoring & Infrastructure

Considering the complexities introduced by SEND and CGA, many network operators have chosen instead to use network monitoring tools that can proactively notify them of changes in link-layer address bindings for either all or a particular subset of network IP addresses. Monitoring is implemented as a more reactive technique to a problem which would ideally be solved proactively. Many proposed specifications to mitigate the multitude of NDP security issues also incorporate active changes in the infrastructure of local networks on which NDP is running. This is not inherently bad, so long as the benefit outweighs the cost and the involvement from administrators is also a low-cost addition.

Monitoring of link-layer to IP address bindings is not a new concept with the advent of NDP: a popular monitoring tool named “arpwatch” [23] existed well before the conception of Neighbor Discovery, and it is still widely used in enterprises today. The well-known NDPmon, a portmanteau of “NDP” and “monitor”, mirrors the functionality of arpwatch and extends its functionality into NDP for IPv6, to provide a way for administrators to know when link-layer bindings suddenly change. As an intriguing evolution, NDPmon was further developed upon by a sophisticated, machine-learning-based anomaly detection system in [24]. As mentioned above, the reactive nature of these systems is simultaneously their greatest advantage and disadvantage: event-driven solutions allow for real-time identification of threats, but usually also require real-time intervention to stop. Real-time threat identification systems are also more subjective and heuristic, having consequently many more false-positives. Generally, if an administrator is interested in preventing on-path attacks in their local IPv6 network(s), but they are perturbed by the mere thought of adding any static Neighbor Cache mappings, then it is a certain possibility that a monitoring solution such as NDPmon will be employed.

The use of configured or ‘smart’, NDP-aware infrastructure has also played a key role in advancing toward generally practical NDP security. For example, the Stateless Address Autoconfiguration (SLAAC) Attack Detection (SADetection) mechanism in [25] uses a centralized server infrastructure to reliably detect mischief in SLAAC processes on the local network. In perhaps the best use of infrastructure to bind together link-layer addresses and IPv6 addresses for interdependent validation, the SEND-SAVI framework in [26] has provided a way for trust chains to be established in binding link-layer addresses with IP addresses and a way to validate them using forwarding infrastructure. This research led to the subsequent, similar work by the IETF in RFCs 7219 [27] and 7039 [28]. Infrastructure-based solutions tend to be more practical since there is no reliance on node operating systems to maintain compatibility with new or updated NDP security mechanisms that may or may not be expected by capable neighbors.

2.5 Preserving User Privacy

Since NDP introduced SLAAC in RFC 4862 [13], the consented methodology used to auto-generate addresses has led to privacy concerns about user tracking, activity coordination across multiple sites, and targeted exploits based on known link-layer identifiers. Alongside the evolution of NDP security mechanisms, war has continued to rage about the best way to use SLAAC to self-assign interface IP addresses while preserving user privacy at each endpoint. Privacy is of course a more relevant concern for IPv6 than IPv4 due to the globally-routable nature of most IPv6 unicast addresses, rather than the masking performed by Network Address Translation in IPv4. The original IPv6 address generation mechanism named “EUI-64” formerly required interface identifiers to be derived from an extension of their 48-bit link-layer addresses (i.e., MAC addresses). This was subsequently deprecated for its obvious privacy concerns of

directly divulging the interface's link-layer address. RFC 8981 [29] discusses the generation of both stable and temporary privacy-focused SLAAC addresses. It slightly augments the generation process of opaque interface identifiers from RFC 7217 [30] and provides insight about more recent address generation methodologies.

Cryptographically Generated Addresses are subjected to privacy concerns because the cost associated with address generation (based on the Sec parameter of the address) are prohibitively expensive for regular address rotation intervals which would deter tracking. This ailment follows closely behind most security techniques seeking to utilize the IPv6 interface identifier address space (typically 64 bits) along with some kind of public-key cryptography or hash of a public key value. Even algorithms as widespread as semantically opaque interface identifiers, as specified in RFC 7217 [30], are shown to be prone to breaches of user privacy once the internal state of the generation algorithm is known. Ullrich and Weippl successfully attack this form of SLAAC address self-assignment in [31], showing that even a chain of temporary addresses generated by these original privacy extensions form a side channel that allows attackers to synchronize to the generator's state and predict upcoming addresses.

There is also controversy about binding link-layer addresses into temporary IPv6 addresses for an interface, a la EUI-64. RFC 7721 [32] and RFC 8064 [33] express deep concerns about using link-layer identifiers in the address generation process, citing network activity correlation, location tracking, address scanning, and targeted exploitation as problematic byproducts. Likewise, Groat et al. discuss in [34] the privacy implications of using the link-layer address, or some deterministic value derived from it and not sufficiently randomized, in IPv6 stateless address generation. Nowadays, as mentioned, this process has indeed been sufficiently randomized by specification of RFC 8981 [29], superseding semantically opaque interface

identifiers to include ephemeral or pseudo-random values as components for address generation. Any SLAAC-related generation algorithm which can introduce some pseudo-random input from a random entropy source during the address generation process should produce addresses with sufficient randomization that prevents tracking and correlation of user activities.

2.6 Link-Layer Address Ownership

The security of any data dispatched or received at a layer above the link-layer becomes susceptible to interception if the link layer itself is not properly secured. Even Section 9 of the SEND specification [16] makes it very clear that regardless of the security protections afforded by SEND and CGA, there are certain issues that are inherent to an insecure link layer. Kiravuo et al. in [35] demonstrate a variety of link-layer vulnerabilities related to Ethernet, including MAC address spoofing, Man in The Middle (MITM) attacks with IPv4's ARP, replay attacks, and more. In a MAC spoofing attack, the threat actor sends frames through a switch with a spoofed source MAC address, causing the switch to direct response packets to the switchport through which the threat actor is communicating. This is less useful if the spoofed node is still online, however, as its own sending of legitimate frames will cause the forwarding table in the switch to revert to forwarding frames through the proper switchport. And if this process continues to happen back and forth in a flip-flop scenario, the legitimate node will be denied service because the switch will not reliably forward any information.

To solve this problem, most solutions will subsequently need to rely on some security mechanism at the link layer to certify or authenticate packets as coming from the correct switchport. Alternatively, nodes can attempt to enforce authentication by means of some knowledge known beyond the networking stack, such as a public-private keypair. While NDP

can enforce this requisite proof of knowledge (through a digital signature or some other means), it cannot enforce it for every packet or frame; thus, the verification only occurs during NDP transactions, such as when Neighbor Advertisements are being received. Due to this asynchronous nature of the protocol, it would be nearly impossible without link-layer security mechanisms to enforce that a frame does or does not originate from the original neighbor who owned the link-layer address.

2.7 Recent NDP Security Research

Well beyond the years of Secure Neighbor Discovery, researchers continue to strive for an ideal NDP: much research is conducted year after year into denial-of-service protections, protocol extensions, and, chiefly, the prevention of spoofing threats. It is important to create a so-called ‘measuring stick’ whereby the effectiveness of these recent endeavors can be reviewed, by evaluating how the goal of Neighbor Redirection attack prevention is achieved while simultaneously preserving the privacy, simplicity, and flexibility of the proposal. These three properties are significant considerations for the widespread success and adoption of a proposed protocol or NDP amendment. Such protocols and proposals to measure will include neither the canonical SEND and CGA specifications, nor Neighbor Discovery monitoring applications, because those have already been discussed and it is concluded that they do not satisfy these three ideals.

The Source Address Validation Improvement (SAVI) framework specified in [28] is an effective mitigation for neighbor discovery spoofing strategies in local networks. It learns and enforces known bindings between source IP addresses and source link-layer addresses, along with identifying the switchport used to communicate the traffic. But SAVI has been criticized for

its complexity and time-consuming implementation in networks of any large scale, along with its high false-positive rate for misreported address bindings. If deployed across the infrastructure incorrectly, there are difficulties with state synchronization as well because of how SAVI forms its associations. Additionally, SAVI is often not compatible with multihomed interfaces having multiple IP addresses across multiple subnets at the same time. As robust as the SAVI solution is, it cannot be considered to satisfy all three ideals of the measuring stick because it is neither simplistic nor flexible.

Praptodiyono et al. introduce the Trust-ND mechanism in [36] as another method specifically designed to abate the Neighbor Redirection attack vector by employing a 32-bit NDP Trust Option which includes a mixture of integrity checking and a soft-security technique of social trust between independent neighbors. This method was later improved upon and made more robust by Hasbullah et al. in [37], mitigating possible attack vectors and granularity issues relating to the Trust Option's Timestamp value. The most interesting aspect of this work is its independence from any centralized infrastructure; instead opting to rely on what neighbors might already know about each other, direct message authentication details, and calculation of a trustworthiness score, and then making NDP determinations from that information.

Decentralization and node-independent trust makes Trust-ND very flexible for deployment in mixed networks, where some nodes may not recognize or acknowledge the newly introduced Trust Option. It is also quite simplistic due to both its introduction of only one NDP option and its usage of an integrity hash and social trust scoring. However, Trust-ND fails the privacy requirement of the baseline measurement being used in this evaluation, because the trust score and knowledge of the source node's past activity relies partially on what the receiver (trustor) knows about the sender (trustee), identified by a packet hash. This means that for Trust-

ND to continually work by calculating high-scoring trust between nodes, both nodes will need to remain at fixed IP addresses and link-layer identifiers, since these are the identifiers used by Trust-ND to identify a trustee. If hosts cannot change their addresses freely, then they will be subject to tracking and activity correlation; thus, Trust-ND does not satisfy the ideals of this research.

Finally, NDPsec was introduced by Al-Ani et al. in [38] to augment Secure Neighbor Discovery practices and guarantee improved performance, amongst a litany of other proposed SEND improvements (algorithm changes, new options, etc.). While this is ultimately a discussion atop the existing SEND and CGA evaluation in this paper, it is worth mentioning due to its significance in the wider body of literature about NDP security. Many of these proposals still fail the measurement against privacy preservation, flexibility, and simplicity in one way or another thanks to their associations with SEND and CGA. However, they all offer a drastically improved baseline by which SEND and CGA can be evaluated in the modern day. Other analyses of these proposals and how they affect the efficacy of SEND and CGA–as well as analyses for SEND and CGA themselves–can be found in [22], [39], [40], [41], and [42].

Recent endeavors striving to secure NDP capture an exceptional amount of time, effort, and ingenuity to close what security researchers rightfully understand as a very important gap in the posture of local IPv6 networks. Of any solution already proposed, one particular methodology might make for a more suitable candidate than another depending on the context of the local network which is being secured. That is why this work does not desire to express that these related proposals are by any means inadequate ways to solve the issue; rather, it is left to administrators to determine which solution is best suited for their own use-cases. Instead, this research proposes an alternative to securing NDP against malicious traffic redirection attacks and

link-local address spoofing with NDP. It identifies what should be considered three crucial aspects for the adoption of any amendment to the NDP: simplicity, flexibility, and privacy preservation. Voucher-Based Addressing and Neighbor Discovery Sessions are thus introduced as further possible candidates to mitigate the Neighbor Redirection attack vector.

Chapter 3 Voucher-Based Addressing

NDP Address Resolution establishes a process for discovering the link-layer identifier (LLID) of a neighbor's IP address. But this process faithfully relies on some nebulous neighbor owning the target IP responding with its own LLID and not, e.g., a malicious node responding with a redirected LLID. If the target IP address is already being correlated with an LLID to which frames are forwarded, it is then sensible to tightly bind the two identifiers together: IP addresses should be provably derived from an underlying LLID. For the sake of individual IP address privacy, this binding needs to be computed in a manner that permits temporary and stable identifiers to coexist and in a manner that will not suffer the privacy concerns of the past.

Voucher-Based Addressing offers local IPv6 networks (1) a common procedure for binding LLIDs to IP addresses, (2) rotatable and private IP address generation, and (3) prevention of subversive on-path attacks. Address bindings use mutual key derivation functions to map public input components to deterministic output IP addresses. These bindings can be subsequently verified, using the same function, by neighboring nodes who seek to assert a target's address ‘ownership’ before initiating communications at higher levels of the network stack. All verifications are decentralized and do not require public-key infrastructure; only shared consensus on a distributed, pseudo-random value used to seed the address generation procedure. Despite its determinism, the address generation process creates rotatable IP addresses which appear statistically random to off-link devices, who are by design unaware of all input parameters associated with the addresses.

The concept of VBA is a cross-application of cryptographic key-stretching techniques to LLID bindings and neighbor consensus for generating random IPv6 addresses. The result is a high-impact, low-complexity, optional feature for the NDAR process, with minimal changes to NDP options, formats, or behaviors. It is proposed as an alternative SLAAC address assignment and verification methodology in contrast to SEND, CGAs, and opaque interface identifiers (from RFC 7217 [30]) in traditional local networks.

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 the proposal. To acquire more prerequisite context, please see Section 2.1 of RFC 4861 [11] for definitions of the following terms: neighbor, node, interface, link, address, router, host, on-link, off-link, IP, ICMP, packet, and target.

- **ND** (sometimes NDP): Neighbor Discovery (Protocol).
- **SEND**: Secure Neighbor Discovery.
- **CGA**: Cryptographically Generated Address.
- **NDAR**: The Neighbor Discovery Address Resolution process; see Section 7.2 of the NDP specification (RFC 4861).
- **NC**: Neighbor Cache, as specified in Section 5.1 of RFC 4861.
- **RS**, **RA**, **NS**, and **NA**: Respectively: Router Solicitation, Router Advertisement, Neighbor Solicitation, and Neighbor Advertisement. A collection of abbreviations for ICMP packet types defined by NDP in RFC 4861.
- **NUD**: Neighbor Unreachability Detection (Section 7.3 of RFC 4861).

- **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.
- **IID**: Interface Identifier. The unique identifier of an interface on a network. See Section 2.5.1 of RFC 4291 [14].
- **SLLAO**: Source Link-Layer Address Option. An ND option indicating the LLID of the packet sender or NDAR initiator.
- **TLLAO**: Target Link-Layer Address Option. An ND option indicating the LLID of the NDAR target.
- **DAD**: Duplicate Address Detection from the SLAAC specification (RFC 4862 [13]).
- **SLAAC**: Stateless Address Autoconfiguration.
- **PKI**: Public Key Infrastructure. A system that manages asymmetric keypairs and digital certificates to verify user identities and to secure network communications authoritatively. Often used in reference to some larger specification such as X.509 PKI.
- **VBA**: Could mean one of two things depending on context:
    - Voucher-Based Addressing (such as "the VBA-enabled subnet" or "VBA mandates this").
    - Voucher-Based Address (such as "a VBA" or "using VBAs"). An 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**: ND Link Voucher option. A data payload intended to be distributed by a responsible node on-link. Details are statefully maintained on neighbors and are used in both generating and verifying VBAs. This term is also used synonymously with vouchers themselves as they are stored at each node per interface.
- **VS**: ND Voucher Summary option. Used to hint to receivers which LV identifier and IEM is being used by the sending node.
- **LOVMA**: Local On-link Voucher Multicast Address. A multicast channel used by VBA-enabled nodes to get non-essential information from the current Voucher Bearer or from other VBA-enabled neighbors.
- **VB**: Voucher Bearer. The on-link node that is solely responsible for dissemination of the LV and authorized by any potential link guarding to transmit Router Advertisements or Redirects with an LV attached.
- **VSR**: Voucher Status Report. A type of data payload sent by VBA-enabled nodes to the LOVMA. It shares information about the node's VBA preferences and is mainly used in optimizations as an optional protocol feature.
- **VCI**: Voucher Capability Indication. A type of LOVMA data payload sent by candidate VBs wishing to indicate their candidacy as a future VB for the link.
- **VHA**: Voucher Handoff Advertisement. A type of data payload sent by the current VB to the LOVMA, signing off on an election process for a new LV.
- **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. 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" when discussing the binding of the link layer to the network layer.
- **KDF**: Key Derivation Function, as defined in Section 3 of RFC 8018 [43].
- **Salt**: An extra random value used in computing a hash which makes it impossible for attackers to precompute output values. See Section 4.1 of RFC 8018 for more information.
- **IC**: Iterations Count, also synonymous with the term “L value”. For the sake of clarity, this term is often not abbreviated. It is defined as an integer value describing the number of times a key derivation function is iterated to produce a final output value. See Section 4.2 of RFC 8018 for more information.
- **Hextet**: A 16-bit aggregation; data that is 16 bits in size.
- **RA-Guard**: Router Advertisement Guard, as specified in RFC 6105 [44].

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 on which malicious nodes can impersonate other neighbors if those victims are somehow disconnected from the link.

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 & Protocol Overview

VBA boasts a trinity of privacy, simplicity, and flexibility. It successfully creates a set of rotatable interface network addresses bound to a single link-layer address, which cannot be discovered or tracked by reversing or correlating any of the generated network addresses. These addresses appear to external hosts to be randomly generated and disjunct. By design, only neighboring nodes communicating Neighbor Discovery packets can know of all parameters used as inputs to the deterministically generated address; knowledge that also acts as a requirement to perform address verification. This same neighbor is aware of a Link Voucher Neighbor Discovery option attached to Router Advertisements, providing shared address generation parameters. The Link Voucher is distributed by a responsible, preordained neighbor on-link and it guarantees a mutual agreement between all link neighbors for VBA generation.

A selectable one-way key derivation function is used to dynamically stretch computation of a hash using a combination of locally known inputs, some of which reside within the agreed upon Link Voucher. The key derivation function is employed to slow the brute-force computation of hash prefixes that might cause a collision or be utilized maliciously. The property of voucher distribution ensures that two neighbors must use equivalent and mutual Link Voucher information when generating and verifying neighbor addresses. Failure to verify a neighbor begets failure to communicate with that neighbor, since its link-layer address will not be entered into the NDP cache of the verifying node.

VBA summarily seeks to maximize its aversion to labyrinthine intricacies succumbed to by other seldom-adopted NDP security proposals, while creating many opt-in opportunities for optimization at the same time. This is a careful balance to strike in specification complexity. It achieves this balance by chiefly minimizing the use of public-key cryptography, infrastructure changes, and centralized registration authorities, to offer the best alternative for securing NDAR transactions that is more adaptable and flexible. Simultaneously, unlike other protocols, VBA amends normative Neighbor Discovery processes as little as possible by modifying address generation, adding new NDP Option types, and inserting a small verification shim as a required pre-cache determination of neighbor legitimacy. The reward is opt-in neighbor LL2IP binding verification, the flexibility to choose interface participation granularly, and the safe derivation of one or more randomized (yet deterministic) interface addresses.

3.3.1 Link-Layer Address Binding

Consider how and why a malicious party might engage in a MAC spoofing attack as presented in Section 3.B of [35]. When two nodes share the same link-layer identifier in a switched network, frames will unreliably be forwarded to one of the two parties based on who most recently communicated through the switch. This flip-flopping of frame delivery causes a confusion of higher-level protocol stacks and will most likely result in a denial-of-service attack on the legitimate node. Therefore, the impersonator can gain little advantage by spoofing a neighbor's link-layer address when both nodes are online, and the principle of MAC address uniqueness per broadcast domain can be established.

During the NDAR 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 such link-layer identifiers become inputs to higher-layer abstractions, such as IP addresses in the case of VBA, then there are 'bindings' of the identifiers with the resultant upper-layer components. These bindings are especially useful if one-to-one, irreversible, deterministic input-output pairs are created as a result, e.g. by incorporating them into hash function inputs. Since VBA generation depends on both these bindings and the principle of MAC address uniqueness at the scope of each local link, VBA verifications are considered valid proofs of MAC address ownership so long as the originally communicating neighbor remains on-link.

VBA verification is run at specific times during NDAR transactions by verifying neighbors, independently mirroring the VBA generation process. The process of verification is 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 'outside' information is found within the 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 report an association of an IP address to an LLID that cannot be bound to it. This means the NDAR process becomes safe from issues of impersonation. A node wishing to contact a neighbor's IP address can no longer be first subversively redirected to a different receiver at a lower level of the network stack.

3.3.2 Key Derivation Functions & Address Privacy

Binding an LLID to a higher-layer address using a simple embedding scheme should suffice if the goals of VBA only included validating address ownership. For example, modified EUI-64 interface identifiers are formed by a long-established address derivation methodology which uses the LLID of an underlying interface. This would sufficiently create a binding, but it would not be private because the LLID of the network interface is exposed in the resultant address and becomes trackable (even in the case that RCM is used). A design goal of VBA is to also establish a privacy-focused address generation procedure which will obscure the node's LLID while permitting on-the-fly address rotations. EUI-64 is by design a fixed and rudimentary address derivation methodology which does not permit such flexibility.

For this requirement, VBAs employ more sophisticated hashing during the address generation process to create a pseudo-random output address. A hash-based address does not allow outside trackers to know the LLID of the node. Using hashing and key derivation techniques ensures that any LLID of an arbitrary length can be reliably bound to an irreversible address suffix that is fixed at 64 bits in length. Furthermore, simply hashing an LLID will only create a one-to-one binding, but many formalized IP address generation schemes already offer ways to derive many privacy-focused addresses from a single input LLID (e.g., Section 5 and

Appendix A.3 of RFC 7217 [30]). These addresses are intentionally obfuscated and preserve the privacy of the node unless reversing parties are aware of all input parameters used by the deterministic address generation function. VBA strikes a careful balance of (1) keeping off-link nodes unaware of local Link Voucher information used in address composition, and (2) ensuring on-link nodes are aware of all parameters used to generate any neighbor VBA. Off-link nodes cannot derive an external target's bound LLID value because they cannot receive NDP messages from the target's broadcast domain, nor can they determine the binding from any patterns of the address itself. VBAs will always appear random as a consequence of utilizing the outputs of deterministic hash functions.

To gain a further advantage, VBA elevates the use of simple hashing to the use of key derivation functions (KDFs), which easily enable a set of one-to-many LL2IP bindings and enforce a minimum address computation time. KDFs accept input iteration counts specifying how many times the pseudo-random function or underlying hash function must be iterated before producing a final result. When used with VBA, they are computed with various inputs that specifically identify the target node's details, and a small fragment of the resulting value is planted into the generated VBA. Iteration counts (ICs) can then be embedded into resultant IP addresses adjacent to KDF hash results, such that the following three components are an inherent value exchanged in any NDAR transaction: (1) the target's reported LLID and IP address (i.e., VBA) in NDP fields; (2) a portion of the KDF's hash output (embedded within the VBA); and (3) the IC value that was supplied when computing the KDF hash (also embedded within the VBA).

Interfaces using the VBA generation procedure 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 embedded IC value produces an entirely new, seemingly random address with no correlation to the previous one produced by the adjacent IC. Nodes falsifying any of the bound values used by the legitimate VBA owner will be rejected by verifying neighbors when resolving and verifying LL2IP bindings during NDAR processes.

3.4 Address Generation

VBAs are indistinguishable from ordinary unicast IPv6 addresses because there is no reserved sequence of bytes, or so-called ‘magic number’, embedded within them to classify them as such. They 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 host. VBAs, alongside many other already-established IPv6 protocols, are 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. Figure 3 provides a simple visual representation of how an IPv6 address is partitioned.

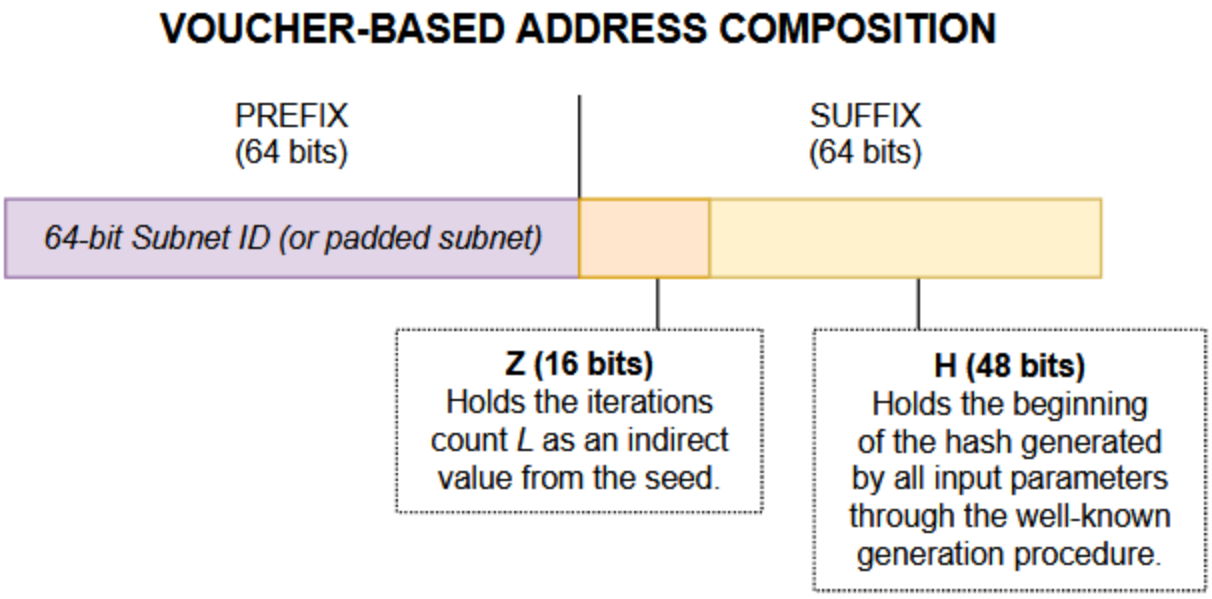


Figure 3. The overall structure of a Voucher-Based Address. The left half of the address is not a result of the address generation procedure and merely consists of the subnet (padded if necessary).

Addresses are generated using inputs from various sources: the shared Link Voucher details, local network interface details, and an arbitrarily chosen iterations count value L. If any of these components are missing, then the generation process will fail and fall back to some alternative method, while adjusting the current IEM to something more lenient. Figure 4 shows the step-by-step process used to generate VBAs on each interface for one or more available on-link prefixes at the same time. In the image, the large cog from step 4 indicates the primary function driving the process; this is the ‘well known procedure’ for VBA generation at each compatible and VBA-enabled network endpoint.

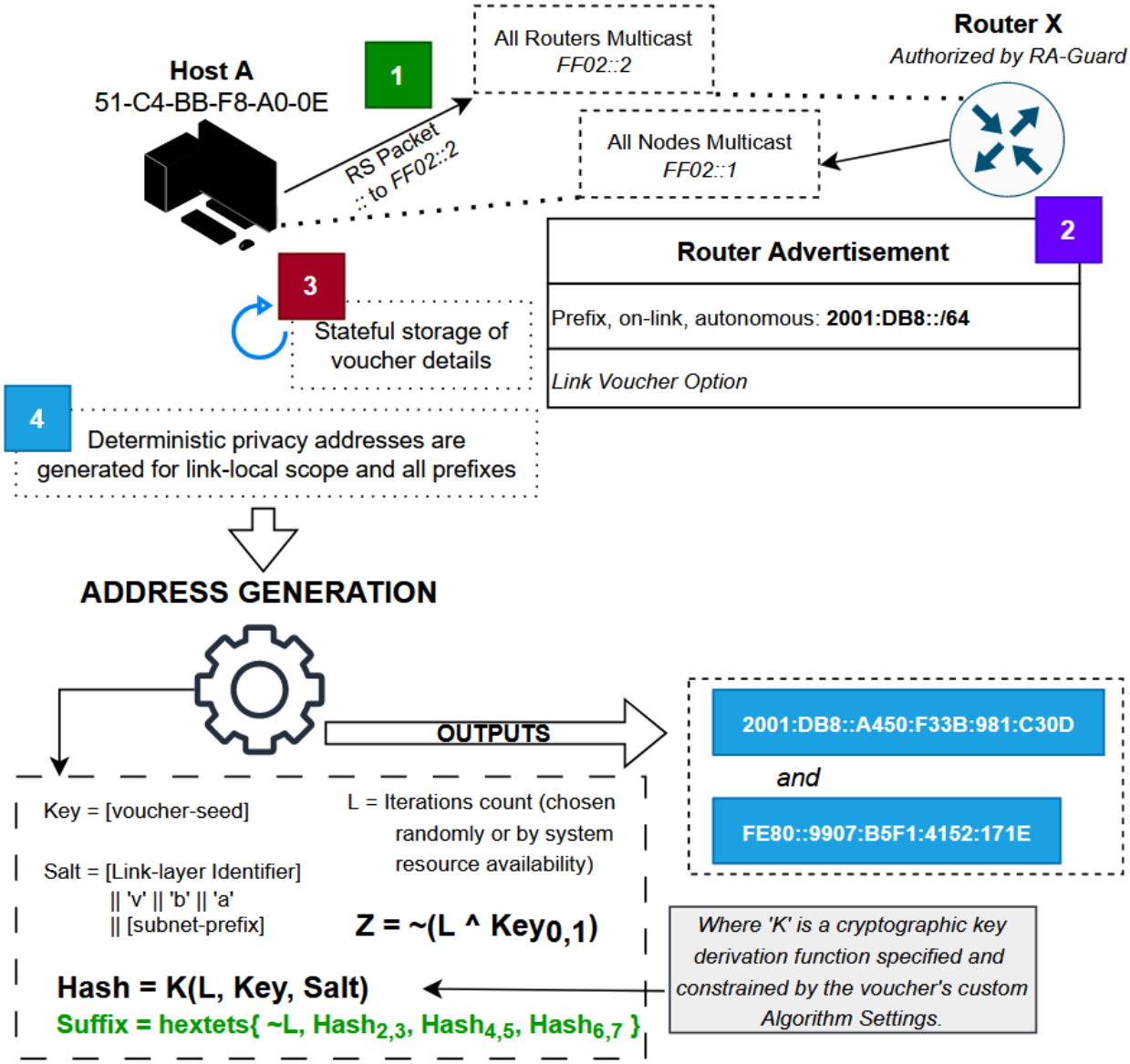


Figure 4. The Voucher-Based Address generation procedure is used to generate all initial interface addresses from the interface on Host A. Host B is a Voucher Bearer authorized by local policy to delegate Link Voucher information to on-link neighbors.

The Interface ID (IID) 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 Link Voucher seed value. The Z value is used to ensure the same input iterations count value, L, will

be unique across different Link Voucher seeds and provide enhanced address privacy. This is especially necessary if the node locally advertises a well-known or Preferred Iterations Count value. The L value is a significant member of the generated VBA: this parameter controls how many times the KDF function specified by the LV 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 H portion of the VBA, where its first hextet (bytes 1 and 2) is replaced with the Z value as shown in Figure 4.

The address generation algorithm is detailed procedurally as follows:

1. A node connects to a network and discovers VBA compatibility from Link Voucher details 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 LV details contain instructions for KDF parameters and algorithm selection, as well as the 128-bit voucher 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.

4. 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 LV seed.

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

3.5 Address Verification

When enabled and enforced by a receiving interface's IEM, the VBA verification process uses the information embedded within the IP address that is provided during an NDAR exchange and mirrors the generation of the address locally. Figure 5 illustrates this process.

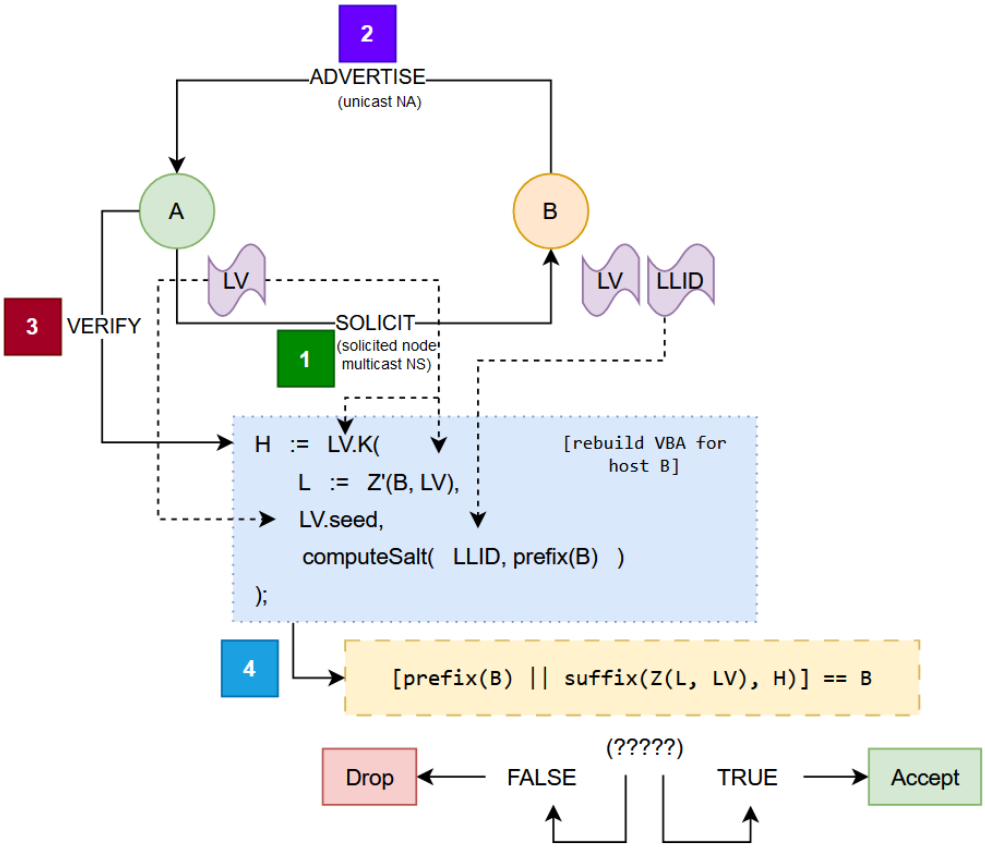


Figure 5. For VBA verification, the entire address generation process is repeated at the verifier using various values known about the neighbor, the Link Voucher, and the current NDAR transaction.

If the reconstructed address of the target node does not match the 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 current link Voucher Bearer. If the verification procedure fails due to an LV 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 varying LVs being distributed simultaneously.

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 embedded in 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.

Z(L, LV) = ~(L ^ LV.seed[0..1])

Z'(B, LV) = ~(B[8..9] ^ LV.seed[0..1])

3.6 Interface Enforcement Modes

Each interface participating in the VBA paradigm–i.e., having any awareness of VBAs whatsoever–must always have the option to set a single local Interface Enforcement Mode (IEM) which determines its handling of NDP traffic in relation to VBA. IEMs allow granular, per-interface flexibility to adjust the behaviors of each interface in real-time. They are specifically designed to be changeable at-will, at any time and for any reason. A small diagram representing the IEM state machine would simply show all states as being transitionable to one another. The IEMs, in order of increasing strictness, are specified as such:

- Address Awareness Disabled (AAD): The interface must disable any generation or verification of network addresses. It must completely withdraw from any activity related to VBA, with the exception that it can still maintain a listen-only awareness of the current Link Voucher state.

- Address Generation Only (AGO): The interface address generation process is followed during SLAAC, but the address verification shim must be disabled and be ignored for all NDAR transactions.
- Address Generation and Verification with Levels (AGVL): Address generation and validation is performed, but any failure to validate a neighbor must not be strictly enforced. Instead, purported LL2IP bindings which fail to validate are tagged in the Neighbor Cache as “Unsecured” entries, and those which successfully verify are tagged as “Secured”. Secured responses are strongly preferred over Unsecured ones, which permits successfully verifying nodes to receive connection priority without denying communicating with neighbors that do not successfully verify. This tagging and preference for Secure communication is directly borrowed from the SEND specification in Section 8 of RFC 3971 [16].
- Address Generation and Verification (AGV): The strictest mode, enforcing all VBA rules without mercy. Any NDAR process not passing the verification shim–i.e., an invalid reported binding–must be dropped immediately.

3.7 Behavioral Neighbor Discovery Changes

VBA does little to modify the NDP in an incompatible way for existing systems, instead opting to change the behavior and decision-making of the ND-compatible software stack based on the current IEM. Figure 6 expresses the application of these changes in a practical scenario, where both hosts 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:

- Since LLIDs are bound to IPs and VBA collisions are highly unlikely, new LLIDs on neighbors have an impossibly low chance of organically producing the same VBA as one already cached by verifiers. Such an unlikelihood implies that any Neighbor Advertisement (NA) packets including a known, cached Target Address that is not in the INCOMPLETE state should be ignored if an included Target Link Layer Address Option (TLLAO) attempts to update the record to a different LLID, or if it attempts to change the Neighbor Cache (NC) entry to a different state. This acts as a protocol optimization and denial of service protection which prevents malicious nodes from attempting to continually imbalance the processing bandwidth of a target.
- The value of an LLID within a Neighbor Solicitation (NS) packet should likewise never update for the same IP Source Address when the current LV has not changed. This is because it is a statistical improbability for any known VBA to have been organically formed from a different LLID when the LV has not changed. Therefore, NS packets with an SLLAO attached must not update the state or values of any current NC entry having the IP Source Address value from the NS.
- Any supposed urgent updates about underlying details for a known VBA are unnecessary. The Override flag in received NAs should not be able to freely update the underlying LLID of a current NC entry. Overrides should also generally not be able to affect the state of any NC entry. However, some devices might wish to support a laxer 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.

This is an insecure practice and is not recommended unless used explicitly for static addresses.

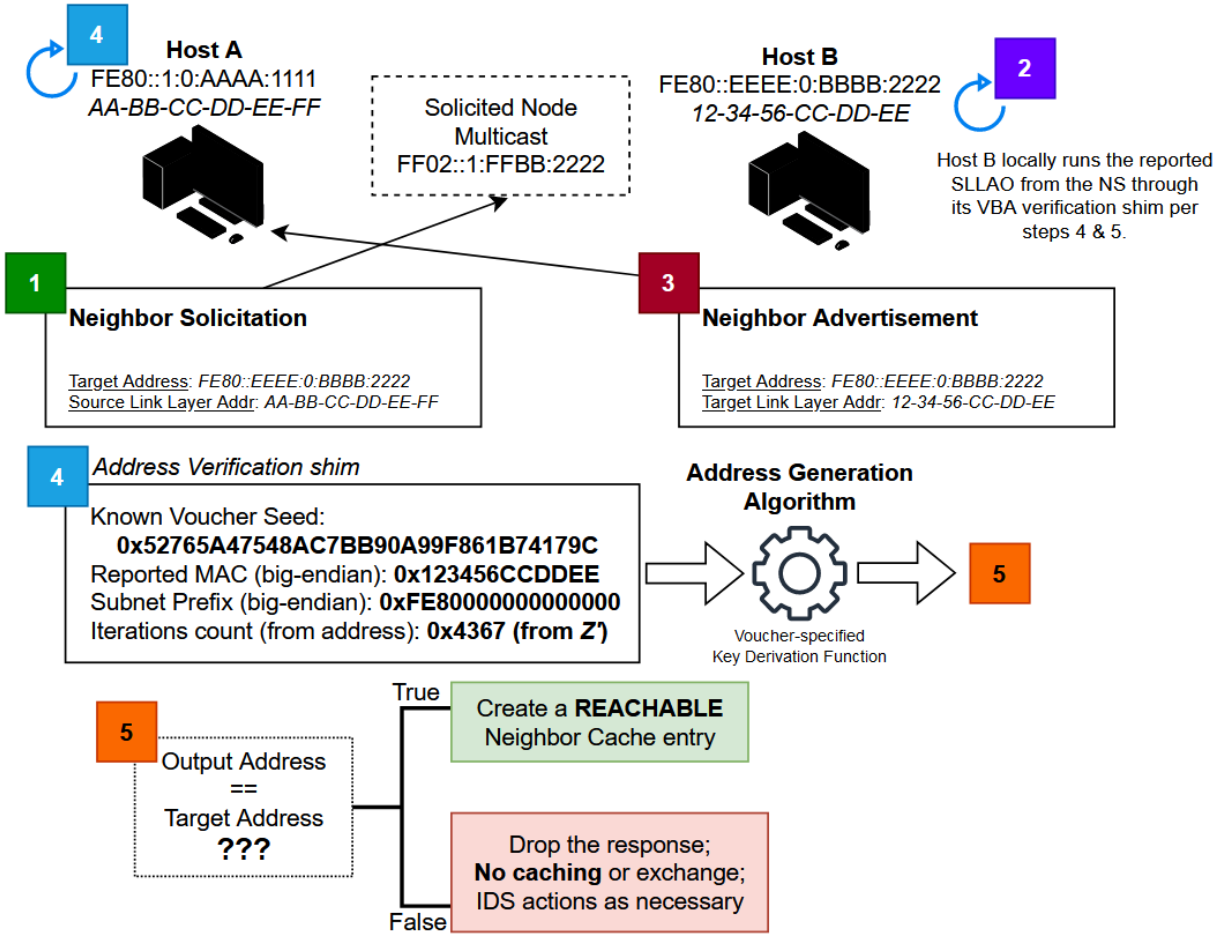


Figure 6. VBA processes do not modify the typical Neighbor Discovery process or exchange. Instead, software local to each interface will act to verify received IP addresses during NDAR based on the interface's selected Enforcement Mode.

3.7.1 The Address Verification Shim

VBA verification is a 'shim' software process–a small functionality that is added as a step between two existing procedures–prescreening incoming requests to insert or update cached

bindings between IP addresses and LLIDs, according to the normative ND process. If the verification shim rejects a binding from entering the NC, or some update of the NC from occurring, then the verifying node will be denied from properly forwarding data frames to the requesting node. This is because a cache entry in the REACHABLE state does not exist and will not be created. Prefiltering in such a manner immediately dismantles any neighbor's opportunity to forge NDAR packets or to redirect traffic maliciously.

Employing the verification shim results in repeated KDF computations that could impact performance significantly for low-power nodes or other embedded systems, so the shim must be optimized and called as seldom as possible. As such, VBA verification should only be performed when updating or creating an NC entry through NDAR exchanges. For the sake of optimization, Neighbor Unreachability Detection exchanges must not use the verification shim when none of the NDAR parameters–i.e., the IP address or the LLID–are being changed. Incoming NDAR packets 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 NDAR 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.

- An NA or Redirect packet is received on a node supporting Gratuitous ND and the Target Address does not already have an NC entry present on the receiver.

The above list is perhaps not all-inclusive and does not consider other ND extensions which may allow certain ND packets to modify NC entries. Except for forward progress indications through NUD, NDAR packets of any type seeking to update any active NC entry (whether state or values), or to create a new entry, must be pipelined through the VBA verification shim process first, depending on the current IEM.

3.7.2 Neighbor Unreachability Detection

The current NDP specification defines Reachability Confirmations which serve to regularly update NC values when one of two types of hints indicates connections with already-cached neighbors are making "forward progress" (Section 7.3.1 of RFC 4861). Forward progress signals that an established connection with a neighbor is still ongoing and that a neighbor is still considered REACHABLE in the NC. Nodes engage in the NUD process to keep their NC entries in their ideal REACHABLE states as a protocol optimization, as it is costly to continually rediscover active neighbors.

VBAs capitalize on this behavior by foregoing address verification requirements when NS/NA transactions only serve to express forward progress. This means any forward progress showing no changes in the expressed LLID and IP address of a current NC entry must allow the record to be refreshed as REACHABLE without requiring expensive use of the VBA verification shim. Any forward progress indicating that a change has somehow occurred in the LLID for a

cached IP address must be ignored and must not update the cache unless it is reverified by the verification shim process.

3.7.3 Link Voucher Updates

Any expiration or rotation of a current LV used to verify VBAs must cause the non-static entries in the NC to be immediately cleared. A change in the LV could occur for any number of reasons, but the swap indicates that the current LV is no longer in active use and therefore any addresses which were previously cached during its reign must be dropped. VBA implementations might wish to categorize or label NC entries by their LV ID value used during their shim verification as an optimization. Labeling cache entries by their temporal voucher ID permits other NC entries to remain despite the LV rotation, such as those entries originating from a new LV after a voucher handoff occurs.

When a new LV is accepted and cached, whether by handoff or due to the absence of a current LV, any current NC entries–especially heuristically determined busy or recent ones–may have their LLIDs pre-computed into the resulting VBAs, which use the parameters specified by the new LV. This occurs as an optimization even if no NDAR transactions have been required for those neighbors yet. The process necessitates that the pre-computing party is aware of the neighbor's optional Preferred Iterations Count value, specified at an earlier time via the LOVMA channel. When VBAs generated from new voucher details can be pre-computed without waiting for a transition to complete, then most neighbors can immediately continue or resume secure communications without any potential delays incurred by re-querying their VBA verification shim processes. It is left to the discretion of each VBA implementation to apply this optimization

where desired. Additionally, static mappings defined in the NC must always be preserved regardless of LV expirations, since LV activity has no bearing on static cache entries.

3.7.4 Gratuitous Neighbor Discovery

Gratuitous ND, defined in RFC 9131 [45], allows routers to preemptively create STALE NC entries from received NAs, to expedite the exchange of local neighbor LLID bindings. VBAs should support this option, since routers preemptively verifying any neighbor's address bindings will allow the neighbor to communicate off-link much faster than if the router first required a full, on-demand NDAR process including VBA verification. Implementations of VBA should be considerate of how Gratuitous ND might interact with certain IEMs requiring VBA verification. For example, if a flurry of NA packets is received in an ostensible attack, the router might quickly find itself with too much queued work and could start dropping packets under load. Implementations might therefore wish to toggle enablement of this feature reactively based on the router's evaluated system load.

3.7.5 Duplicate Address Detection

When generating a VBA, the node follows the normative means of Duplicate Address Detection (DAD) specified by the SLAAC RFC (in section 5.4 of RFC 4862 [13]). The DAD procedure ideally follows any other applicable DAD optimizations that might already be present or known by the node (e.g., RFC 4429 [46], RFC 7527 [47], etc.). Upon detecting any duplicate address, nodes using VBA must select a different L value during VBA generation to assign themselves a different, non-conflicting address. Regenerating a series of new VBAs can quickly become computationally expensive based on the amount of address collisions detected by DAD,

and this can be abused by malicious denial of service attacks. To counter this weakness, VBA implementations should employ one of two optimization options based on the initially selected L value during VBA generation:

L > 4

Cache the 4 leading KDF computations (L-4 through L-1) during address generation.

L <= 4

Cache the KDF computation result at the L value only.

Implementations should always prefer the option where the L value is greater than four, because L-4 through L-1 comprise intermediate KDF outputs that are already required in order to calculate the hash at the final L value. Conversely, any L value at or under four will cache the generated KDF hash only at L and then increment the input L by one for each DAD collision, up to four times. Figure 7 demonstrates this process in more detail.

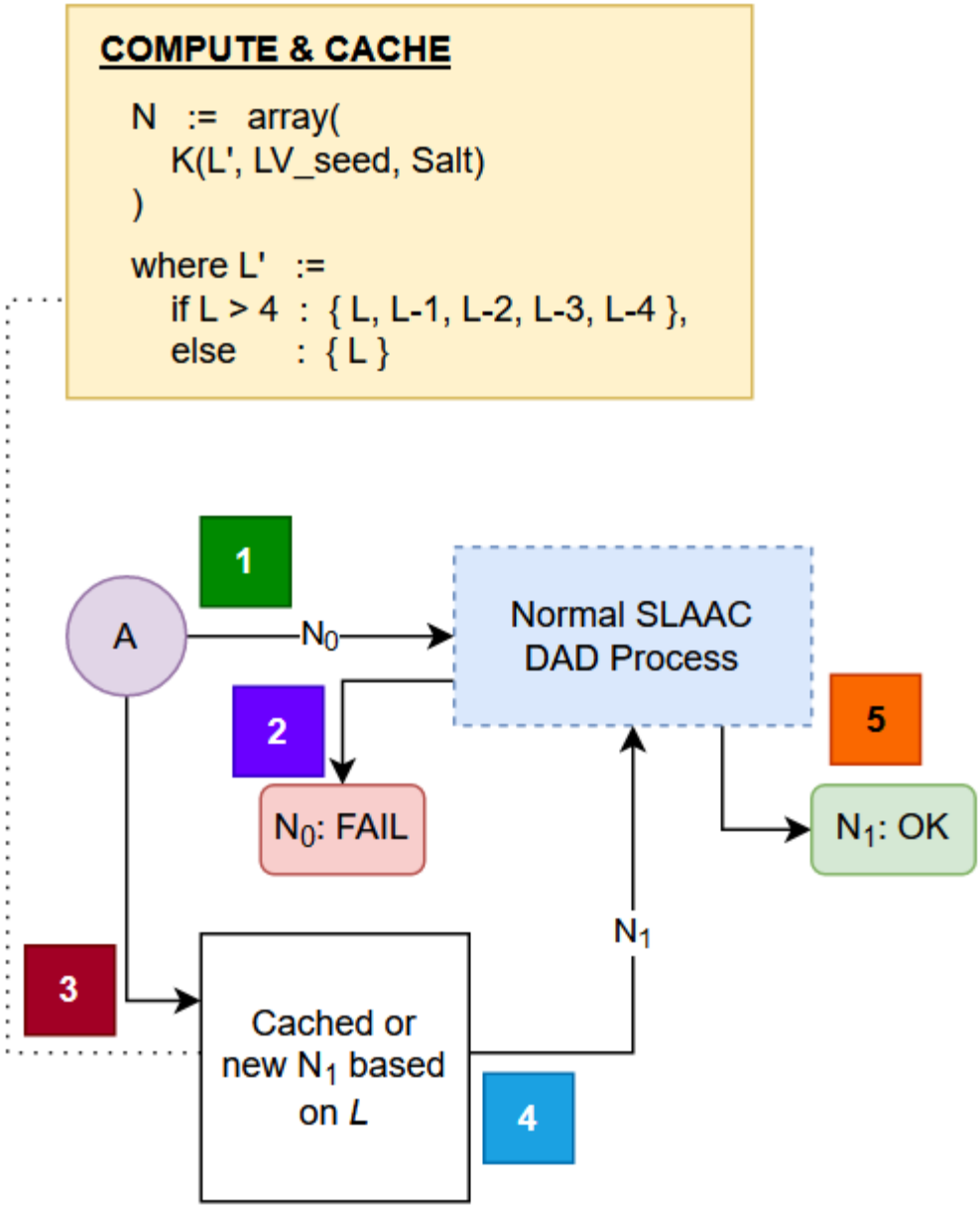


Figure 7. The DAD optimization process for VBA is shown. When L is greater than four, the array N consists of any precomputed and preserved Key Derivation Function outputs that are already intermediate values leading up to the computation of the KDF at L iterations. Otherwise, N is an array of a single element and computing the next L+1 value is trivial at a low iteration count.

In the Figure, Moment (1) shows Node A engaged in DAD using the address suffix generated by the KDF with the iterations count set at L. After the collision is detected in Moment

(2), Moment (3) shows the new VBA being immediately tried using the cached hash value N_1 with L-1 iterations as the input IC. N_1 is chosen without needing to recompute the KDF hash. The DAD process is then successful and there are no longer any duplicate addresses, so the node assigns itself the generated VBA. To further cement this important optimization procedure, a written example process follows.

1. A new network node has received LV details; it specifies using PBKDF2 as the KDF.
2. The node arbitrarily selects 0xFF04 as its link-local-scope L value.
3. The node will iterate the PBKDF2 function through 0xFEFF.
4. The PBKDF2 cipher output for 0xFF00 (or L-4) iterations will be cached.
5. The node will do the same for the next three iterations counts 0xFF01, 0xFF02, and 0xFF03.
6. It will compute the final PBKDF2 round at 0xFF04 iterations and will use the result to generate a valid link-local VBA according to the address generation procedure.
7. When following the normative DAD procedure, a collision is detected for the VBA.
8. The node then immediately falls back to the KDF hash result from the L-1 value at 0xFF03 to generate the next possible link-local VBA.
9. This new VBA is completely different and does not register a DAD collision, so the interface can successfully use the generated VBA.
10. The optimization has successfully removed the need to recompute the PBKDF2 algorithm up to a newly assigned L value, saving time in the VBA-enabled SLAAC and DAD processes.

If all 5 attempted IC values result in DAD collisions, then the node should give up and use some other implementation-specific course of action to contact an administrator or log a system management error. Genuine and benign DAD collisions are a dangerous prospect for VBA. Address collisions imply that a separate LLID with an equal L value, voucher seed, and KDF parameterization has generated a KDF hash collision in a rare occurrence (at least within the first 8 bytes), exposing the possibility for node impersonation. Some implementations might attempt to use trusted mechanisms to detect such VBA collisions. This might be done by means of intermediate device monitoring, such as switching hardware and other network infrastructure, with action(s) taken appropriately based on its detection.

Nodes encountering a duplicate address will by necessity require a different L value to regenerate their current VBA. If the node uses and advertises a Preferred Iterations Count value, then it is highly recommended that the node sends a gratuitous VSR update to the LOVMA channel with the new preferred value. Any further protections to mitigate denial of service attacks abusing the DAD mechanism are beyond the scope of this research. Since VBAs do not modify the actual DAD process, any predefined DAD denial of service protections can likely apply similarly when using VBAs.

3.8 Link Vouchers

The Link Voucher is an optional attachment to Router Advertisement and Redirect messages which dictates the parameters used by neighbors to generate their VBAs. By agreeing on these shared, distributed parameters during address generation, neighbors are able to verify each other's addresses independently by partially using the same information. Establishing a link-local baseline for VBA generation parameters also enhances the privacy of generated IP

addresses, because external nodes will not be aware of all inputs utilized in the generation of a final interface address (or set thereof). Link Vouchers work as the namesake of VBA: they are vouchers upon which all subsequently compatible unicast IP addresses generated in the SLAAC process are based upon.

3.8.1 Acquisition

Interfaces connecting to the link for the first time are required by VBA to accept and cache the first valid LV received from any neighbor. If the interface intends to supervise the delegation and creation of the LV as a Voucher Bearer, it follows a process of attempting to discover a currently established LV. If an existing LV is not detected in sufficient time, then it will create and transmit its own. LV options can be discovered by issuing an ordinary Router Solicitation packet according to normative ND processes. Interfaces receiving multiple valid LVs simultaneously will opt to use the LV with the most recent Timestamp value (that is, the Timestamp value closest to the local system time of the receiving device).

If an active LV expires–meaning no updated LV has been received within the number of seconds specified in the Expiration field of the most recent LV–then the interface will again revert to accepting the first received LV or becoming a VB. This decision is based solely upon the dynamic decision-making of the underlying VBA implementation. The same Expiration time for an LV can also elapse for an interface while it is disconnected from the link. If such an expiration occurs, then that interface must again follow the same LV acquisition process.

Because LV distribution to interfaces requires automatic Trust on First Use (TOFU) [48], it is essential for more adversarial networks to implement some form of protection against rogue LVs at a lower or intermediate level. In the cases where these protective mechanisms are not

available or sufficient, administrators might choose to set LV information on each node manually, in which case all other acquisition processes are overridden. Administrators in this situation should also choose to employ some form of intrusion detection to better mitigate rogue LVs from appearing and affecting the entire LAN.

3.8.2 Managing Transitions

The node responsible for the LV can at any time issue a handoff of that responsibility to another node by using Voucher Handoff Advertisements (VHA) on the LOVMA channel. During the period of transition between the previous LV and the new one, VBA-conscious interfaces subscribed to the optional LOVMA channel will receive VHA multicast packets specifying the new LV. These LOVMA-connected interfaces are strongly recommended to allow both LVs to be cached, so that VBAs generated using either LV are immediately valid. The same interfaces are also strongly recommended to begin VBA generation with the new LV parameters to be active in parallel with existing VBAs from the previous LV. Such a recommendation is in anticipation of the new LV becoming fully active once the previous LV finally expires.

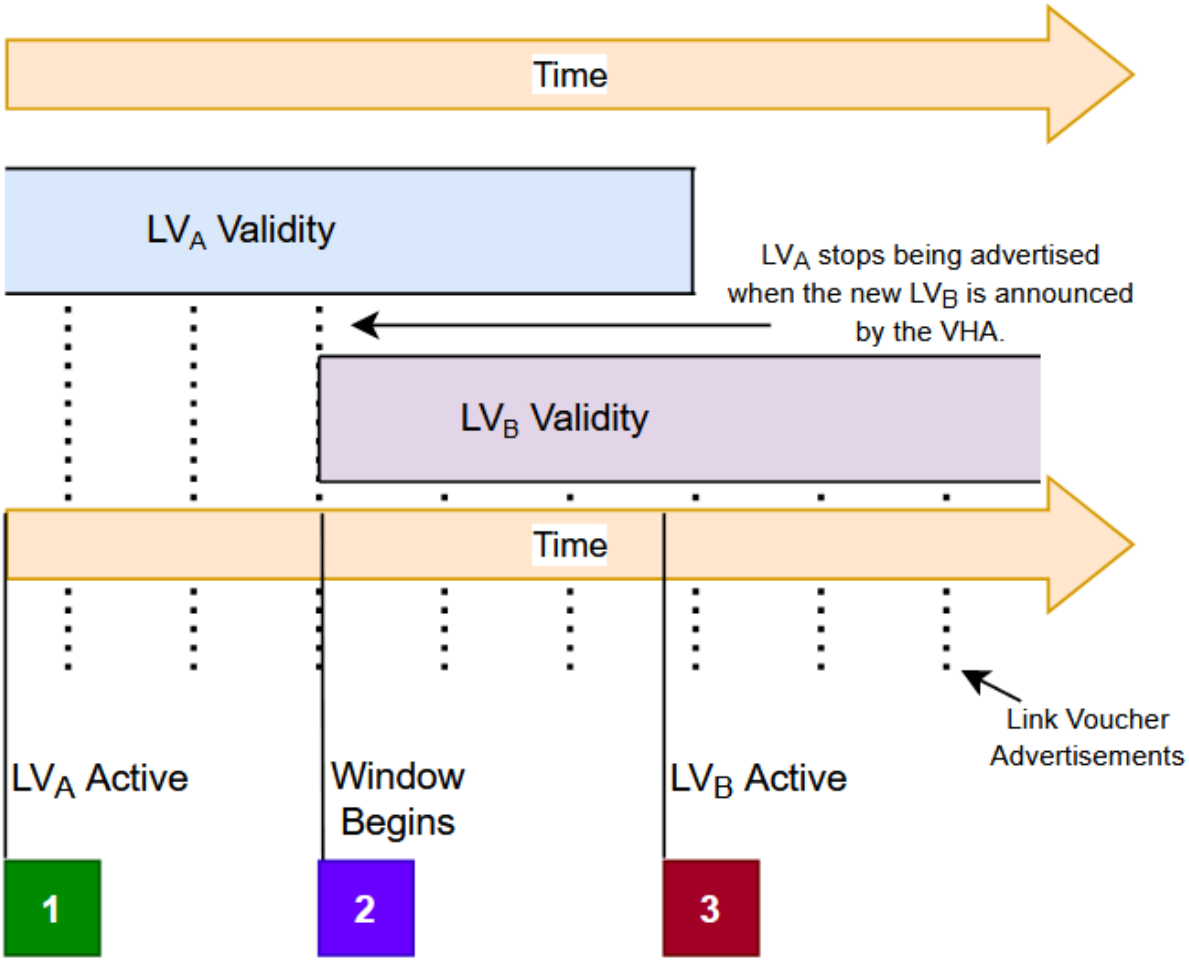


Figure 8. The Link Voucher transition process occurs in a window of overlap where both the previous LV and the new LV are considered valid for verifying neighbor VBAs.

If another VHA appears indicating a third LV is being appointed for election, receivers will ignore that VHA until one of the two LVs from the original VHA has expired. This prevents VHA flooding which flags several active LVs on the same link as being valid, causing an 'address storm' that drains available resources from link nodes. Once the LV transition window ends, the number of valid LVs must return from two to one. The transition window ends when the original responsible node fails to refresh its LV within its LV-specified Expiration time, thus

purposefully letting its LV time out. An expiration event usually indicates a final transfer of responsibility for the LV.

For interfaces that do not subscribe to or regard optional LOVMA channel VHA datagrams, the LV transition process becomes more akin to a hard handoff. Due to LV acquisition requirements, these interfaces will not trust the new LV until the previous LV has fully expired, at which time any provided LV becomes acceptable. For this reason, any VBAs preemptively generated with the upcoming LV will not be successfully verified by neighbors who are unaware of the LV transition, until the transition window has ended, and the new LV becomes the primary voucher on all nodes. All implementations are therefore strongly recommended to listen for and parse VHAs in order to secure the handoff process, preventing rogue VBs from timing their own malicious LV injections during a hard handoff.

When a handoff is completed, the LV for the interface has changed. Any time the stored 32-bit identifier of the active LV changes to another, the interface's Neighbor Cache must be cleared and all VBAs, whether generated or verified, must be derived from the parameters of the newly active LV only. This is true even in the case of a hard handoff. The transition window provides an opportunity for optimization: if neighbors are aware of the upcoming LV, then they might opt to preemptively generate their new VBAs in anticipation of the completed LV transition. Implementations might also choose to utilize this transition window for pre-caching the computed addresses of cached neighbors who advertise their own Preferred Iterations Count values specified in LOVMA channel VSR packets.

Finally, if the current node responsible for the LV either disconnects from the network or lets its LV expire without an election process, then the network becomes open again according to the LV acquisition process. When this occurs, other nodes are permitted to fill the LV vacuum

with their own. If none of the other nodes assume responsibility while the primary VB is away or not transmitting updated LVs, all VBA-enabled interfaces retain the most recently stored, valid LV for the purposes of VBA generation and verification until a new one becomes available. When any new LV becomes available, the LV acquisition process subsequently applies because the previous LV has expired and not been renewed.

3.8.3 Option Structure

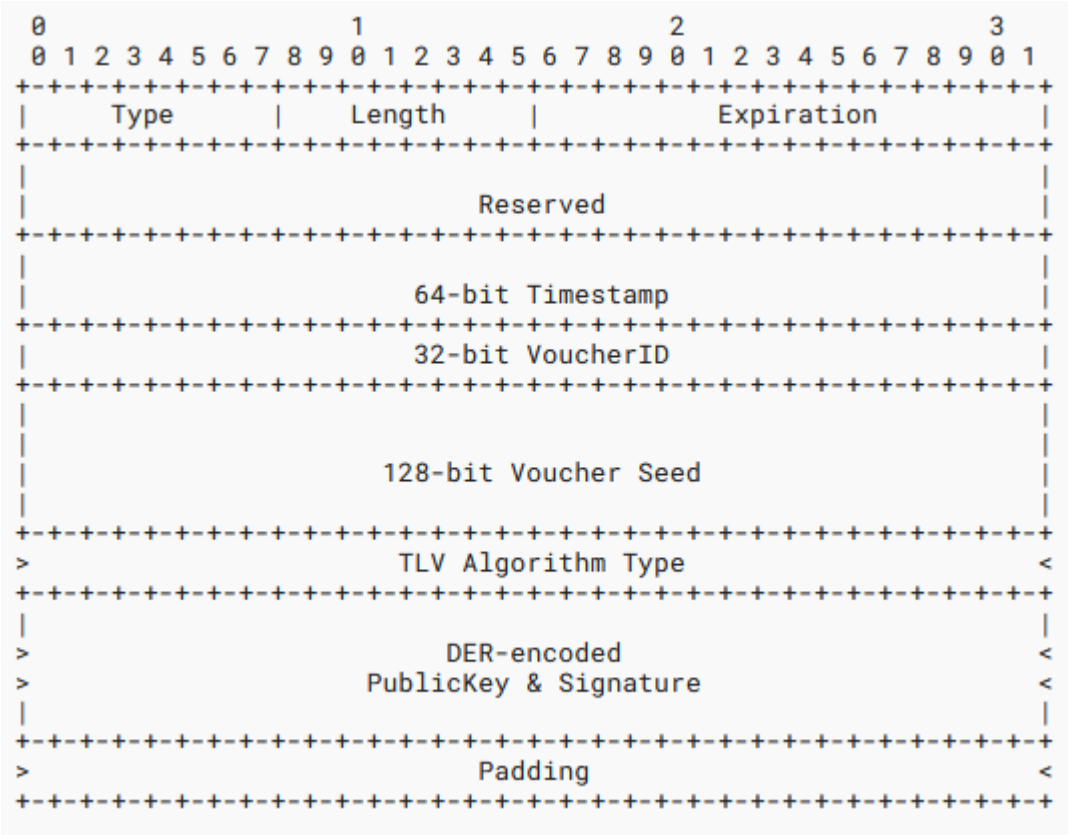


Figure 9. The binary structure of the Link Voucher NDP option; only considered valid by receivers when attached to Router Advertisement and Redirect packets.

Figure 9 shows the 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 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 initialized to 0 by senders and ignored by receivers.

Timestamp

A 64-bit value representing the local system time of the sender at the moment the LV option is constructed in system memory.

VoucherID

A pseudo-random 32-bit value which uniquely identifies an ongoing Link Voucher instance. For the sake of optimization and connectivity, this must not change between distributions of the same unique LV.

Seed

A 128-bit pseudo-random value used as an input in VBA generation on-link. This value is required to be the same for each distribution of an LV identified by a VoucherID. 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

A variable-length field containing a DER-encoded ECDSA [49] public key of type SubjectPublicKeyInfo according to Section 2 of RFC 5480 [50]. 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:

1. The 16-bit Expiration value.
2. The 64-bit Timestamp value.
3. The 32-bit VoucherID value.
4. The 128-bit Seed value.
5. 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 [51]. The signature must be a DER-encoded ASN.1 structure of the type ECDSA-Sig-Value (Section 2.2.3 of RFC 3279 [52]). The final field appears as the two adjacent DER structures from Figure 10.

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 0 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 10. The adjacent DER structure definitions for encoding the ECDSA PublicKey and Signature values of a Link Voucher.

3.8.4 Processing Rules

Voucher Bearers will always respond to Router Solicitation packets with a valid LV if they are the designated and authorized VB deemed to issue LVs on that link. This is true regardless of whether the VB is using a Redirect or Router Advertisement packet to distribute the voucher. Sending nodes wishing to distribute an LV must first check the link for an already-active LV. This entails following a process of router discovery, then only assuming LV responsibility if no LV is already present; it is performed like so:

1. Send a Router Solicitation to the All Routers multicast group at FF02::2.
2. Wait for a response containing an LV for at least two seconds before sending another RS.
3. Repeat this process two more times.
    a. If an LV is received from an RA or Redirect response, accept and use the parameters of the received LV, as long as its fields are valid. This condition

dictates that the sender must not send its own LV, nor should it propagate any instances of received LV options.

b. If no LV is received after the three total attempts, and…
    i. … the sender is not a router: the sender's LV may be distributed on the local link as an option attached to an appropriate Redirect packet.
    ii. … the sender is a router: the sender may attach its LV to an appropriate RA packet.

Senders of LVs must always maintain stateful information about their own LVs, so reliable and consistent vouchers can be sent on-demand. The rotation of stable LV information–the ID, voucher seed value, or algorithm details–are signaled in advance using the LOVMA channel, which will initiate a LV transition window to the new VBA generation parameters. Expiration values must be set to an appropriate value, and senders may adjust this value at-will without requiring a handoff by simply updating it in a subsequent LV transmission. Timestamp values must always be set to the precise local system time as a big-endian 64-bit value. The sender's LV must always be unique and cannot consist of forwarded or duplicated copies of other LVs. Additionally, the voucher's seed value cannot be preserved between different LV ID values or between LV handoffs; it should consistently be a randomly assigned value when first associated with an LV ID value.

For receivers, an LV option appearing with any packet except Router Advertisements or Redirects is always ignored and never processed. Nodes set to the AAD IEM should still regard and cache LVs according to the voucher acquisition rules, in case the interface changes its IEM at a later time. Nodes with static LV details assigned on their interface(s) should fully ignore

received LVs. Nodes acting as authorized VBs are required to disregard any received LV options on the links for which they are already the active, responsible VB. Receiving nodes must statefully maintain and update all LV information per interface, if and only if the received LV is successfully verified according to field rules, its cryptographic signature, and its expiration settings. The most recent, valid, and unexpired version of the LV is what is always cached and preferred on the receiving interface. A received LV that does not contain a valid signature, timestamp, or expiration is required to be ignored.

Nodes acquiring a new LV for the first time become locked to that LV and its public key through an automatic Trust on First Use mechanism, where the public key remains trusted until it provides some instruction to transfer that trust to another party (or until LV expiration). Nodes therefore must not accept LVs which contain any other public key details, or signature fields which do not originate from the saved public key value of the current LV. This period of trust in the public key remains in effect until the cached LV expires or until another LV is elected in a handoff process.

Received LVs which contain different VBA generation parameters (voucher ID, seed value, algorithm settings) must be ignored and cannot update any previously cached LV entries, even if the signature field is valid. Likewise, any difference in the public key value should cause the LV to be ignored, regardless of the LV's other contents or origin. LVs with invalid timestamp fields must also be ignored. Timestamps are considered invalid if the value falls outside the range [CURRENT_TIMESTAMP - Expiration] to [CURRENT_TIMESTAMP + Expiration], where CURRENT_TIMESTAMP is the precise 64-bit, local system time measured by the receiving node. In cases where the precise system time is measured in sub-second intervals like microseconds, the unit of 'seconds' in the Expiration time still applies and must be converted

properly for accurate arithmetic with CURRENT_TIMESTAMP. This timestamping process ensures LV validity remains flexible even with minor clock drifting across nodes on the local network.

Finally for LV receivers: voucher handoff processes allow two LVs to be considered valid and cached simultaneously. When a receiver is subscribed to the LOVMA channel and detects a VHA electing a new LV, it should ignore further LV updates from the previous LV public key or ID association(s). This will ensure on the client side that the previous LV follows through with its commitment to self-expire after its most recent issuance, according to the rules of voucher transitions.

3.8.5 Algorithm Selection

Section 5 of RFC 8018 [43] specifies the definition of a Key Derivation Function: "A key derivation function produces a derived key from a base key and other parameters. In a password-based key derivation function, the base key is a password, and the other parameters are a salt value and an iteration count…" There are three default key derivation algorithms that are to be included with all basic implementations of VBA. Future versions or extensions might wish to add and formalize new KDF algorithms and their corresponding Type identifiers. Any Algorithm Type option not specified in this section or in future addenda to VBA must be ignored by receivers of LVs.

An Algorithm Type choice 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. Receivers will always ignore padding and senders will always initialize padded areas to zero.

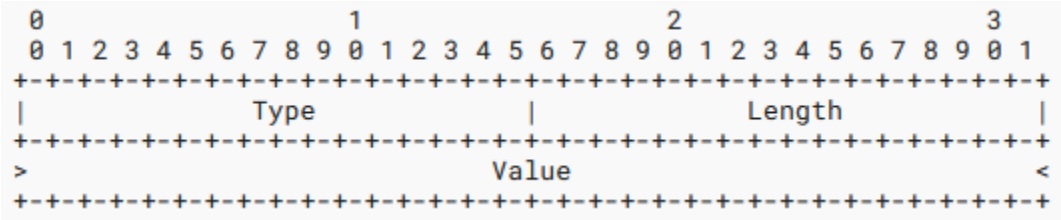


Figure 11. The basic structure of a TLV field that is used in the Algorithm Type field of a Link Voucher, where the Type and Length fields are a maximum width of 16 bits. The angle brackets to the sides of the Value field indicate a variable length field.

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 [43]. 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. VBA explicitly uses PBKDF2 with SHA-256 as its pseudo-random function; therefore, implementations using this Type identifier are required to use SHA-256 as the underlying PBKDF2 pseudo-random function.

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 an LV to amplify the baseline cost of computing the PBKDF2 KDF across the link.

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 [53]. 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. The in-memory data used by the KDF computation is not treated as a confidential item or as a priority. Implementations of VBA should always prefer to employ this Algorithm Type over others in LVs when there is no specific reason to opt for another Type, provided all participating network devices 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 LV 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 which determines 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 LV will be distributed. This value is required to be a minimum of 8 * Parallelism and cannot be set to zero. Receivers would need to adjust the minimum MemorySize value accordingly if the value specified in

the LV does not meet the minimum threshold, according to the requirement, for the actual degree of Parallelism being used.

Scrypt

The Scrypt KDF algorithm is specified in Section 6 of RFC 7914 [54]. 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 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

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

3.9 Voucher Summaries

A Voucher Summary (VS) packet is intended to be an optimization feature of VBA that implementations do not need to recognize or acknowledge. It indicates to receivers of NDP traffic a current LV VoucherID value held in the cache of the sending node when generating the VBA attached to the ND packets. It also tells neighbors which IEM is being used by the sending node's originating interface. This optional optimization is helpful for receivers to preemptively determine whether a reported VBA binding will be able to validate, based on whether the given VoucherID identifies the same present, known, unexpired LV in the receiver's Neighbor Cache.

3.9.1 Option Structure

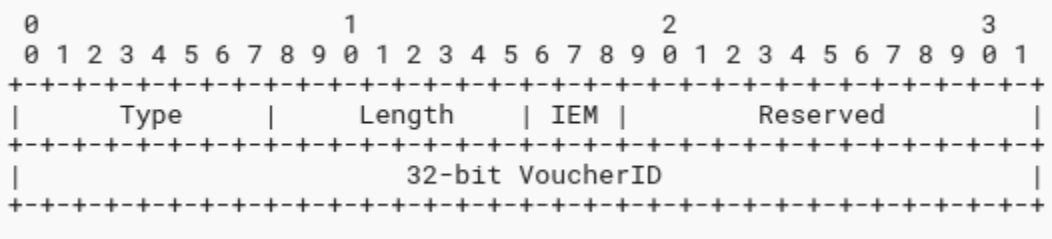


Figure 12. The binary structure of a Voucher Summary ND option. This small stub can provide significant optimizations in busy VBA-enabled LANs.

Figure 12 shows the structure of the Voucher Summary NDP option and its field names. Each field is defined by the sender as such:

Type

The unique NDP Option Type identifier for Voucher Summaries is 64.

Length

Always set to 1. The total length of the VS from the Type through its end, inclusive, in units of 8 octets.

IEM

A 3-bit identifier representing the current Interface Enforcement Mode of the sending interface. Table 1 provides the IEM value-to-identifier mappings to convert the 3 bits into a valid IEM.

Reserved

A 13-bit field reserved for future extensions. This is set to zero and ignored by receivers.

VoucherID

The 32-bit Voucher ID of a valid LV retained on the sending interface, used to seed the generation of the VBA from which NDP traffic is being sent.

Value	IEM
0	AAD
1	AGO
2	AGV
3	AGVL
4	Reserved
5	Reserved
6	Reserved

Table 1. Interface Enforcement Modes are mapped from a simple set of 3-bit values. Reserved mappings indicate possible future extensions to the set of valid IEMs used with VBA.

3.9.2 Processing Rules

Senders are advised to include this option with all Neighbor Solicitations or Neighbor Advertisements. When a valid LV is not currently available on the sending interface, senders are required to set the IEM to AAD (0) and initialize the VoucherID to 0, because no VBA can currently exist on the interface if no LV is retained. The IEM value should be accurately set to the active IEM of the sending interface and the 32-bit VoucherID field is equal to the cached LV Voucher ID field stored for the sending interface’s source VBA.

The VS option is not required and should never become required at any future time. Any ND packets not including it will be processed as they normally would by the receiver's IEM and other normative ND processes. The same is true for VS options which are ignored due to incorrect formatting, or some other validation determination made by receiving VBA implementations. If the receiver's IEM is set to AAD, then the VS option is always ignored by

receivers. Processing of valid VS options is affected by which particular ND packet types they are attached to:

Neighbor Solicitations

The receiving interface will disregard the NS if there is a non-zero VoucherID present and it does not match the Voucher ID of any active LV Voucher ID on the receiving interface. To 'disregard' means to not respond to the NS and to do no further ND processing based on the received packet. If the receiver is aware of an in-progress LV handoff on the link, then both the active LV Voucher ID and the upcoming LV Voucher ID must be considered valid.

Neighbor Advertisements

Processing VS options attached to NA packets affects VBA verification procedures on the receiving interface based on the sender's IEM. Receivers are required to first inspect the IEM field to acquire the active mode of the sending interface. If the IEM indicates an AAD mode at the sender, then VBA is completely disabled at the sender and the received NDP traffic does not originate from a VBA-generated IP address. Thus, when the sender is using the AAD IEM, the receiver should behave according to their own IEM at their receiving interface:

- **AAD**: VBA is not enabled, so the VS option is ignored by the receiver anyways.
- **AGO**: The VS option should be ignored since VBA verification is not being performed on the receiving interface.
- **AGV**: The packet should be dropped immediately because the sender’s IP address is not a VBA.

- **AGVL**: The Neighbor Cache entry should be immediately flagged as Unsecured, and the VBA verification process should be skipped.

If the sender's IEM is not set to AAD and the sender's VoucherID field matches an active LV Voucher ID on the receiver's interface, then processing of the ND packet proceeds to the VBA verification shim as applicable per the receiver's IEM. Again, if the receiver is aware of an in-progress voucher transition on the link, then either the current LV Voucher ID or the upcoming LV Voucher ID are considered valid.

Any other ND packet types

The receiver will entirely ignore the attached VS option.

3.10 Voucher Bearers

The Voucher Bearer (VB) is the on-link node responsible for the current, active, majority-accepted Link Voucher. While any node can be a VB, it is highly recommended that link routers maintain this role–in particular the default gateway of the link–so that proper protections can be configured by network administrators to protect the link from rogue LVs and malicious RA messages.

3.10.1 Appointments

Any willing node can be elected to serve as the link VB, whether by manual configuration or by a process of election and appointment from the current VB on the LOVMA channel. Current link VBs wishing to transfer LV responsibility to another candidate VB must use the LOVMA channel and issue handoffs per the defined election process. Nodes must not be

forced into VB responsibilities without first offering their capability either through their own NDP LV option attachments or through valid and fresh LOVMA channel VCI packets.

If the current VB is not a router or responsible for routing subnet traffic, then it is required to distribute the LV via a Redirect packet with an LV option attached, instead of using an RA packet with the LV option attached. The Redirect packet is expected to conform to the constraints of normative Redirect parameters and processes (Section 8 of RFC 4861 [11]). VBs might wish at any time to let their own LVs expire, even if they do not opt to elect another VB or if there are no other VB candidates available on the LOVMA channel. VBs should not let their own LVs expire without first appointing a responsible successor node, because they are responsible for maintaining link harmony and agreement between interface VBA implementations. If there is no successor node, then the most recent LV will remain current in the network until another node assumes VB responsibility, according to the LV expiration and acquisition process, even if the VB responsible for that LV is not issuing any more advertisements.

3.10.2 Router Advertisement Guarding

Fake or malicious LVs can be problematic for nodes that do not yet have the state of a valid LV stored for their interface(s). When first joining a local network, the connecting interface–assuming it is using SLAAC–searches for a default router to use as a gateway to external networks, seeks prefixes, and enables VBA generation on the interface if an LV is received. Undesired or ‘bogus’ Router Advertisement and Redirect packets are a known problem in local IPv6 networks with no active mitigations or protections, causing the potential failure and hijacking of neighbor traffic. Securing RA-Guard on infrastructure slightly complicates the

proposed simplicity of VBA, but since RA-Guard has no direct interaction with the VBA software implementations themselves, it is a transparent and fully decoupled change that an administrator can maintain separately. An excerpt from the RA-Guard specification in RFC 6105 [44] is noted that explicitly mentions compatibility with SEND:

> "RFC 6105 describes a solution framework for the rogue-RA problem (RFC 6104 [55]) where network segments are designed around a single L2-switching device or a set of L2-switching devices capable of identifying invalid RAs and blocking them. The solutions developed within this framework can span the spectrum from basic (where the port of the L2 device is statically instructed to forward or not to forward RAs received from the connected device) to advanced (where a criterion is used by the L2 device to dynamically validate or invalidate a received RA, this criterion can even be based on SEND mechanisms)."

Due to its inherent flexibility according to this quote, the RA-Guard system should ideally be augmented and deployed with VBA awareness, capable of tracking the state of LVs and LOVMA channel VB elections. This will allow an intermediate network device such as a switch–or fleet thereof–to only require an active RA-Guard Learning Mode for a short initial period and then to subsequently 'follow' the correct LV around, similar to what other network nodes are already instructed to do. The difference with RA-Guard in this scenario is to restrict the forwarding of frames containing encapsulated RA or Redirect packets when and where appropriate, based on what the system already understands about the state of LVs on the network. One notable exception to this, however, is that an RA-Guard implementation might drop its protections if and only if the most recent and legitimate LV has expired (i.e., timed out)

without a successor VB, because some responsible VB needs to be able to supersede an expired LV.

In the case where the elected VB is not a link router nor is responsible for routing traffic, and Redirect packets are being used with the LV option, a Redirect-aware flavor of RA-Guard is strongly recommended to also expect these in its learning processes. Use of RA-Guard is primarily suggested for networks with a revolving door of endpoints, such as public networks, or networks which need to fortify and guarantee their security posture using key infrastructure devices. Appointments or elections of new VBs should be considered with caution because the lack of Public Key Infrastructure certification or authority introduces a problem with bogus claims of initial identity. The exact deployment of RA-Guard is beyond the scope of this research, but it is strongly recommended where possible in order to ensure VBAs faithfully serve their purpose to protect nodes from malicious Neighbor Redirection attacks during the NDAR process.

3.11 The LOVMA Channel

The Local On-link Voucher Multicast Address (LOVMA) channel is defined for the explicit purpose of sharing non-essential, independent, VBA-related details between participating nodes. All nodes with VBA awareness, regardless of their IEM settings, are strongly recommended to join this group, though participating (including subscribing to the multicast group) is entirely optional. Nodes are not required nor expected in any case to make practical use of any LOVMA channel traffic. However, appointed link VBs are always required to join the LOVMA channel.

This multicast group is conveyed over the IP address FF02::ABBA. A helpful mnemonic to remember this address is to think of ABBA as the closest possible hexadecimal rendition of the words "a VBA". The designated UDP port on which all LOVMA channel data is sent and received is 2196, with no exceptions. Senders of LOVMA channel data are strongly recommended to send from a link-local VBA bound to the interface being used to communicate on the LOVMA channel, unless they are using the AAD IEM (in which case the sending address can be any link-local IP).

3.11.1 Constraints

When utilizing the LOVMA channel for any purpose, experimental or deployed, implementations are required to regard these constraints:

- LOVMA traffic is always considered unidirectional. Nodes should not send unicast responses in reply to any received multicast traffic. This recommended constraint chiefly acts to prevent asymmetric traffic volumes from creating traffic amplification denial of service attacks through an abuse of the LOVMA channel.
- All LOVMA channel datagrams are required to be User Datagram Protocol (UDP) (defined in RFC 768 [56]) packets only.
- VBA-enabled nodes must not assume that any other VBA-enabled nodes on the local network are subscribed to the LOVMA multicast group or receiving any of its related datagrams. However, nodes may safely assume the presence of their active link VB on the LOVMA channel.
- Subscribing nodes cannot offer any trust of LOVMA channel packets, unless some datagram verification procedure is explicitly declared in the specification of the unique

UDP packet structure (e.g., by using packet signatures or some other cryptographic mechanism).

3.11.2 Commonalities of LOVMA Datagrams

This section outlines and formalizes some commonalities between UDP datagrams which should be expected to appear on LOVMA channel at any time. All packets traversing the multicast group must use a Type-Length-Value (TLV) format, as shown in Figure 13. Type is an 8-bit integer identifying the datagram type, Length is an 8-bit integer specifying the length of the packet–including the Type and Length fields–in units of 4 octets, and Value is the following object to be expected and parsed by receivers. The Type or Length fields cannot be set to 0, and any received UDP datagrams that are received as such will be ignored by receivers.

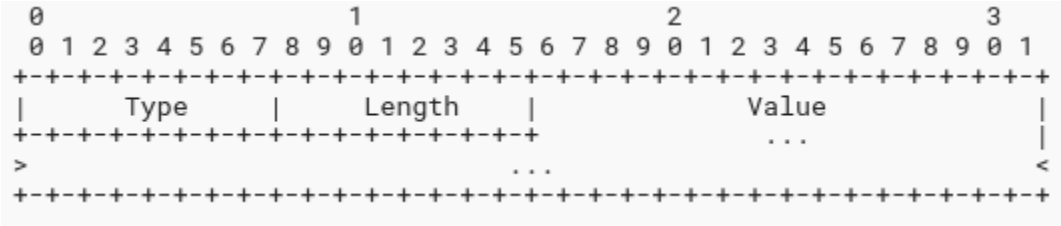


Figure 13. The Type-Length-Value structure specific to LOVMA channel UDP packets. The Value field is always a variable-length field, but it must always round up in size to the nearest 4-octet boundary. All packets traversing the LOVMA group will have this base structure in common.

3.11.3 Voucher Status Reports

A node might opt to occasionally send Voucher Status Reports (VSRs) to the LOVMA channel to gratuitously let other nodes know of its presence as a VBA-supporting interface, in

addition to information about its VBA-related state. Sending interfaces are required to add their sending interface's LLID onto VSR packets, allowing receivers to easily identify the sending interface by LLID. This is as opposed to using the IP Source Address of the datagram to associate the sender with one of its potentially many assigned IP addresses. Receivers are not required in any manner to parse any of the fields presented in VSRs. Receivers may also confirm the IP Source Address and LLID by running the VBA verification procedure to confirm the validity of the resulting binding, but it is not necessary and therefore not recommended due to its potential costs.

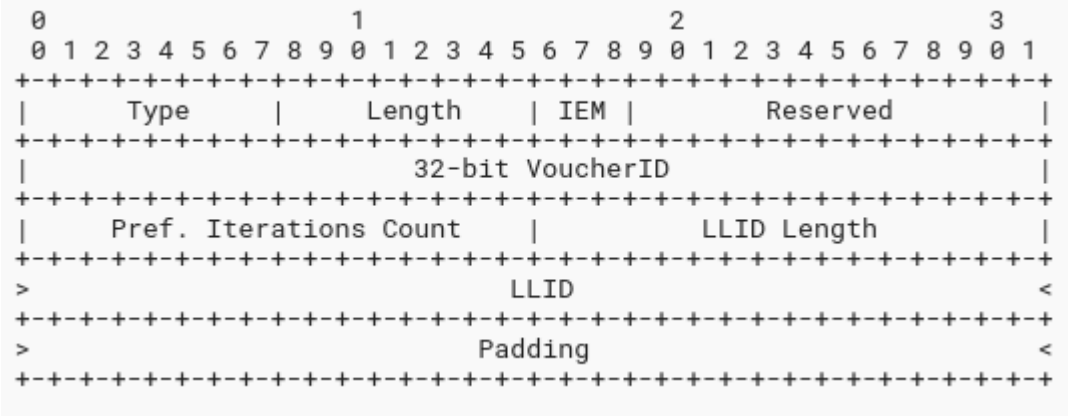


Figure 14. The binary structure and field names of the VSR UDP packet for the LOVMA channel.

Figure 14 shows the structure and field names of the VSR packet. Its fields are as follows:

Type

VSRs are always identified by an ID of 1.

Length

The variable length of the datagram in its totality, in units of 4 octets.

IEM

A 3-bit value identifying the IEM of the sending interface. See Table 1 for a mapping of all possible IEM values to use in this field. This value is required to be set to the AAD IEM if the sending interface does not have a currently cached, active, valid LV.

Reserved

Space reserved for future use. This field is set to zero and ignored by receivers.

VoucherID

The ID of the cached, active, valid LV that is stored on the sending interface. Senders are required to initialize this field to zero if no LV is present or if their sending interface is operating in the AAD IEM.

Preferred Iterations Count

Senders can use this field to indicate a preferred IC value used consistently when generating VBAs for each on-link prefix at the sending interface. For optimization purposes, receivers can associate this field with the provided LLID field’s value to preemptively calculate new VBAs for the sender when the active LV changes.

LLID Length

The length in bytes of the LLID field. Stored as a big-endian value.

LLID

A variable-length field containing the sending interface's LLID.

Padding

Any extra padding set on the datagram to round its total length to an even 4-octet boundary. This field is set to zero and ignored by receivers.

3.11.4 Voucher Capability Indications

A node might wish to notify the LOVMA channel about its potential candidacy as a link Voucher Bearer by sending a VCI datagram. The VCI is an informational packet required to be dispatched in order to be considered for election by the current VB. Receivers are typically intended to be the current VB, but any node can make use of VCI details upon receipt. Nodes must not consider VCI packets as valid LVs in any way, however.

The current VB may maintain a state of unexpired VCI packets, especially when it intends to elect a new node responsible for the LV in an automated process. Current VBs are not permitted to elect a new VB without first receiving a VCI datagram from the candidate indicating its readiness to be elected. Sending nodes also must not assume that the issuance of a VCI packet is guaranteed to lead to their eventual election as a link VB; the decision is ultimately rendered by the current VB. Such election by the current VB must be indicated by receipt of a signed VHA datagram from the LOVMA channel, whose signature verifies with a corresponding public key matches that of the current, active LV.

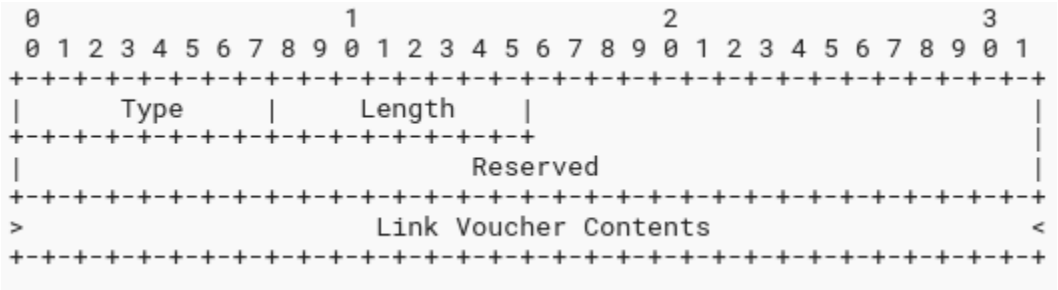


Figure 15. The binary structure and field names of the VCI UDP packet for the LOVMA channel.

Figure 15 shows the structure and field names of the VCI packet. Its fields are as follows:

Type

VCIs are always identified by an ID of 2.

Length

The variable length of the datagram in its totality, in units of 4 octets.

Reserved

Space reserved for future use. This field is set to zero and ignored by receivers.

Link Voucher Contents

The entirety of the ND Link Voucher option to be attached to future RAs or Redirects, including its ND option Type and Length values. The validation of this field follows the same instructions outlined by the processing rules for LV options. Receivers must not expect the Signature or PublicKey of the LV option to be the same as that of the current LV, because this packet type is only announcing a node's candidacy for future election, and it is not attempting to declare a new LV as a VB. Receivers must also ignore the entire VCI if validation of the embedded LV fails for any reason, including invalid cryptographic signatures, inappropriate field values, etc.

3.11.5 Voucher Handoff Advertisements

The current VB may at any time elect a new VB using the VHA datagram on the LOVMA channel. This handoff initiation notifies subscribing VBA-enabled nodes of a tentative change in the current VB and implies that a different public key will be used to sign the new LV. The VHA datagram can also be used to change typically immutable LV fields for the same sending VB, such as when updating the algorithm parameters. If the cryptographic signature on

the VHA is valid according to the current VB's known public key value, listening nodes will accept the start of the handoff process (i.e., voucher transition window). Thus, both VoucherID fields (from the current, active LV and the upcoming LV) become temporarily valid for the link. If the Signature field is not verifiable using the current LV's public key, then receivers must ignore the VHA packet. If there is no current LV and a VHA is received, then it must also be ignored.

The voucher transition window duration is based on the Expiration field from the current VB's LV at the time the VHA is received by neighbors. Exceedingly long Expiration fields entail exceedingly long voucher transition windows, and there is no limit to the duration of a handoff maneuver. The VHA retransmission frequency is variable but is recommended to follow the same cadence as the node's previous RA or Redirect issuances. VBs initiating a voucher transition are required to send at least one VHA notification every 5 seconds for a minimum of 3 minutes, or the length of the current LV Expiration field, whichever is shorter. If the Expiration field value is high, then nodes handing off VB responsibility can choose to stop transmitting VHAs after this minimum threshold has elapsed.

Candidate nodes considered for VB election are required to be gathered (1) from manually configured parameters on the VB device or (2) from a pool of senders of recent, unexpired VCI notifications on the LOVMA channel. When the elected node observes the VHA packet granting it VB responsibility, it must begin sending gratuitous RAs or Redirects to the link for which it is now a responsible VB. The new VB becomes responsible for sending an RA to the local link following each receipt of a valid, unexpired VHA from the previous VB. This creates an echo effect where the repeated issuances of the VHA can immediately make neighbors aware of the LV to which they are transitioning. After 2 minutes has elapsed from the time of

VB election, the new VB must consider the LV parameters (including the public key value) of the previous VB as invalid, and therefore will not trigger any more RAs driven by receipt of VHAs from the previous VB.

VHAs are also required to be used for indicating a change in active LV details using the Refresh bit. This indicates that the handoff consists of a transition between LV parameters from the same VB rather than a change of responsible VBs, so no varying public key information is accepted when the Refresh bit is set. Using the VHA for this purpose affords neighbors enough time to fully transition addresses between varying LV parameters, just like in an ordinary election. For example, if a current VB wishes to change the baseline difficulty setting for the on-link KDF function, it must use the VHA to transition the current LV state into the new one using this process on the LOVMA channel. Such a process is required because changing KDF settings will alter the validity and composition of all VBAs on the link.

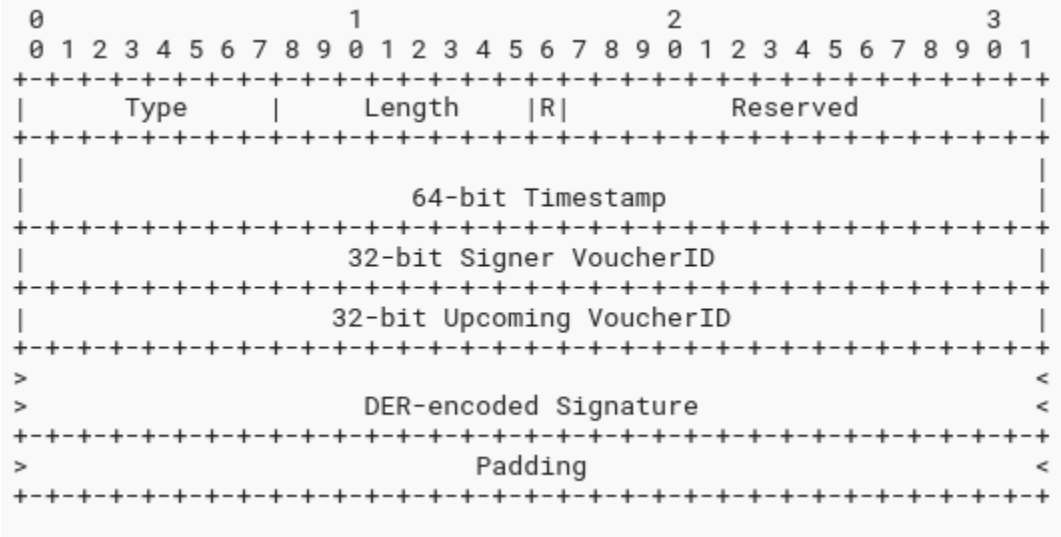


Figure 16. The binary structure and field names of the VHA UDP packet for the LOVMA channel.

Figure 16 shows the structure and field names of the VHA packet. Its fields are as follows:

Type

VHAs are always identified by an ID of 3.

Length

The variable length of the datagram in its totality, in units of 4 octets.

R (Refresh)

A single bit that, when set, indicates the voucher transition is only an LV refresh and does not represent a change of the link VB to a different node. This field should be used to establish new LV parameters from the same VB, such as changing the current seed value.

Reserved

Space reserved for future use. This field is set to zero and is ignored by receivers.

Timestamp

The current precise, local system time encoded as a 64-bit value. Timestamps must be considered invalid if the value falls outside the range given by [CURRENT_TIMESTAMP - 120] to [CURRENT_TIMESTAMP + 120], where CURRENT_TIMESTAMP is the precise 64-bit system time measured by the receiving node and 120 is in units of seconds. If the CURRENT_TIMESTAMP is measured in sub-second units like microseconds, then the 120 value is required to be converted to a proportionate value. This requirement ensures timestamp validity remains flexible despite the possibility of minor clock-drifting across the local network.

Signer VoucherID

The VoucherID of the active LV which is signaling the handoff to the Upcoming VoucherID. Nodes recognizing this VoucherID for their active LV are required upon

receiving this VHA datagram to disregard any further advertised LVs–valid or not–with this VoucherID. A receiver must also ignore this packet if the Signer VoucherID does not identify any active LV held in the cache for its receiving interface.

Upcoming VoucherID

The VoucherID of the upcoming LV which will be assuming active status on the network after the voucher transition window fully elapses.

ECDSA Signature

A variable-length field containing a DER-encoded ECDSA [49] signature, derived using the private key corresponding to the public key of the LV identified by the Signer VoucherID. The signature is computed over a series of sequential octets, constructed in the following order:

- The 1-bit Refresh field as a 1-byte integer, where a value of '1' equates to the bit being set and '0' equates to the bit being unset.
- The 64-bit Timestamp field.
- The 32-bit Signer VoucherID field.
- The 32-bit Upcoming VoucherID field.

The algorithm used in signature computation is ecdsa-with-SHA256, as defined in Section 3.2 of RFC 5758 [51]. This field is required to be a DER-encoded ASN.1 structure of the type ECDSA-Sig-Value (see Section 2.2.3 of RFC 3279 [52]). The implied public key value used to verify the signature must be equal to the public key of the active LV on the receiving interface. If the current LV's public key cannot validate the Signature field, then the VHA must be considered fraudulent and be subsequently ignored.

Padding

Any padding necessary to round the packet size up to the nearest 32-bit (4-octet) boundary. This field is always set to zero and is ignored by receivers.

3.12 Optimizations

This section briefly summarizes the different optimizations built into the conceptualization of Voucher-Based Addressing. Optimizations are crucial because any addition or modification to a protocol like NDP that is so thoroughly ubiquitous can have grave or reaching consequences for adoption if its costs are too high. VBA should include every mechanism possible to ensure its performance is commensurate with its simplicity.

- **Avoiding Repeat Verifications.** Neighbor Discovery NUD features are used to avoid continuous reverification of active neighbors between unique Link Voucher instances. Neighbors do not need to be reverified when there has been no change to their pre-verified bindings from a prior NDAR exchange.
- **Duplicate Address Detection.** The SLAAC DAD process is optimized to reduce the burden of regenerating alternative VBAs from scratch when collisions are detected.
- **The LOVMA Channel & Preemption.** The LOVMA multicast group affords various VBA optimizations: it enables the use of gratuitous handoffs for neighbors to detect upcoming LV changes; it allows hosts to note their Preferred Iteration Count values for upcoming VBA generations, new link prefixes, and LV transitions; and lastly, it permits nodes to become candidates for VB election when the current VB will no longer maintain responsibility for the LV.

- **Key Derivation Function Selections.** The various options for KDF algorithms and their parameters permit issuers of LVs the flexibility to dictate a baseline difficulty setting for VBA generation and verification on the local network. From this baseline, which implementations might choose either from default settings or from other details gathered about the link, nodes are permitted to scale the computational difficulties of each VBA they generate based on their locally selected IC values for each generated VBA.
- **Voucher Summary Options.** Voucher Summary options allow nodes to exchange identifying information about the LVs used to construct or verify the target address related to the current NDAR exchange. Doing so saves wasted computation time if the two hosts disagree on the ID or parameters of the active–possibly upcoming–LV. Behavior with this option is subject to the active IEMs of the two communicating interfaces during this exchange.

3.13 Transition Considerations

It is unrealistic to assume that VBA would be deployed simultaneously across all nodes in even tiny local subnets, because not every active node will receive compatibility at the same time. There will almost certainly be network devices present which have no support for VBA. It is also certain that–like IPv6 itself–some hardware vendors and software developers will never implement compatibility with VBA or provide necessary operable 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) simplistic processes that do not require complex

interactions between neighbors. This capability grants VBA significantly more opportunities for adoption than other protocols like SEND that seek to modify NDP and impose mandatory infrastructure.

3.13.1 Dual-Stack Communications

A pure IPv6 local network using the AGV IEM across its nodes will simply not be able to communicate bidirectionally with 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 due to the gateway's binding verification requirement. In the case of dual-stack local networks, IPv4 traffic can be used as an insecure (i.e., spoof-able) 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 [57] 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 non-supporting 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.

3.13.2 Adjusting IEMs

Local IEMs can be adjusted on nodes communicating directly with non-VBA neighbors to better accommodate their 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 Secured responses. In the case of a subnet router in a mixed network–that is, a local network consisting of devices with mixed VBA support–using the AGVL IEM can again prove very advantageous for the sake of accommodation. Assuming most nodes use VBA and a few cannot, only those few nodes remain at risk of Neighbor Redirection attacks.

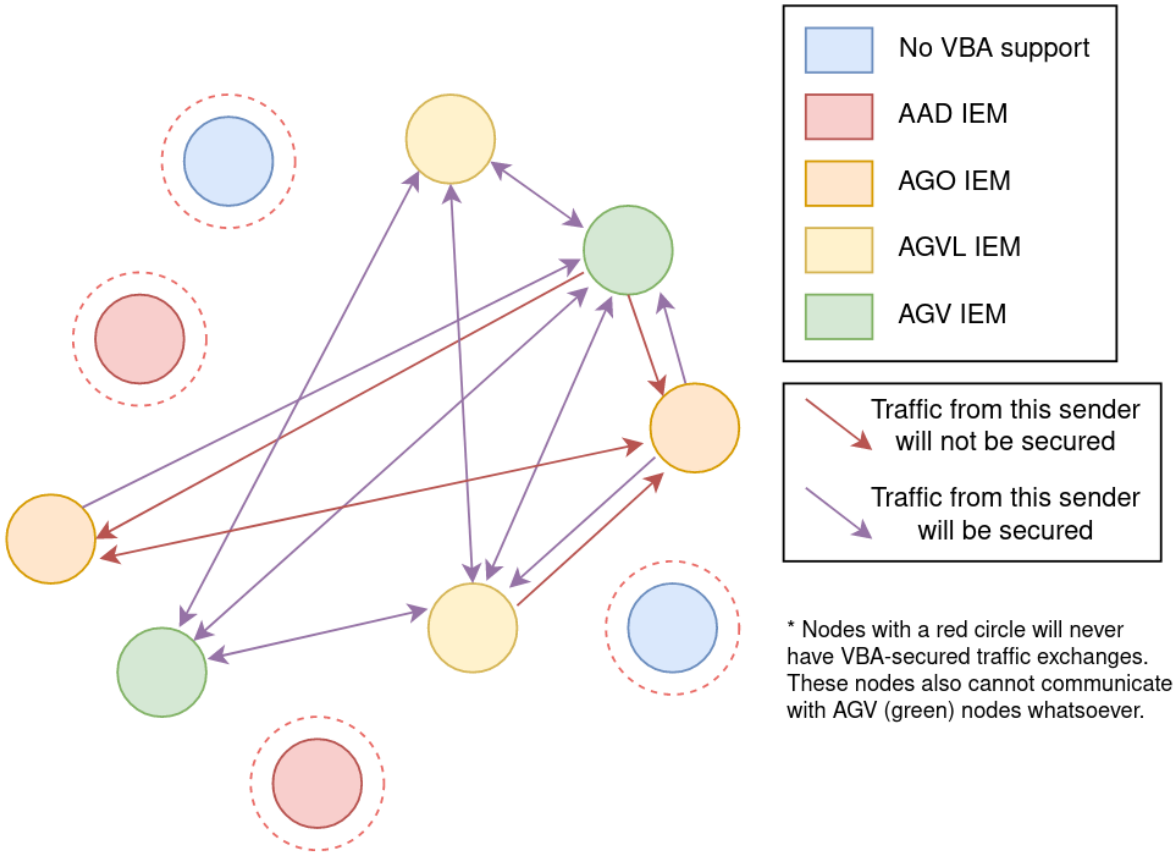


Figure 17. A mixed local network is shown where a single valid Link Voucher is delegated to VBA-capable neighbors. Neighbors without VBA capabilities are shown in blue, VBA-aware neighbors are shown in red, orange, yellow, and green for the IEMs AAD, AGO, AGVL, and AGV respectively. Different links are shown between some hosts to indicate their connectivity and security within this transitioning network.

3.14 Security Considerations

Since VBA is an inherently security- and privacy-focused amendment to the way Neighbor Discovery functions, careful consideration of each security issue is merited. This subsection will briefly discuss a few key points which deserve attention, but it is not all-inclusive of the possible issues that VBA could encounter in practice.

3.14.1 Collision Resistance

VBA generation only preserves 48 bits from a resultant hash output from a key derivation function. While a collision is highly unlikely, nodes process SLAAC self-assignment as they do with the normative DAD process. Even if a collision is improbable, its possibility requires that a reaction to it inevitably occurring must be defined. Potential hash collisions expose a weakness of VBA because LL2IP binding is done through a deterministic hashing process and nothing else; in other words, there is no other mechanism used for certifying the resultant IP addresses. Thus, any other spoof-able LLID on the same link, in combination with a fixed iterations count, producing the same 48-bit H portion of the VBA suffix will result in an equally valid VBA according to the verification procedure.

The employment of cryptographic KDFs drastically reduces the capacity for attackers to discover address collisions and to use them for malicious purposes (like on-path attacks). This brute-force resistance is a consequence of each KDF intentionally requiring more time than traditional hashing to compute. Section 3 of RFC 8018 [43] defines a KDF in the exact manner this specification intends to apply it:

> “Another approach to password-based cryptography is to construct key derivation techniques that are relatively expensive, thereby increasing the cost of exhaustive search.

> One way to do this is to include an iteration count in the key derivation technique, indicating how many times to iterate some underlying function by which keys are derived."

Therefore, KDFs are applied to VBAs for the added purpose of slowing collision discoveries. This same tradeoff of requiring more time for address computation in order to protect against brute-force enumeration is a strategy also recommended for use in password storage systems to protect user secrets [58]. To prevent any possible time-memory tradeoff attacks, the LV is distributed between nodes to ensure that input parameters generating VBAs are always generously salted by a 128-bit pseudo-random value, as well as the subnet prefix, so they can never be pared down to a simple dictionary attack. Additionally, leaving LV parameters to some external controlling factor allows a maximization of the space used for the KDF hash within the resultant VBA.

The seed value dictated by the active LV is also intended to rotate occasionally to prevent long-term attacks. An attacker searching for inputs producing a colliding address is therefore subjected to the misery of enumerating many different link-layer addresses in order to generate a suitable IP address that matches the target's 48-bit hash suffix. This spoofed IP address must also embed the same iterations count as its target because it needs to be an equivalent IP address, thus reducing the attack surface even further. If the target IP address contains a high iterations-count value, then the duration of the collision searching process will be compounded even further. All the while, these collision-producing inputs must be obtained before the rotation of the current LV, which will reset the hypothetical attacker's marathon entirely.

LVs also allow the specification of shared KDF algorithms and baseline address generation difficulties on-link, permitting a dynamic adjustment of the base computation time required to derive and verify all local VBAs. For example, adjusting computation time to be approximately 20 milliseconds per address for the least powerful node, and an estimated 1 millisecond for the fastest, produces a negligible delay in processing legitimate ND messages. Simultaneously, this hypothetical time taken to compute each VBA hamstrings any node, regardless of computing power, from being able to compute collisions expeditiously. So, responsiveness is prioritized for legitimate nodes while also protecting their claimed address ownership. Continuing the above example, hypothetical 1-millisecond VBA generation times for the most powerful nodes still equates to attempting only 1,000 spoofed LLIDs per second (or 3,600,000 LLIDs per hour). If the LLID in this case were an IEEE 802 MAC address, a set of 3.6 million attempted MAC addresses is equivalent to only about a millionth of a percent of the total address space (set to 2^{48} when not accounting for reserved MAC address ranges).

3.14.2 Computational Fairness

The selection of an appropriate key derivation function and its associated baseline parameters is essential to properly scale the difficulty of discovering hash collisions on the local network. The choice of KDF is also essential for determining the fairness of cost in computing the generated address. A network having widely varying computing power across nodes will cause widely disparate VBA computation times when using CPU-bound KDFs instead of using memory-bound KDFs for address calculations [59]. Even when using memory-bound KDFs like Argon2d, the proper delegation of baseline algorithm parameters in the LV should always tend toward being more forgiving for systems with fewer computing resources. The balance of low

computation latencies with high security might be difficult to determine, but implementations might attempt to discover and apply defaults that achieve this goal as universally as possible.

3.14.3 Hijacking or Desynchronizing Link Vouchers

Theft of the VB role can be achieved by a few different means based on the level of security employed in the local network. Without RA-Guard, false VBs are free to limitlessly advertise their own rogue LVs to other nodes. For nodes already on the network with an acquired, active LV, this is only a problem if VHA packets in the LOVMA channel are not being used to initiate voucher transitions and the current LV expires. For nodes joining the network for the first time, there is a timing opportunity for an abuse of automatic Trust on First Use: illegitimate VBs can ‘capture’ unsuspecting joiners of the network by racing to be the first provider of a ‘valid’ LV.

If the legitimate VB goes offline, has large gaps between LV transitions, or is not able to transmit any updated LVs to the network, the current LV can time out. When an LV expires, the process of VBA LV acquisition requires nodes to accept any incoming LV as providing further direction and consensus for link neighbors. If a malicious node uses other denial of service techniques to force the current VB offline for long enough, then the malicious node can force an expiration of the current LV and gain control of it on the entire local network. Another much less feasible attack might involve a theft of the cryptographic private key associated with the current LV. Any compromise of the LV key will result in directly impersonable LVs or VHAs which would remain unquestioned by any VBA-enabled neighbors.

Relatively short expiration windows for LVs should remain disallowed for LVs because of (1) possible time synchronization issues between neighbors, (2) 'address storm' prevention,

and (3) compensation or grace for sluggish VBs that somehow cannot send LVs to the link fast enough before a timeout event. Most relevant to this section are items 2 and 3. The possibility of an 'address storm' is prevented by relying on this mechanism: malicious VBs cannot over-rotate the current LV and completely exhaust link nodes, who will be very busy trying to keep up with VBA generation and verification processes. And compensating for a slow VB with longer expiration windows requires malicious nodes to force the legitimate VB off the link for longer to mutiny it as the link VB.

Hijacking, tampering with, or otherwise desynchronizing the LV can be used for either malicious denial of service attacks or to set the difficulty of VBA computation to a very low threshold. There are a few different ways this process could be abused to take advantage of gaps in the protocol:

- Denial of service attacks could result from setting LV parameters to an excessive difficulty. By asking local nodes to verify and generate VBAs according to absurd KDF settings–even for lower iteration counts chosen on each node–outrageous amounts of computing power could be wasted or withheld from being applied elsewhere. This could potentially consume enough resources on a neighbor to disconnect it from the network entirely.
- Consider a situation where Group_A represents hosts aware of legitimate LV_A and Group_B represents hosts aware of malicious LV_B. Having multiple LVs active on the same link will inevitably lead to different logical subnetworks, where Group_A hosts are generating and verifying VBAs according to a completely different LV than Group_B. Depending on per-interface IEMs, hosts from one group will be completely barred from communicating with hosts in another.

- A malicious VB could transmit an LV dictating use of a KDF algorithm with very minimal requirements. For example, using PBKDF2_SHA256 with an ITERATIONS_FACTOR of 1. Targeting hosts with low IC values would of course be most efficient for discovering a valid and spoofed LLID that produces an address collision. Undermining the entire subnet by controlling and loosening the baseline difficulty of VBA generation affords the attacker a greater advantage by greatly reducing the computation costs of on-path attacks.

All the concerns of abuse from this section allude to the importance of guarding the local link from rogue LV options in the first place. Though on-path attacks are still less feasible with VBA enabled–regardless of LV control–the risks assessed above remain and are not outweighed. An enhanced RA-Guard with awareness of VBA is recommended in this situation to protect the network from adversaries attempting to hijack the LV. Other solutions for (1) denial of service attacks disconnecting the current VB from the network, and for (2) the deployment of accurate and VBA-aware intrusion detection systems, are beyond the scope of this research.

3.14.4 Regarding Denial of Service

The goal of VBA is primarily to counter on-path attacks in local networks while maintaining simplicity, flexibility, and privacy. Mitigation of Neighbor Discovery denial of service attacks is therefore an auxiliary goal that could be achieved by applying other protocols and related research. Placing the burden of solving all these problems onto VBA could reduce its flexibility and practicality by forcing it to apply many different mitigation strategies at the same time, rather than leaving them as optional add-ons for a protocol which is already simplistic and

low-cost by design. However, denial-of-service attacks are still a serious concern when employing VBA; even more so because it is an additional layer on top of the default protocol which already has weaknesses. When a denial-of-service topic is presented in this section without a solution, it is strongly implied that the implementation of VBA should find or layer another way to mitigate the problem–or at least maintain an awareness of the weakness during development.

Neighbor Solicitation Flooding

Section 4.3.2 of RFC 3756 [15] outlines an attack targeting last-hop routers that inundates a network with traffic destined to on-link hosts which do not exist. VBAs do not suffer from this attack vector or from any situation involving the repeated creation of Neighbor Solicitation packets, as there is no extra cost incurred in creating them.

When a VBA-enabled node is receiving a flood of NS packets rather than sending them, particularly if the NS packets contain spoofed SLLAOs, then the node may be forced to compute a large volume of VBA verifications in a short interval. This could easily lead to resource exhaustion if the receiving interface's LV parameters specify more difficult baseline KDF settings. A malicious node may also initiate a series of connections from bogus IP addresses that demand return traffic at higher layers of the network stack, such as TCP SYN floods. This would necessitate that the target of the attack engages in an NDAR transaction to determine the LLIDs of the supposed initiating IP addresses, if the LLIDs were not provided in NS SLLAOs. If the bogus initiating IP addresses use high IC values, then the influx in the volume of verification work could quickly exhaust resources on the target.

Neighbor Advertisement Flooding

Neighbor Advertisement floods, either with (1) randomized target addresses and TLLAOs or (2) randomized TLLAOs for a known target address, will not affect VBA networks or the VBA verification process for enforcing interfaces. VBA Neighbor Caching behavior for NAs does not by default permit the presence of an Override flag to affect an NC entry, nor do NAs affect cache entries which have matured beyond the INCOMPLETE state. A more effective attack vector is listed in the previous section (for solicitations). Falsified incoming connections could bait a target node into sending many NS packets, each of which an attacker could reply to with a bogus, high-iteration VBA to verify through the shim process.

Over-rotation of Link Vouchers

Large local networks might have thousands of devices on the same logical link using NDP to resolve each other's LLIDs from IP addresses. When a network is of this size and the LV is handed off to another VB through the election process, optimizations for nodes with fewer resources could get excessively costly when attempting to pre-generate anticipated VBAs according to the new LV parameters. To reduce this burden, implementations can choose to either limit their optimizations at a certain cache size or pre-generate VBAs only for the most recently contacted, high-traffic neighbors on the link.

3.14.5 Static & Anycast Addressing

Networks requiring a mix of ephemeral addresses in parallel with static, stable, long-term addresses will encounter difficulties deploying and maintaining VBAs if bindings are not statically entered into node neighbor caches. Preserving the state of an LV long-term will not be a feasible strategy to maintain stable addresses, as long-term LVs could lend themselves to the malicious discovery of address collisions. Assigning long-term addresses to hosts on a VBA-enabled network can be accomplished using a few 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 which failed to verify. Additionally, it is not necessary to set AGVL on the interfaces with static addresses (unless such interfaces also interact with other on-link, non-VBA static addresses), because IEM affects neighbor verifications and does not impose restrictions on statically-assigned local interface addresses.
- If local nodes simply do not interact with the static addresses, then the only affected parties are the node(s) with the static assignments and the subnet gateway, which will ostensibly route traffic to and from the static address(es). Most RAs will specify a link-local address as the subnet gateway: if this is the case within the subnet, then only router-to-host traffic will fail VBA verification. This is because the router needs to be aware of

the LLID corresponding to the static IP address to forward frames, but the node forwarding to the router can always safely verify the router's binding with its link-local VBA. Therefore, a static entry in the NC of the router should correlate LLID(s) to the static IP address(es) on each node. Doing this for each long-term static IP address will mitigate any potential on-path attacks for both neighbors in the NDAR exchange, while still ensuring all other NDAR transactions verify according to VBA requirements and the levels of strictness configured on each network interface.

- Simply use manual NC entries across the whole subnet wherever interactions with the static IP addresses may be required. The use of manual NC entries may alleviate the requirement for VBAs at all, depending on how and where the related static IP addresses are set in the first place as well as their amount and importance in the network. This approach assumes the LLID of each interface carrying one or more static IP addresses is also stable and unchanging over time.

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 created for on-link prefixes within VBA-enabled networks, because the ownership of the address cannot be bound to a particular LLID. The IPv6 Addressing Architecture specification (RFC 4291 [14]) outlines a Required Anycast Address in Section 2.6.1. VBA-enabled links should maintain compatibility with this requirement by disabling verification for on-link subnet anycast addresses. For example, a host using SLAAC to generate an address in the subnet 2001:db8:700::/64 will disable VBA

expectations and verifications for the address 2001:db8:700::. Because VBA protections must be disabled for this target host, implementations should highly consider avoiding use of the subnet Required Anycast Address altogether wherever possible.

3.14.6 Unsolicited Traffic from Neighbors

VBA disables automatic caching of neighbor address bindings without first following its own address verification processes. Considering this constraint, 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–as is the case for many UDP application services–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 on-path attacks and therefore is not concerned with policing incoming traffic that does not require an NDAR exchange.

Chapter 4 Neighbor Discovery Sessions

Voucher-Based Addressing is vulnerable to identity impersonation attacks where an active LL2IP binding can be hijacked by malicious hosts without needing to search for an address collision. This attack is accomplished by denying connectivity to a target node that owns a desirable LLID, then assuming its LLID in its absence. Methods used to deny connectivity before impersonation could include any denial-of-service technique to which the local network, or the target node itself, are vulnerable to. Once the target node is offline, the malicious node can then assume its LLID and intercept all frames destined for it without needing to perform a classic NDAR redirection attack. This can occur even in VBA networks because the advertised LL2IP binding from the malicious node is considered legitimate.

Networks using SEND can also experience impersonation attacks, but only in the short-term because SEND relies on CGAs to enforce knowledge of a private cryptographic key in ND transactions. The SEND specification itself declares this a known problem in Section 9.1 of RFC 3971 [16]. When a new request for ND arrives, such as during the NUD process, impersonating nodes will not be able to sign responses nor generate a valid RSA Signature option because they do not know the original private key. These concerns have more gravity for techniques akin to VBA than SEND for this very reason: VBA does not demand proof of knowledge of some private value that is publicly verifiable.

To enforce a knowledge requirement that can validate identity without using public-key cryptography, ND Session Options (NDSOs) are used to exchange session details via Zero-Knowledge Proof (ZKP) techniques. These options do not interfere with any VBA verification

processes, and act to further bolster confidence in neighbor verification over a long-term session. ZKP allows a node to express knowledge of a particular password without exposing the password itself; in the case of ND sessions, using a clever hashing construct known as Reverse Hash Chain Revealing (RHCR). A session also allows a neighbor to transition communications to another LLID for the same target IP address while still preserving its proof of identity with ZKP. To summarize, NDSOs reassure two communicating neighbors of one another's persistent identity beyond purported values embedded in network traffic; potential threat actors attempting on-path attacks will not be able to provide continual ZKP in any circumstance. This is the primary motivation for introducing Neighbor Discovery Sessions.

4.1 Terminology

A glossary of terms and acronyms related to Neighbor Discovery sessions is necessary to index, organize, and comprehend the many different aspects of the research. Terms from the Voucher-Based Addressing Terminology subsection in Chapter 3 may also be used throughout this chapter.

- **ICM**: Interface Configuration Mode. One of four different ND Session operating modes specifying the behavior of an interface when sending or receiving ND packets.
- **ZKP**: Zero Knowledge Proof. An authentication methodology that relies on the holder(s) of the password to prove they have knowledge of it, without revealing the password itself.
- **RP**: Root Password. Used to form the RH of a Hash Chain for use in RHCR ZKP. Its value must be pseudo-random each time it is generated and can be discarded after creating the RH.

- **HC**: Hash Chain. A cryptographic primitive consisting of a Root Hash that is iterated and salted a fixed number of times to create a Final Hash. The Final Hash represents the ultimate 'link' in the full chain of intermediate hashes from the Root Hash.
- **RHCR**: Reverse Hash Chain Revealing. A piece-by-piece, backward revelation of components comprising a hash chain. Used by ZKP in maintaining ongoing ND Sessions and proving sender identity.
- **RH**: Root Hash. The very first hash in a HC which is derived from a throwaway pseudo-random initial value. This hash represents the 'password' of the chain and is often used in this specification to establish the next HC for a session when revealed.
- **FH**: Final Hash. The last and publicly known hash in a HC which is given to external parties to store for ZKP validation. This value is never used as a proof of knowledge of the Root Hash and is only used for RHCR verifications.
- **SN**: Session Node. Either one of the two participants engaged in an ND session. When pluralized, this describes both participating nodes in the end-to-end session.
- **SI**: Session Interface. The node being discussed or observed in a session; the perspective or node acting as a point of reference.
- **SP**: Session Peer. The opposite link node that is participating in or establishing an ND session, relative to the perspective node (the Session Interface).
- **SRC**: Session Reachability Confirmation. Simplified ND Reachability Confirmations (see Section 7 of RFC 4861 [11]) consisting of unicast-only NDAR transaction packets with valid, authenticated NDSOs attached to them. These are used to propagate a session and reset the SETs or SITs on SPs receiving them.

- **SIT**: Session Invalidation Timer. A 1-minute timer that initializes upon receipt of an invalid NDSO with an existing SessionID, a New-flagged NDSO whose ND packet contains an IP address matching a current NC entry, or a new session initiation. After this timer elapses, the identified ND session will forcibly expire unless some form of Session Reachability Confirmation is received. Persistent sessions will neither use nor enable the SIT for any reason.
- **SET**: Session Expiration Timer. A timer whose duration in minutes is set by an initialized session's Expiration value upon the initiation and confirmation of a new session. When this timer elapses, the ND session it is linked to is invalidated and destroyed. SRCs will always refresh the duration of this timer.

4.2 Design Overview

Voucher-Based Addressing can enable neighbors to verify each other's identities using both link-layer address binding and the principle of MAC address uniqueness on-link. The latter principle, however, is only applicable to on-link nodes who are still active and who have not been impersonated. If a node disappears, is somehow intercepted, or goes off-link for any reason, then its MAC address becomes again available for the taking. An opportunity like this is an alluring prospect for impersonators which are motivated to intercept traffic that might not be encrypted at higher layers of the network stack. Put another way: a valid VBA is meaningless if the owner of the MAC address is malicious and somehow intercepts frames (including NDP transactions) before they reach the genuine on-link target. With an insecure link layer, some method of authenticating neighbors with one another must be implemented.

Neighbor Discovery Sessions are carried and communicated as an abstraction on top of existing NDP traffic, tasked only with ensuring that the Session Peer (the communicating neighbor) still knows the current password for the given NDP channel. Such an ongoing assertion for proof of knowledge is practical for NDP because its stored Neighbor Cache entries are always temporal; that is, they are bound to a fixed time before they must be refreshed, lest they enter a stale state. Trust on First Use (TOFU) is a concept used in other technologies like PGP [48], finding a practical application in ND sessions; albeit in a more automatic fashion where the 'first use' is trusted automatically.

When a Peer provides either an initiation of a session or an acknowledgement of session initiation, it will always give the initial ZKP parameters used to validate the subsequent ZKP tokens provided during the session. Once the Hash Chain used by ZKP has no more tokens to offer, the Root Password for the peer's Hash Chain is rotated to another value by revealing the current Root Hash. Such usage of chained, one-time proofs of knowledge ensures that each revelation of a new ZKP token provides a way for a neighboring node to continually validate its identity to its neighbors. This authentication mechanism is the driving idea behind any use of ND sessions.

4.2.1 Core Objectives

ND Sessions aim to satisfy a few core goals for Neighbor Discovery:

- **Identification**. Any session established with a peer at the NDP level is guaranteed to be continually legitimate through its duration, extended and prolonged by renewing the session regularly, with the use of Zero Knowledge Proofs.

- **Performance**. A fixed reliance on SHA-256 and very simple concatenate-and-iterate movements lend themselves to drastically improved session performance on participating systems with minimal resources. SHA-256 is perhaps one of the most ubiquitous hashing algorithms, while also being significantly collision-resistant (see Section 4 of [60] for more details).
- **Simplicity**. Avoiding reliance on public-key cryptography or Public-Key Infrastructure is a primary goal of establishing sessions, since performance degradations or extra complexities demanded by these tools have in the past resulted in slow or nonexistent adoptions.
- **Privacy**. Sessions are end-to-end and do not rely on a centralized infrastructure to communicate some kind of registration information or other details. The use of RHCR always ensures the privacy and protection of session Root Password values, while also rendering any packet replay attacks infeasible.
- **Flexibility**. Nodes have the flexibility to determine their own participation in per-interface, per-connection sessions through the use of various Interface Configuration Modes, similar in structure and utility to the IEMs used by VBA.

4.3 Impersonation Protection & Authentication Model

Sessions are authenticated with ZKP by revealing a series of subsequent, one-time hashes in a reverse order from a revolving Root Password (ultimately, a Root Hash); all mentions of a 'hash' with ND sessions imply the use of the SHA-256 algorithm unless otherwise explicitly specified. The requirement of ZKP prevents potential adversaries from both (1) acquiring and using the Root Hash, and (2) inserting themselves into the middle of any ND session that is

already ongoing. Requiring a new ZKP token attached to each NDP packet demands that each participating node must retain knowledge of their current Hash Chain RHs in order to derive the next token, automatically barring nodes unaware of the RH from hijacking the session.

The RH is unique on each Session Node (SN) at a per-session granularity, meaning there are two ZKP processes mirroring each other at each SN for any established session. The recipient of an unseen ZKP token from a Session Peer (SP) must always have sufficient knowledge from a preexisting session with that target node in order to validate the received proof. Performing validation requires the mixture of node-specific parameters (i.e., the interface link-layer address and IP address) which the SP cannot easily change–or rather which, given any change during communication, will invalidate subsequent proofs. In the authentication model, SNs are free to establish a new session from their new LLID or they can transition the current session to it. The latter option is more preferable as it preserves and extends an authentication which is already established.

The Root Password chosen by the session initiator seeds a new session. It is formed into an initial Root Hash (RH) which is then iteratively salted with other parameters by concatenation and hashed again, up to an arbitrary CounterMax times, to form a Hash Chain (HC). The final hash at CounterMax iterations beyond the RH is termed the Final Hash (FH) and the length of the HC is given by CounterMax. The length of the HC is sent with the FH, an expiration value, and a 32-bit session ID value to the peer, who can then use those details, combined with the sender's claimed link-layer address and IP address, to authenticate any subsequently received ZKP tokens. Once a session has been initiated by the sender, the recipient of the initiation will repeat the same process with a completely different RP seed unknown to its peer, after accepting and acknowledging the initiation of the session. In this model, each SN maintains constant

awareness of its own HC and its peer's HC, as well as the most recent ZKP token revelation for each.

4.3.1 Reverse Hash Chain Revealing

ND sessions utilize a two-way mechanism for their chains known as Reverse Hash Chain Revealing (RHCR), first rudimentarily conceptualized by Lamport in 1981 [61] and applied to IPv6 addressing by Nikander [62]. As previously mentioned, the Final Hash is the terminal and public token reached after CounterMax iterations from the Root Hash have been computed. Thanks to the unidirectionality of hashing, revealing any intermediate hash 'further back' in an HC constitutes proof of knowledge of a secret value (i.e., Root Hash) without revealing the secret itself, because the intermediate hash must have been computed somewhere along the iterations sequence between the RH and the FH.

By sending an intermediate hash (or ZKP token) to a peer along with a Counter value, indicating the number of additional iterations on the token required to reach the FH, the peer can validate the proof. Once a value at Counter has been revealed, any submitted value less than or equal to Counter no longer constitutes a valid ZKP and will be rejected if received by the peer. Figure 18 details the functions used to compute each part of a whole per-session Hash Chain on each Session Node.

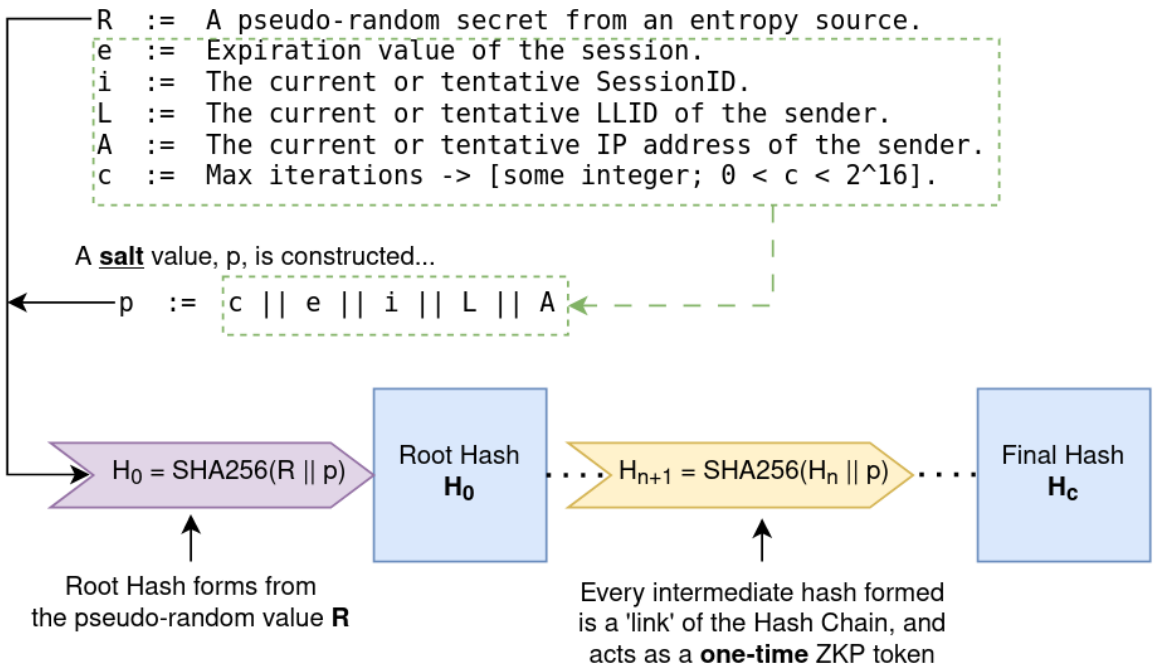


Figure 18. Each Session Node maintains its own internal Hash Chain state, constructed with a simple hash-and-iterate procedure. Once this Hash Chain is prepared, the node is ready to begin providing Zero Knowledge Proofs to the Session Peer.

To more explicitly explain the HC formation procedure given in Figure 18, an understanding of why revelation occurs backward from the public Final Hash H_c is required. Awareness of the Root Hash H_0 enables computation of any subsequent token H_n where $\{0 \leq n \leq c\}$. Receivers who capture some intermediate hash H_x where $\{0 \leq x \leq c\}$ are only able to derive H_y where $\{x \leq y \leq c\}$. Therefore, any newly received token H_z, where $\{z < x\}$, demonstrates knowledge that cannot be known by anyone except those with knowledge of hashes preceding H_x; such proof is a valid assertion for knowledge of the RH and thus proves the sender's identity. Because tokens are derived from one-way hashing, any revelation of H_y implies the receiver's x becomes equal to y, shifting lower the required z value for ZKP from the sender.

The stored state of each HC, whether local or for the SP, must include information about the most recently used Counter value, to track the incoming Counter value for which received ZKP tokens may be considered valid. When a node wishes to provide a ZKP token to its peer, it will increment the local HC's Counter value by 1 and send the result of the RH iterated [CounterMax - Counter] times. When a HC has no more tokens to reveal except its RH–or when the chain has been depleted down to around 10% of the remaining tokens–the Session Node should simultaneously (in the same NDSO) reveal the RH to the peer and establish a new FH to use, with the same CounterMax value.

Using this process, revelation of the RH is equivalent to exposing the password the SN was using for ZKPs with its peer. This is why it is sufficient and requisite information used to establish the next HC in the session. Its revelation is, as suggested, also sufficiently randomized and not predictable in practice, to prevent eavesdroppers from predicting and intercepting it. Session Nodes should never reveal their Root Hashes back-to-back or in any predictable manner, lest interception become more feasible. Figure 19 provides a clearer visualization of the RHCR process.

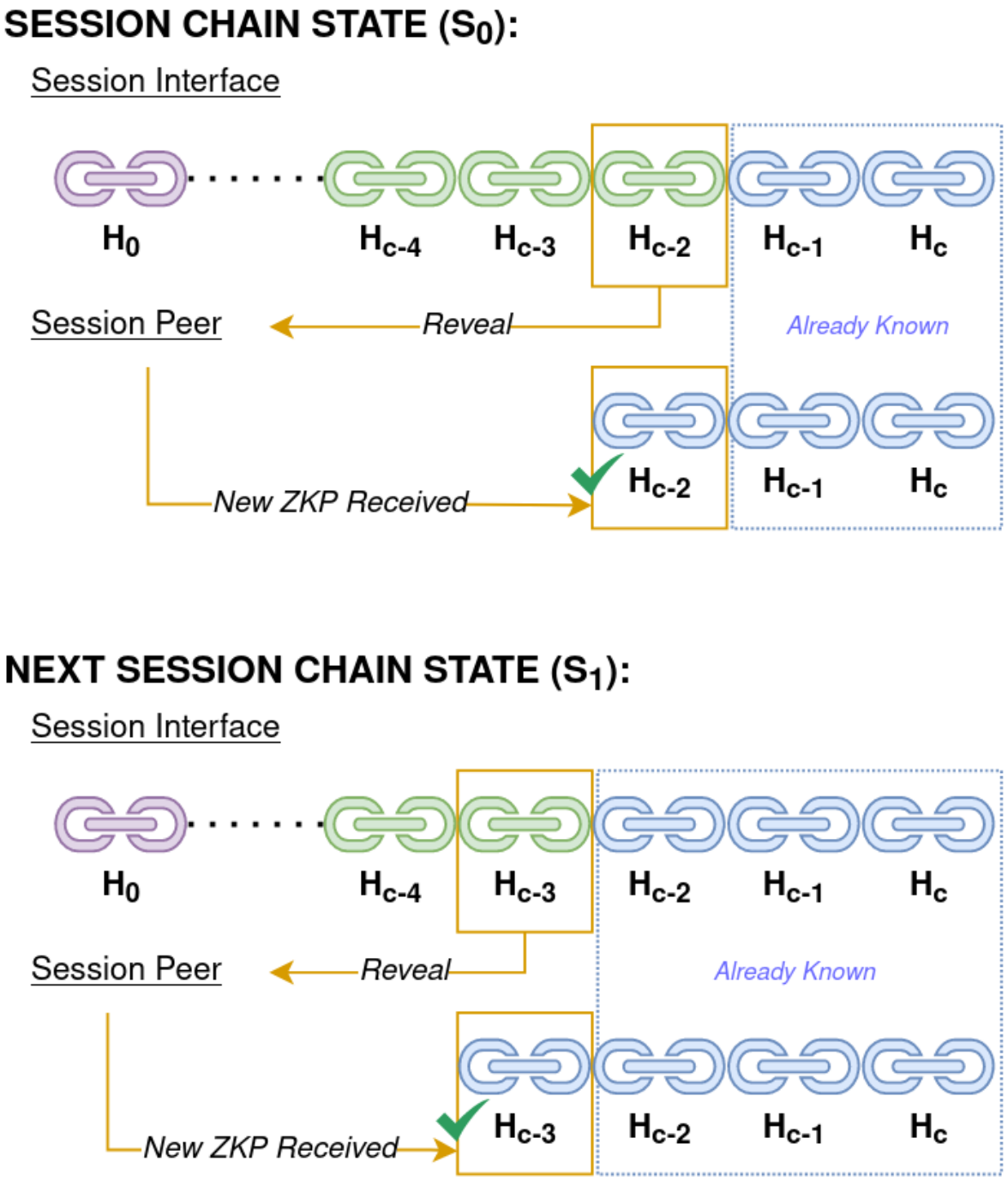


Figure 19. A visual representation of the Reverse Hash Chain Revealing concept used in ND sessions. In state S_0, the intermediate hash (i.e., ZKP token) at H_{c-2} is revealed to the Session Peer. The peer validates this token by running the H_{n+1} function 2 times to get H_c. The next state S_1 shows how the previous ZKP token is now no longer a valid proof, and H_{c-3} further back in the Hash Chain must be revealed to continue session authentication. This ZKP authentication occurs from both Session Node perspectives with each dispatched NDSO.

4.4 Interface Configuration Modes

A set of four Interface Configuration Modes (ICMs) is defined for ND sessions, similar in function to the IEMs of VBA, but operating completely independently of them. Having a few operating modes to choose from will help to foster adoption of NDSOs, especially in mixed networks where not all nodes recognize NDSOs. Interfaces will operate in any one of four modes affecting ND session behaviors, chosen for the interface by the implementation's discretion or by manual configuration:

- Ignore mode. Nodes will not initiate new sessions and will ignore all received NDSOs.
- Gregarious mode. SIs will always attempt session initiation when a session with the target NDAR transaction IP address (i.e., the SP) is not already ongoing. SIs will neither expect nor require SPs to reciprocate with any acknowledgements or to initiate sessions. NDSOs are considered only optional addenda to ND traffic, but when a session is established with a peer they become required for communication with that SP until the session is terminated.
- Gracious mode. SIs will attempt session initiation when no session with the SP is already active. Receivers in this mode will ignore all ND packets which do not contain NDSOs. The Persistent flag within NDSOs will always be ignored.
- Strict mode. SIs will attempt session initiation when no session with the SP is already active. Receivers in this mode will ignore ND packets which do not contain NDSOs. The Persistent flag within NDSOs is enforced when set on any initiation NDSO.

The recommended default ICM for most networks is the Gracious mode, which prevents sessions from locking peers out of communication from one another, either by malice or by

forgetfulness of the SP, by enabling the Session Initiation Timer for all sessions. The Strict mode is more security-focused because SITs can be disabled entirely by setting the Persistent flag in session initiations, thus allowing sessions to be prolonged indefinitely by receiving valid Session Reachability Confirmations. ND session compatibility and features can be disabled on interfaces at any time and for any reason, but doing so will likely disrupt all current sessions that the SN is actively participating in. This might create communication problems across the network due to dropped or misaligned sessions, so it is done with great caution.

4.5 Packet Structure

The Neighbor Discovery Session Option is an addendum to ordinary NDP traffic, carried through the extensible Options format described in the NDP specification. Information in NDSO stubs act as a type of overlay embedded into NDP traffic to carry session details.

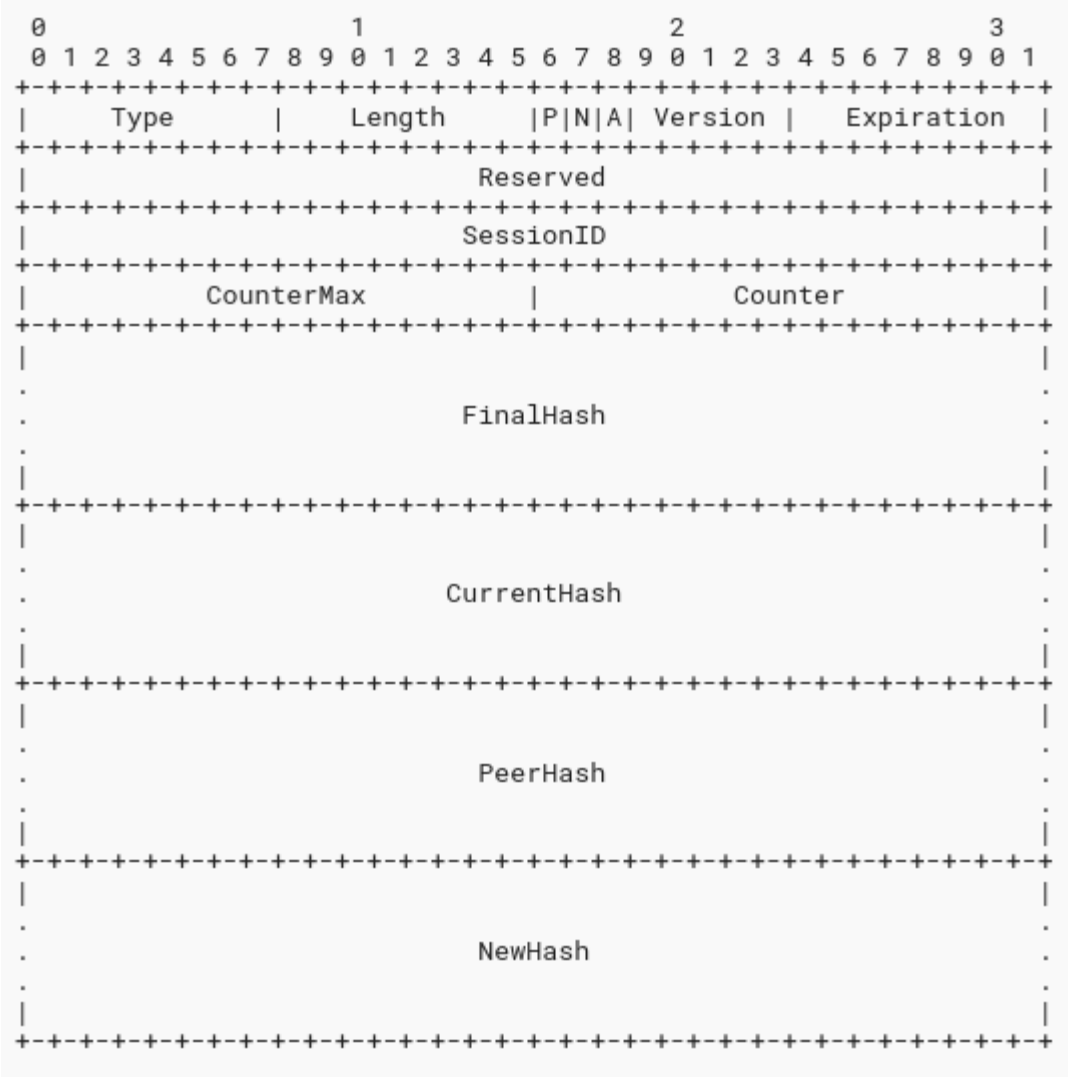


Figure 20. The binary structure of the Neighbor Discovery Session Option. This option should be sent with each NDP packet with neighbors who already have pre-existing sessions active.

Figure 20 shows the structure of the NDSO. Its fields are defined as follows:

Type

The NDP Option Type identifying NDSOs is always 65.

Length

Always set to 18; it is the total length of the NDSO from the Type through its end, inclusive, in units of 8 octets (64 bits).

P (Persistent)

When the NDSO is initiating a new session, this flag indicates that the exchange is intended to establish a long-term, ongoing ND session that cannot be invalidated except by expirations. As this is a high-fidelity setting, SNs must trust one another not to lose critical session details or state for any reason. Because the initiated session cannot be invalidated with the SIT (except the initial SIT), its full Expiration timer is required to elapse without SRCs, which presents a dangerous opportunity for malicious session lockout. Nodes will not honor this Persistent flag unless they are explicitly using the Strict ICM.

N (New)

A flag which indicates that the NDSO intends to establish a new session for authenticating NDP communication between two nodes. The ND SLLAO is required to be included in the outer ND packet as an additional option when this flag is set. If no SLLAO is attached to the ND message, then receivers must ignore the NDSO.

A (Acknowledgement)

A flag which indicates that the NDSO is responding directly to a previous NDSO having its New flag set, to acknowledge the successful formation of a new session. The ND TLLAO is required to be included in the outer ND packet as an additional option when this flag is set. If no TLLAO is attached to the ND message, then receivers must ignore the NDSO.

Version

A 5-bit version identifier that could be used in future protocol versions to dynamically change the structure of the option. This field is required to be set to a value of 1, but

future additions to the structure of NDSOs might increment the static version number based on forthcoming features or other modifications. This is helpful if, for example, SHA-256 were no longer a viable hashing algorithm to use for ZKP.

Reserved

Space reserved for future use. This field is set to zero and is ignored by receivers.

Expiration

Dictates the amount of time taken in minutes for the ND session to expire when no session activity occurs or SRCs are received. The ‘expiration' of a session means both storage of related session state and associated NC entries are immediately destroyed. This value is required to be greater than or equal to 1.

SessionID

A pseudo-random 32-bit identifier used to track and pinpoint an ongoing session between two nodes. This value cannot change while a session instance is kept active, lest NDSOs be lost or invalidated at the receiver.

CounterMax

A big-endian value expressing the number of times a secret Root Hash H_0 is iterated with the function H(x) to produce the Final Hash H_c, where c is equal to CounterMax. This value should never be a value greater than 10,000 and should always be greater than or equal to 500.

Counter

A big-endian value less than or equal to CounterMax, expressing the number of iterations with the function H(x) over the CurrentHash value which are required to form the FinalHash value. If this field is set to CounterMax, then the CurrentHash is required to be

equal to the Session Interface's current Root Hash, and the NewHash field is required to be set to the new FinalHash value of a newly-formed Hash Chain on the SI, with the same CounterMax length.

FinalHash

The full SHA-256 Final Hash value from the SI's current Hash Chain. This value will not change per SessionID, unless the previous NDSO provided a non-null NewHash value and a valid Root Hash in the CurrentHash field.

CurrentHash

The full SHA-256 ZKP token which, when iterated Counter times with the function H(x), will produce the FinalHash value.

PeerHash

The currently known FinalHash value of the Session Peer. This field indicates to the peer which HC the SI is currently expecting ZKP tokens for. It is helpful for possibly desynchronized sessions where the SP must know which intermediate hash to provide based on the Counter value sent in the previous NDSO. This field is initialized to a null value (0) when the NDSO indicates session initiation–i.e., when the New flag is set.

NewHash

When the CurrentHash value equals the Root Hash of the SI's current Hash Chain and Counter is equal to CounterMax, this field is set to an upcoming Final Hash to use in the SI's next HC, which is required to inherit the current CounterMax value. Otherwise, this value should always remain a null value and be ignored by receivers.

4.6 Packet Processing Rules

4.6.1 Senders

Senders must not append NDSOs to NDP traffic if there is not an active session with the target of the ND packet, and the sender does not intend to initiate a new session. If the sender wishes to initiate a new session, it must set the New flag and initialize both the PeerHash and NewHash fields to a null value (0). When initiating a new session, it is the sender's responsibility to determine the new session's properties: the SessionID, the CounterMax value, session persistence, the Version, and the Expiration value. If the sender is acknowledging receipt of a session initiation request and the session initiation is accepted, then the sender must reflect and use some of the properties chosen by the initiator (the SessionID, Version, and Persistence values) in its acknowledgement NDSO. The acknowledgement NDSO must also by its functionality set the Acknowledge flag to express its intent to acknowledge and agree to the two-way session establishment Acknowledgements must also include the value of the initiator's FinalHash in the PeerHash field, so the initiator can confirm that the recipient has received the correct Hash Chain details.

The CounterMax value is not recommended to match the value set by the initiator. Right before sending the NDSO, senders must ensure that the current Counter value of the HC_{SI} (the Hash Chain of the Session Interface) has been moved to its next value and that the generated ZKP token at that updated value is inserted into the CurrentHash field. The Counter value must always be a value greater than the previously sent NDSO Counter value for this SessionID from the perspective of the sender's HC_{SI}. If the sender's NDSO intends to expose a Root Hash in order to move to a new HC_{SI}, then the CurrentHash will be set to the HC_{SI} Root Hash, and the

Counter field will be equal to the CounterMax value for the HC_{SI}. In this same packet, the NextHash field will consist of a non-null value indicating the FinalHash of the upcoming HC_{SI}.

When the NDSO does not indicate a change of the RH for the HC_{SI}, then the NewHash field is always required to be a null value (0). If the NDSO represents a session initiation request, then the PeerHash field is required to be set to a null value (0). Importantly, senders should never reveal the Root Hash of the current HC_{SI} if there is any doubt of connectivity with the genuine SP. Revelation of the HC_{SI} RH must be done selectively and judiciously to protect the session from malicious hijacking.

4.6.2 Receivers

Option processing rules for receivers consists of a tangled web of instructions that adhere to the many functional rules and constraints established for ND sessions. This section attempts to capture all related rules for receivers, but at the risk of redundancy does not bother to state what may seem obvious based on what has already been presented. Any detail withheld implies that implementations or future research should be equipped to reasonably decide the correct course of action based on the purpose of ND sessions. A formal, future specification of ND sessions will include many more explicit and technical rules which do not distract from the primary objectives of securing link-layer address ownership and device identification.

Nodes receiving invalid NDSOs are required to reply with an ICMP Destination Unreachable packet having a Code value of 5 (of type Source Address Failed Ingress/Egress Policy) per RFC 4443 [12]. If the receiving node does not know the LLID of the NDSO sender, and if no SLLAO or TLLAO was attached to the incoming ND packet, then the ICMP Destination Unreachable message is not required. This is because the receiving node would not

know where to forward the packet or frames. The ICMP reply is intended to signal to senders that their most recent NDSO was in some way invalid and not accepted, so that the sender can take immediate and appropriate action in response.

NDSOs are required to be received only via unicast ND traffic. The exception to this rule is the initial multicast Neighbor Solicitation that resolves a target's IP address to its LLID while initiating a new session. NDSOs attached to other multicast ND messages, or on ND packets other than NSs or NAs, must be silently discarded and must not invoke an ICMP Destination Unreachable response from the receiver. Receivers must expect NDSOs to conform to the ND session rules and option structures for the indicated Version value of the session. If there is any discrepancy in the structure of the received NDSO compared to what is indicated by its Version, then it must be ignored and discarded.

Any received NDSOs that are not initiations (as marked by the New flag) are required to be discarded if there is no active session with the sending source IP address. Active sessions can be found by searching local session state storage for the purported SessionID value given in a received NDSO. If the received NDSO does not match the Persistent, Expiration, Version, or CounterMax fields of the SP's active session details–or it contains a Counter value less than or equal to some Counter value already received for the known HC_{SP}–then it must be considered invalid. If the local ICM specifies that sessions may be formed, then any incoming NDSOs appearing to come from an IP Source Address that is already associated with an LLID in the local NC, or which attempt to affect the local NC for that IP address, should be given special attention. If a session is already correlated to the NC entry and the current ICM is not permissive, then incoming ND packets from that IP Source Address without NDSOs must be ignored. If a session is not pinned to the NC entry, then it is left to the local ICM to determine the next course

of action for handling the NDSOs. The New and Acknowledgement flags must not be set if the NDSO is not an initiation or an acknowledgement of an initiation, respectively.

4.7 Modifications to Neighbor Discovery Behavior

The introduction of the NDSO option proposes only a few modifications to the behavior of Neighbor Discovery: it simply adds a new option and inserts itself into key points within the ND process (similar to VBA). More specifically, the NDAR and NUD processes are targeted as crucial areas of session enforcement based on the ICM of the SI. Any ND packet seeking to modify the NC of a neighbor is required to follow session establishment and maintenance procedures per the current ICMs of the sending and receiving interfaces. If the active ICM dictates that a session must be active, then senders are required to provide the correct ZKP authentication to change the known LLID, the known target IP address, or any other session information.

Multicast ND messages will never include an attached NDSO, except initiation NDSO messages which can be sent with a Neighbor Solicitation packet to a solicited-node multicast address. Implementations for NDSOs can choose to ignore this optional capability, restricting NDSOs to unicast-only traffic by only delaying session initiation until the NUD (unicast NDAR) process begins, shortly after SNs exchange their SLLAO and TLLAO details. The NUD process is required to ignore normative Reachability Confirmations coming from upper-layer protocol ‘hints’ if the current receiving ICM is not in the Ignore mode and a session is already ongoing with the neighbor. For example, TCP transmissions with an SP are no longer sufficient to keep the state of the NC entry at the SP in a REACHABLE state for the active session. Nodes instead use unicast NDAR messages, also known as Session Reachability Confirmations, to proactively

avoid having any STALE cache entries; acting as if the NC entry already were in the STALE state to begin with.

NC entries will never be purged if unexpired, valid sessions are still stored and associated with them, unless the SP indicates a change in the LLID or the IP address being secured. This allows session persistence features to ‘remember’ the parameters associated with NDAR transaction instances in the long-term; because part of ZKP token generation and HC formation relies on NC entry details to form the salt value used in the hashing function H(x). Additionally, if a session is invalidated, expired, or otherwise destroyed, then the associated NC entry is always immediately purged regardless of each SN’s active ICM.

4.8 Session State & Lifecycles

Every ND session maintains a lifecycle: from initiation and acknowledgement, through maintenance with SRCs and rotating Hash Chains, to eventual session expiration or invalidation. All sessions are initiated, and all sessions will eventually end. Throughout their lifecycle, as more and more Session Reachability Confirmations are processed, Hash Chains will begin to run low on ZKP tokens; at which point each Session Node will arbitrarily choose to rotate to the next Hash Chain. The process of swapping Hash Chains can happen indefinitely and safely as long as each Session Node does not create any predictable pattern for the revelations of their Root Hash values. Finally, the end of sessions always comes by means of invalidation or expiration: the former occurring when incoming invalid NDSOs trigger the SIT and it elapses, and the latter occurring when no SRCs have been received for the session within the expiration window.

4.8.1 Preserving Session State

It is essential to always preserve the state of ND sessions and to keep the information as recent as possible. While the responsiveness of a node in caching or updating received session details is not of critical importance, it is instrumental in preventing any possibility of replay attacks against SNs. Because session durations may extend up to hours at a time without any activity (if persistent or not invalidated by an SIT), SNs are required to use persistent, non-volatile storage to preserve session details. To preserve session details means persistently preserving the state of any associated NC entries is also required, regardless of the time elapsed since the last Session Reachability Confirmation was received.

If a node is powered down or otherwise loses data in its non-volatile storage, then it must not lose session information. Depending on the activity of the link and the volume of active sessions, these details might update at a rapid pace. Therefore, implementations may choose to keep in-flight session details in a volatile memory pool and regularly shadow it in batches to non-volatile, persistent storage. All techniques for maintaining persistent session storage information are left to independent implementations to handle.

4.8.2 Initiations

When a node initiates a session, it should indicate its intent by setting the New flag on an NDSO to append to any NS or NA packet, along with the appropriate LLID-providing option (SLLAO or TLLAO). If the receiver of the NDSO is not already engaged in some session identified by the NDSO's SessionID field or by the initiator's IP Source Address, then it can choose to respond with an NDSO including its own HC details, with the Acknowledgement flag set. If a non-persistent session already exists based on the information provided by the initiator,

then the NDSO must instead be considered an attempt to trigger the SIT (depending on the current ICM of the receiving interface). If both NDSOs are valid and no session had existed prior, a successful two-way handshake has been completed to establish a new one. To discover in-progress sessions, nodes can search information in their local session storage with the following process:

- Check the SessionID field of the incoming NDSO. If it matches a known identifier that is stored locally with associated NC details, then the NDSO can never be considered as an initiation.
- Examine the IP Source Address of the NDSO, if and only if the incoming packet is not a multicast Neighbor Solicitation. If this IP address appears in the local NC and has an existing LLID associated with it, then there may or may not already be an associated session with the sender. An examination of the local session state storage should be able to provide more information per the implementation.

Any deviation from this process, including strange behavior or spurious NDSOs not first using the New flag to indicate initiation must be considered invalid by receivers. Sessions must not be considered as started or active without first being initiated or acknowledged by the SP. Once a new session is formed, both SNs, regardless of the ICM on each interface, will set and start a 1-minute SIT that must not be canceled or stopped unless a valid Session Reachability Confirmation is received from the SP. This process prevents rogue or bogus sessions from being initiated and then kept around for the entirety of the indicated Expiration time for the fake session. Thus, a flurry of fake sessions will not linger because they will never receive positive confirmation that the SP actually exists and is responsive on-link. The first valid SRC to be

received from the SP will always cancel the initial SIT and enable the full Session Expiration Timer with a duration specified by the SP's Expiration selection. Each subsequent, valid SRC from the SP must restart the SET at its full duration, allowing the session to be prolonged indefinitely if desired. In non-persistent sessions, the 1-minute SIT can be triggered again at any time by receipt of an invalid NDSO message.

4.8.3 Shifting Link-Layer & IP Addresses

There are times when the LLID or IP address of a sending node will change. Certain node-specific parameters–like the node's communicating IP address(es) and LLID–are components of the salt value that is reconstructed by SPs when validating ZKP tokens. Moving to an entirely new HC on the SI is the only option available to keep a session alive and carry it across the transition between an IP address or an LLID. This same requirement to transition away from the current Hash Chain also necessarily applies to requests that change the sending HC's CounterMax value or its Expiration value. Nodes wishing to update salt-affecting values for their HCs (the session expiration, the session's ID, the current CounterMax, the SI's LLID, and/or the SI's IP address engaged in the session) must reveal their current HC's RH and transition to a new HC FH in the same message. This is in order to move to a new Hash Chain that is constructed from the updated salt value.

Session Peers will be capable of smoothly detecting the transition between Hash Chains and adjust accordingly, since exposing an RH is a high-priority action. To repeat, the process of notifying the SP of this change involves (1) exposing the RH while simultaneously (2) expressing which FinalHash comes next for the upcoming HC to be originated from the SI and

validated by the SP. Armed with this safe transition capability, most sessions are flexible enough to transition uninterrupted between any change of Session Node details.

4.8.4 Session Maintenance

Session Reachability Confirmations are very similar to Neighbor Reachability Confirmations as defined in Section 7.3.1 of RFC 4861 [11], except an SRC does not include the 'hints' given from upper-layer protocols in its definition. SRCs consist solely of valid NDSOs that are attached to unicast NS or NA packets exchanged as a result of the normative NDP NUD process. Certain session-related timers rely on SRCs to provide consistently recent confirmations that an NC entry is still owned by the original peer participating in the session.

Any received SRC usually allows extension of the session by either restarting or altogether clearing the related timer(s). During the maintenance of a session, SRCs for non-persistent sessions must be frequently sought by SNs to prevent any coupled NC entries from entering a STALE state. In persistent sessions, a STALE state is not avoided for NC entries, but their association to their sessions lingers until the SET elapses (if at all). The proactive behavior of SRCs simultaneously provides reachability confirmation while prolonging an active session, persistent or not, by restarting all relevant timers on a rolling basis.

4.8.5 Timers, Failures, & Invalidations

Each session is pre-equipped with a pair of timers that simultaneously control different protections and optimizations. The Session Invalidation Timer and the Session Expiration Timer control session invalidation and expiration, respectively. The timers will destroy the session at the SI interface if either one of them elapses. They are refreshed and triggered by key actions

based on the various optimizations and protections. Figure 21 demonstrates the placement and duration of each timer for persistent and non-persistent sessions, with different triggers that might cancel or refresh each timer.

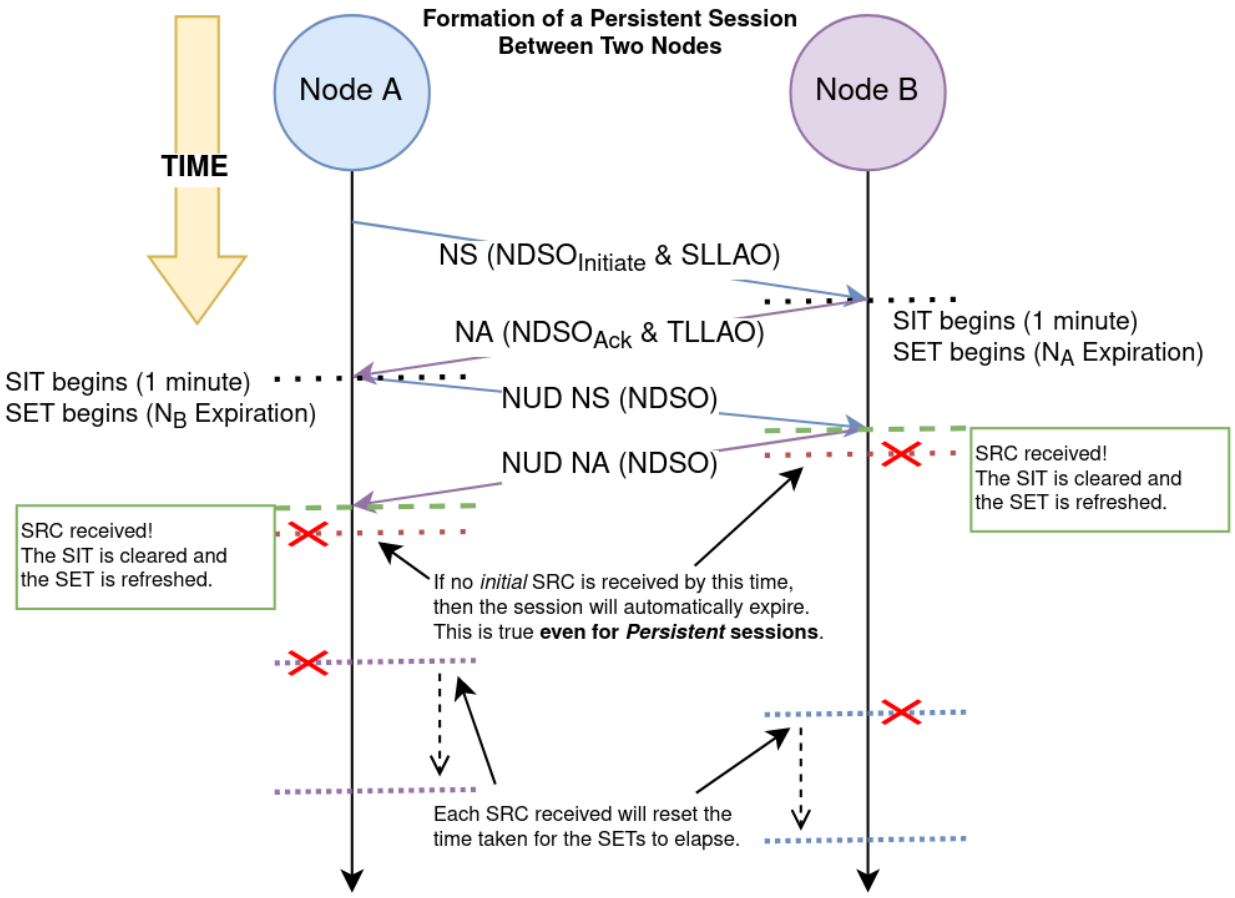


Figure 21. ND sessions use the SIT and SET timers to prevent malicious session flooding, session lockouts, and undying sessions. Being freed from these potential issues is a careful balance between session rigidity, security (with persistent sessions), and flexibility in case a session is locked out (with non-persistent sessions).

4.9 Example Session

This section will briefly produce a contrived example of a legitimate NDSO exchange and session establishment between two neighbors, demonstrating how sessions are intended to

function and how they might be attacked. Node A (N_a) is the soliciting node initiating the session (the SI) while Node B (N_b) is the receiving and acknowledging Session Node (the SP). Neither of these two nodes have preexisting knowledge of the link-layer address of the other, so they must be resolved using the normative NDAR process.

- N_a (SI) has the IP address 2001:db8::12bc:8090/64 with an interface LLID of 3B-55-0C-00-12-33.
- N_b (SP) has the IP address 2001:db8::aaaa:bbbb/64 with an interface LLID of BC-71-A3-89-CC-60.
- Both Session Node interfaces are operating in the Gracious ICM.

N_a must now perform NDAR for the IP address of N_b to forward link-layer frames to the correct address, per normative NDP processes. Since both interfaces are using the Gracious ICM, no ND messages will be accepted without requisite NDSOs that properly validate.

1. N_a receives an instruction to connect with N_b, so it must resolve the LLID of N_b's IP address at 2001:db8::aaaa:bbbb/64 (an on-link address that does not require routing to reach). If this target IP address were on a separate subnet, the session would instead have been established between N_a and the appropriate router used to reach N_b.
2. Using a pseudo-random seed value, its own sending interface LLID, and source IP address, N_a generates the HC_{SI} with an arbitrary CounterMax of 5,000. The FH and the $H_{4,999}$ CurrentHash is calculated and the Counter of the HC_{SI} is incremented to 1, to match the CurrentHash step at $H_{[CounterMax - Counter]}$. The Expiration value is also set to 45 minutes, specifying the peer should expire the session after that time if no SRCs are received. The session is not labeled as Persistent, but the New flag is set.

3. N_a sends a solicited-multicast Neighbor Solicitation to ff02::1:ffaa:bbbb, including the initiating NDSO with current parameters from HC_{SI}, a non-Persistent Expiration value of 30 minutes, and a pseudo-random SessionID value. The NS also includes the SLLAO option giving the sending interface's LLID of 3B-55-0C-00-12-33 as required.
4. N_b receives the multicast NS. Before entering the LLID and IP Source Address from the sender into its NC, N_b searches for a valid NDSO according to its active Gracious ICM. If one is found, it examines the SessionID to determine which session this might be relevant to. Since no session yet exists by the received identifier because it is new, N_b checks its NC to determine if a cache entry for the IP Source Address already exists. If a cached entry does exist, a 1-minute SIT for that NC entry's session begins because the incoming packet is a re-initiation of a session. If a cached entry does not exist, N_b begins calculating and storing the details of HC_{SP}.
    - Since the incoming NDSO packet claims the Counter is 1, N_b must use the function H(x) to iterate the input CurrentHash with the appropriate salt value only 1 time (per Counter) to get the value of FinalHash. Once that computation is completed and verified, N_b creates a new NC entry per the ND specification. The SIT of 1 minute begins on N_b for the SessionID and the SET at the Expiration value (45) given by N_a, in minutes, also begins on the node for the session. The SessionID is then locally correlated to the new NC entry.
5. N_b then forms its own HC_{SI} as in Step 2. It opts to use a CounterMax of 8,000 and a first Counter value of 1.

6. N_b sends a unicast Neighbor Advertisement with the Acknowledgement flag set on the attached NDSO, using its own HC_{SI} and session details. This NA also includes the required TLLAO option to indicate the advertised LLID of BC-71-A3-89-CC-60.
7. N_a receives the unicast NA and follows a similar session lookup process as in Step 4. Parsing is almost immediate because the SessionID is already a known and shared value between the two nodes from the initiation NDSO sent earlier by N_a. N_a validates the incoming Acknowledgement NDSO and stores the SP's HC information in its HC_{SP}. The SessionID is then locally correlated to the new NC entry from the received NA. The 1-minute SIT begins on N_a for the SessionID. The SET of 30 minutes, per the SP's instructions, also begins on the same node for the SessionID.
8. The first set of qualifying NUD messages exchanged between the two SNs with NDSOs attached act as initial SRCs, so the SIT timers are canceled upon their receipt and the expiration timers are each reset to the specified Expiration times set by each SP. These expiration timers (SETs) are refreshed at each node every time a valid, verified NUD packet with an NDSO (i.e., an SRC) is received.

The established session will then enter a stable maintenance phase after this initial 'handshaking' process, where NUD packets with NDSOs attached will continue to refresh the SITs and SETs for each SN in the exchange. If an SI's LLID changes (or it attempts to force a new one), its own HC_{SI} Root Hash must be revealed to the SP. The current HC_{SI} at the SI must simultaneously move to a new Hash Chain built with this new LLID, in order to update the associated Neighbor Cache entry at the SP accordingly.

This example session represents an overlay atop the normal NDAR process, but theoretically this process of session initiation can occur during any point of Neighbor Discovery depending on each SN's ICM setting. Initiation may even occur in the middle of NUD transactions that have been ongoing for quite some time. In such cases, as long as the LLID option of whichever type is present with the NDSO, then the session can be formed and maintained the same as any other time in the ND process.

4.9.1 Attempting to Subvert the Session

In this section, key steps from the example exchange above are explored for exploitation by a malicious neighbor, labeled N_m. This hypothetical malicious node has no particular goal and simply seeks disruption of the target session by any means necessary.

- Between steps 2 and 3, N_m can immediately respond to the NS from N_a with both an invalid NDSO and its own LLID in the TLLAO. This is an attempt at a classic traffic redirection (i.e., on-path) attack and is also an attempted hijacking of the session's acknowledgement. It is ultimately a race condition: if N_b receives the NS multicast and intends to respond, then N_m must reply as quickly as possible to lock in and redirect the session. Solving the issue of redirection attacks upon session initiation, as an abuse of Trust on First Use, is beyond the specific scope of what ND sessions aim to solve. It is better left to solutions which perform some form of LLID binding or verification, where nodes cannot falsify or redirect an LLID for a local IP address. That is the exact purpose of Voucher-Based Addressing.
- Once the session enters its 'maintenance' phase, N_m could somehow intercept a ZKP intermediate hash from one of the two SNs (through some extrapolated means). Even so,

providing this ZKP to the SP–as long as it is not the RH of the SI's HC–does not permit changing any NC entry options at the SP because the computed Final Hash uses the known LLID and target IP address of the cache entry from the SI. However, if the intercepted ZKP is the RH of the SI's Hash Chain, then the session can be hijacked entirely.

Knowledge of the RH allows the SI to change the LLID of the NC entry on the SP. This is precisely why (1) NDSOs are almost entirely unicast-only options and (2) revealing the RH of an HC is intended to be randomized beyond a certain Counter threshold–to make RH ZKP revelation harder to predict. This is an unfortunate vulnerability intrinsic to NDSOs but is largely infeasible to exploit due to the amount of impersonation effort (or luck) required to obtain a valid RH at the right time. If N_m is constantly arbitrating the SN connections at the link layer, then there are likely greater concerns about the security of the link that go beyond single-node impersonation attacks.

- In any phase of the process, N_m could try to blast either SN with New sessions matching the IP address of its peer. Non-persistent sessions, such as what is provided in the example, will begin a 1-minute SIT that can only be canceled by legitimate SRCs. As long as both N_a and N_b remain responsive, this is not a problem because legitimate NDSOs and SRCs will always cancel the respective timers that invalidate sessions. But if either node is away or off-link during these short timers, then the session could be invalidated after one of them elapses.

  The solution to repeated attempts of N_m triggering the SIT is to enforce the Persistent flag for the session during initiation. This would require both SNs (N_a and N_b) to switch to

using the Strict ICM, which would allow session persistence to be toggled during initiation. When the SIT is removed, the SNs maintain high fidelity in one another to remember all session details. They will never use SITs to preemptively invalidate their session before the entirety of their SETs elapses. Implementations for sessions can permit their own on-the-fly changes to active ICMs which might allow SNs to 'upgrade' ongoing sessions to persistent ones by some SN agreement, but specific details of doing so are left to each implementation and are beyond the scope of this research.

4.10 Transition Considerations

Transitioning to network-wide use of ND sessions is fairly simple regardless of the deployment environment. It is a matter of properly adjusting the configuration of each interface on-link to accommodate the capabilities of neighbors. The majority of nodes are suggested to start in the Ignore ICM when they are not ready to parse any NDSOs. Nodes should never be in the Strict or Gracious ICM while being a member of a transitioning network, because encountering a neighbor without support for NDSOs will make NDAR transactions fail consistently.

As more nodes become capable of managing sessions, the Gregarious ICM should be gradually introduced to have ‘pockets’ of successful sessions. Once those are proven to function properly, clusters of devices with Strict or Gracious ICMs configured could work without issue. Ultimately, a successful transition to using enforced sessions on a local network is accomplished by gradually sweeping all nodes from the Ignore ICM through to the Strict or Gracious ICMs, as the projected deployment situation and network infrastructure permit.

4.11 Security Considerations

Just as with VBA, the introduction of ND sessions is an inherently security-conscious and privacy-focused amendment to the way Neighbor Discovery functions. Therefore, careful consideration of each security issue is merited. This subsection will briefly discuss a few key points which deserve explicit attention, but it is not all-inclusive of possible issues that sessions might encounter in practice.

4.11.1 Address Bindings & Lockouts

Sessions use an automatic Trust on First Use authentication, necessarily granting trust to the first legitimate respondent in a newly-initiated session. Once the session has started and the first Session Reachability Confirmation is received, the session can be extended indefinitely as long as both Session Nodes remain online and responsive to one another. Session authentication HCs on both sides each rely on the known LLID and target IP address of their SP to form the FH from some starting intermediate hash within the HC of the SP, since the salt value in the HC includes the SP's LLID and IP address. There is nothing native to NDSOs which prevents illegitimate LL2IP bindings from being reported through NDAR-driven on-path attacks. If these attacks are performed successfully despite a session overlay, attackers could subsequently 'lock out' any genuine respondents who truly own the solicited (or advertised) IP addresses, thus indefinitely denying their ability to form a genuine session with the peer.

Other solutions exist to provide some proof of address ownership in one way or another (e.g., Voucher-Based Addressing and Cryptographically Generated Addresses), and those specifications are designed to resolve this falsification issue in the first place. But the focus of this subsection seeks instead to emphasize the importance of recovering from a session lockout

when there is no LL2IP binding verification in place during NDAR transactions. Once malicious nodes have been removed from the link, the legitimate neighbors who own the impersonated IP addresses must have the ability to recover communications with the original peers whom they were soliciting (or accepting session initiations from). The Session Invalidation Timer for non-persistent sessions is a 1-minute timer that can be used to invalidate a session that otherwise might have spanned until its full expiration time (via the Session Expiration Timer). Removing lockouts for persistent sessions by design requires manual intervention from a user or system administrator.

When malicious nodes are removed from the link and proper Neighbor Cache associations are restored where required, the victims of impersonation only need to reattempt establishment of their sessions by re-initiating new ones. If a preexisting, non-persistent session is found that is already correlated to the initiator's purported LLID and IP Source Address, then the session will be invalidated by the SIT after one minute elapses without the target SP receiving any SRCs. Since the impersonators are no longer active, there will be no nodes available to respond with valid NDSOs providing SRC, and the SIT is guaranteed to fully elapse. The SP will subsequently be freed to reestablish a session with the legitimate neighbor node.

Reconnecting neighbor nodes might find themselves unable to reattach to a session which was in-progress, as indicated by receiving ICMP Destination Unreachable message in response to their session-in-progress NDSOs. When this happens, the reconnecting nodes should immediately attach NDSOs with the New flag set onto a new Neighbor Solicitation packet, in addition to the required SLLAO attachment. This solicitation will eventually invalidate the session if it is not marked as persistent. Before this occurs, the node will wait one minute for the SIT to elapse. After waiting, the node will either continue the initiated session or try another

initiation NDSO. If this option is again met with an ICMP Destination Unreachable response, then the sender should keep trying to invalidate whatever session lockout exists, and the sender can utilize other means in parallel to alert appropriate system administrators about its total failure to connect.

4.11.2 Session State Pruning

To mitigate attacks seeking to exhaust session tracking capabilities at a victim node, implementations should determine how to manage a flood of incoming sessions. If the resources on a Session Node are becoming more constrained and approaching their limit, then to preserve system resources it is suggested that the least active sessions should be pruned. While session flooding may be a rare event, it is an important security feature to consider for network devices having fewer resources at their disposal. ‘Pruning' in this case means deleting session details which have expired or have not been recently active within some arbitrary window of time, to make space for higher priority or newly-initiated sessions that might have more activity.

The Session Reachability Confirmation feature of ND sessions affords protections against this problem when massive influxes of sessions are malicious. Sessions use a 1-minute initial Session Invalidation Timer regardless of session persistence settings, cleared only when at least one SRC from the peer is successfully received. Implementations might wish to use their own algorithms and heuristic analyses of active sessions in order to determine which of them are considered the 'least active' targets for pruning. Sessions should not be pruned if there is no valid reason to do so because clearing any session details without a dire reason will defeat their protections easily. It will allow malicious parties to use floods on-demand in order to 'override'

or completely dump legitimate sessions and temporarily lock out legitimate nodes from each other.

4.11.3 Regarding Denial of Service

The goal of Neighbor Discovery Sessions is to validate persistent peer identities at a very low-layer, local protocol while maintaining simplicity, flexibility, and privacy. Mitigation of NDP denial of service attacks thus becomes an auxiliary goal, beyond the scope of this research, that might be achieved by applying other protocols and related research. Again, just as with VBA, placing the burden of solving all of these problems onto ND sessions could readily reduce their practicality. It will force them to apply multiple simultaneous mitigation strategies as part of the core protocol specification, rather than leaving the mitigations as optional and decoupled add-ons. However, denial-of-service attacks indeed remain a serious concern when practically applying sessions because it is an additional overlay atop a default protocol which already has its own weaknesses. When a denial-of-service topic is presented in this section without a solution, it is strongly implied that resources beyond this research should be applied to better mitigate the problem.

Bogus Session Option Replies & Hijacking

A series of bogus NDSOs cannot invalidate a session if the legitimate SNs are both still online and are receiving link traffic normally. This is because the 1-minute initial SIT will always be canceled upon receiving any valid SRC indication from the SP. Persistent sessions are always immune to this attack. Additionally, NDSOs with the correct SessionID are difficult to spoof or determine without traffic interception because NDSOs

only appear in unicast packets beyond an initial Neighbor Solicitation multicast. However, falsifying a target IP address in an SLLAO or TLLAO stub is an easier way to bypass the requirement for knowing the current SessionID.

Flip-flopping Session Expiration Timers from invalid received NDSOs might become a problem on some systems where sessions are not persistent and ICMs are not strict enough. Where such concerns might apply, some kind of network intrusion detection system capable of determining floods of bogus NDSOs from neighbors is likely the best option. The system could be tasked with either reacting to the threat(s) directly or notifying an administrator who can resolve the issue–the choice is left to the implementation and use-case.

Malicious neighbors knowing the SessionID to target with bogus NDSOs might be able to simultaneously (1) start an SIT on an SP with which it wants to communicate or intercept packets from, and (2) somehow disconnect the other SP from the link through exploitation until the 1-minute SIT elapses. Once the disconnected peer reconnects, it will be denied communication with the SP and 'locked out', because the malicious host has hijacked the session. Once again: either session persistence or some other form of network monitoring and denial-of-service protection is required in this instance to guarantee that the malicious node cannot forcibly disconnect neighbors from the link.

Resource Exhaustion

Storing additional state information about sessions could be considerably costly on some devices if NDSOs are abused to generate and store many fake sessions. Session State Pruning attempts to mitigate this security concern by ensuring that resources from unused

or bogus session initiations are returned expediently and intelligently without affecting legitimate, recently active sessions.

Resource exhaustion only becomes a true concern when session-capable nodes are active and very communicative on a large logical link with many other legitimate, session-capable nodes connected. Even with thousands of true neighbors on the same link, however, the increase in overhead from sessions–compared to what is already stored from NDAR transactions–is negligible. The only extra data points being tracked are session properties, small SI HC structures, small SP HC parameters, and the Neighbor Cache information with its association to sessions. Because management of system resources in varying situations resides far outside the scope of this paper, it is best left to implementations and operating systems to determine the best course of action for resolving resource constraints brought about by very busy shared links where ND sessions are actively processed.

ICMP Destination Unreachable Flooding

Attackers might try to send a high volume of forged ND packets to many neighbors in parallel with bogus, nonsense NDSOs. By using an attack target's forged IP Source Address, the ICMP Destination Unreachable messages will be maliciously reflected to the victim by the recipients of the bogus NDSO. The goal of this would be to elicit a massive volume of ICMP responses aimed at the target in a classic volumetric (asymmetric distributed) denial of service attack.

This attack is not feasible because the volume of forged ND messages is symmetric to the volume of ICMP responses sent by nodes. It is only sensible if the attacker does not wish

to send their own traffic flooding directly to the target when the link bandwidth does not support a direct flood, or if they have been otherwise administratively denied from communicating with the target directly. In fact, the ICMP Destination Unreachable responses might not even be sent if the forged packets (1) do not contain SLLAOs or TLLAOs, (2) there is no session already in place between the forged ND packet recipient and the active target's IP address, or (3) either the IP Source Address field or the LLID in the options is not correct.

Chapter 5 Prototyping & Results

5.1 VBA Generation Performance

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 the same as they would across other hardware. With the partial VBA generation implementation presented in Appendix 1, it is possible to create applications which can spawn VBAs on-the-fly based on any set of dynamic or precompiled parameters. Appendix 3 lists the output of a hash generation program using the code found in Appendix 1, giving a more concrete depiction of how more concrete VBA inputs and outputs might interact.

Using the minimal Key Derivation Function costs for each default algorithm, Figure22 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–because extracting an averaged set of data has proven challenging–the relationships between the data and their interdependencies demonstrate how VBA might perform in practice.

Argon2 key derivation is highly preferred as the default Algorithm Type for VBA Link Vouchers, because, as per the data, 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 projections of baseline computation time for each VBA based on minimal KDF settings.

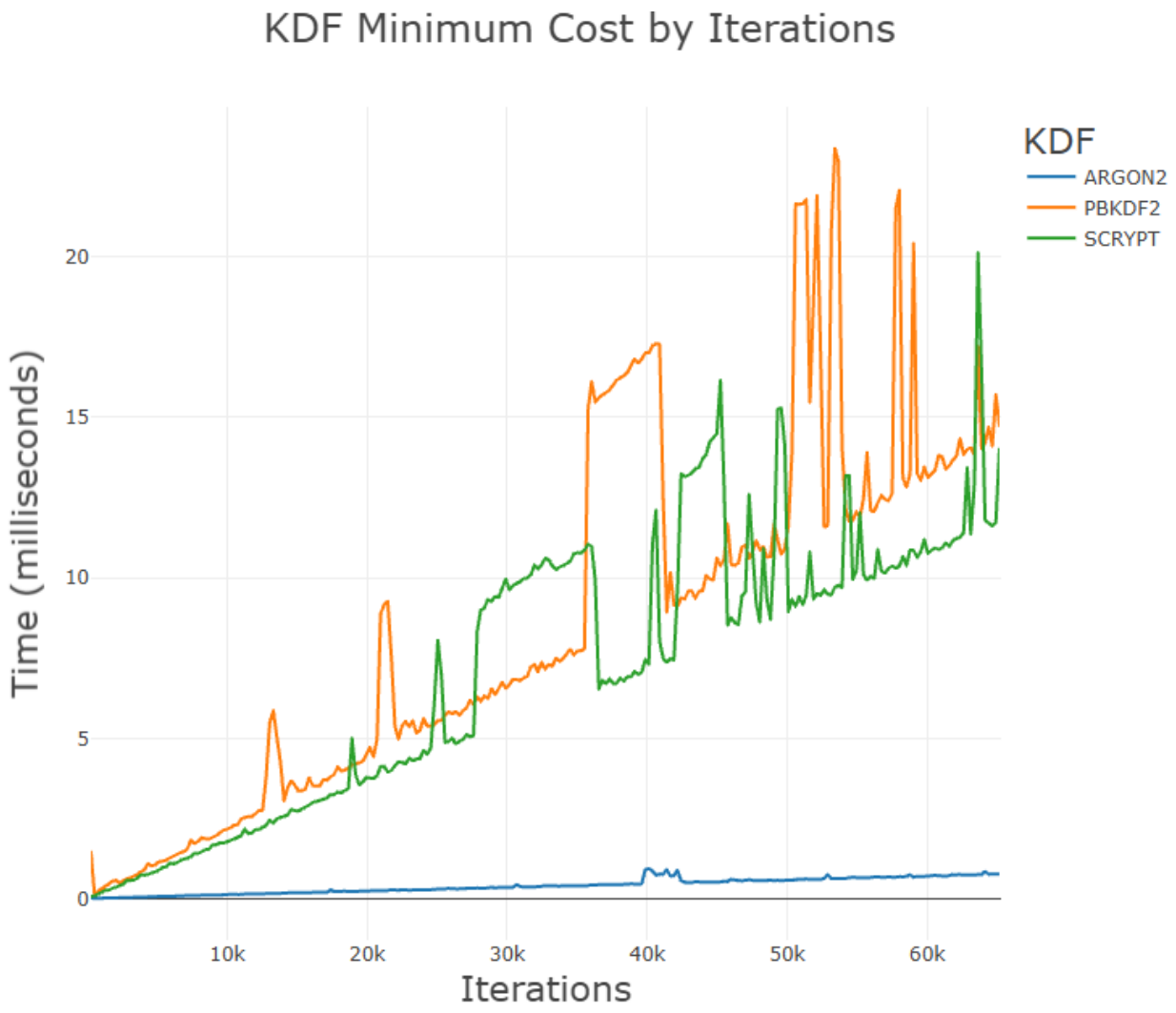


Figure 22. The three default key derivation functions are employed and benchmarked in VBA generation procedures with their minimum possible baseline difficulty settings. Each increase in the iteration count for each KDF expectedly shows mostly linear increases in address generation times. All outliers and deviations from the observable linear pattern are due to spurious slowness of the local processor on which these tests were run.

Next, the same laptop was used to evaluate the individual relationships between iteration counts at an arbitrarily high difficulty for each Key Derivation Function. Results from each Figure below 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 count values 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. As a reminder, selecting the iterations count is in the control of the VBA generating node. Neighbors who wish to ensure the legitimacy of a received LL2IP binding will be expending the same, symmetric amount of time and energy to verify the binding.

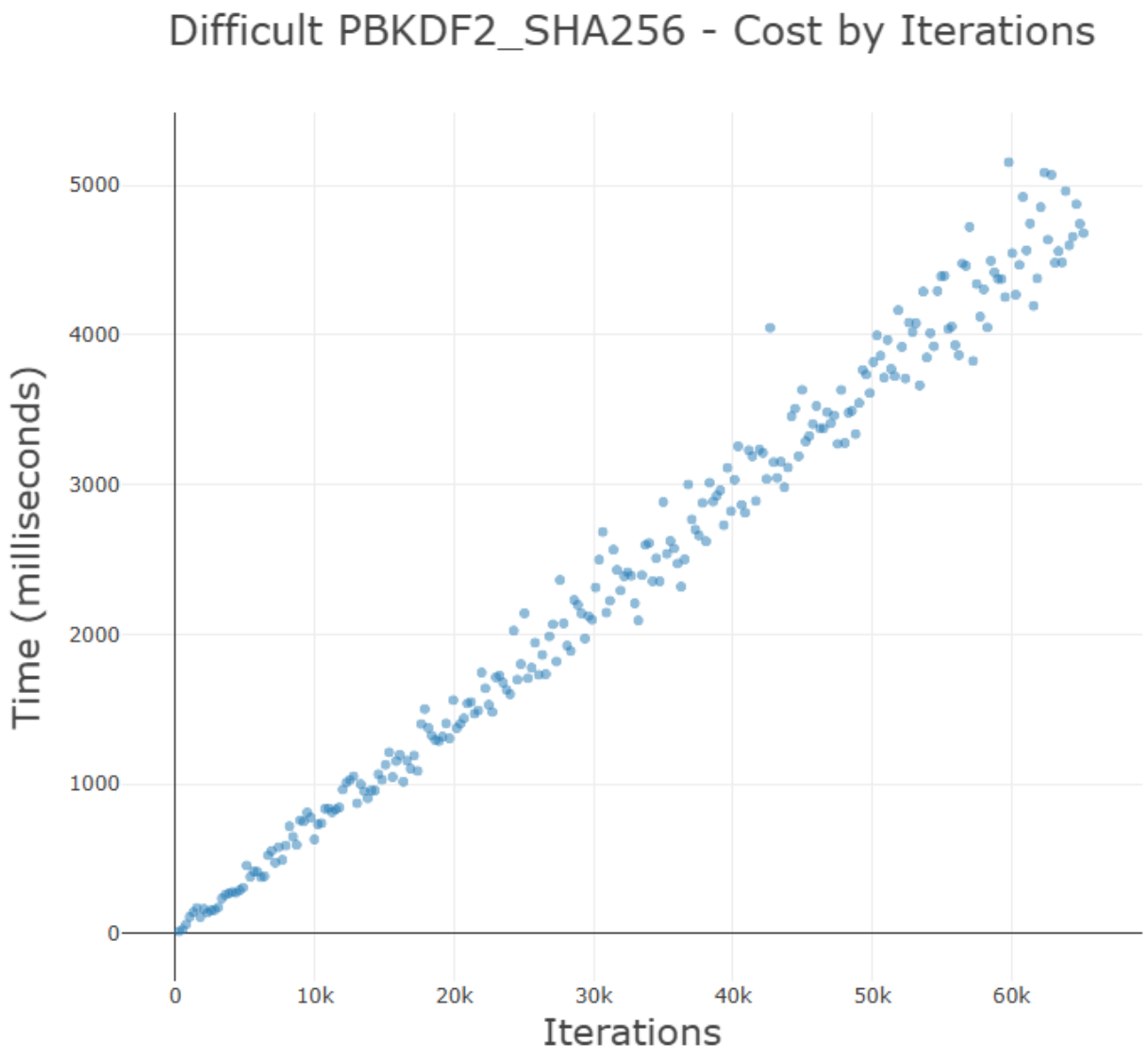


Figure 23. A high difficulty setting with the PBKDF2_SHA256 algorithm shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increase.

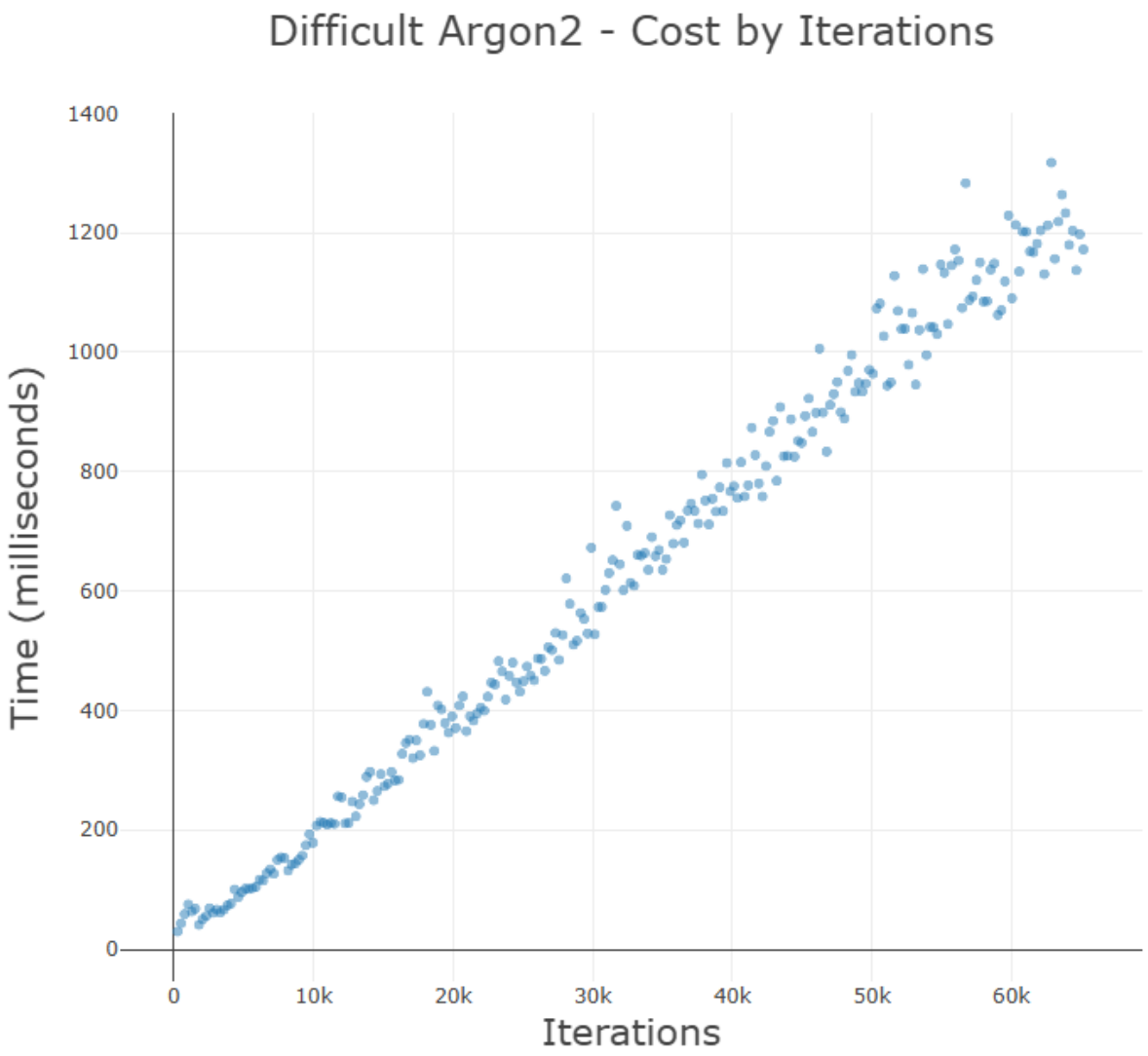


Figure 24. A high difficulty setting with the Argon2 KDF shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increase. The scale of the Time axis is smaller than the other KDFs.

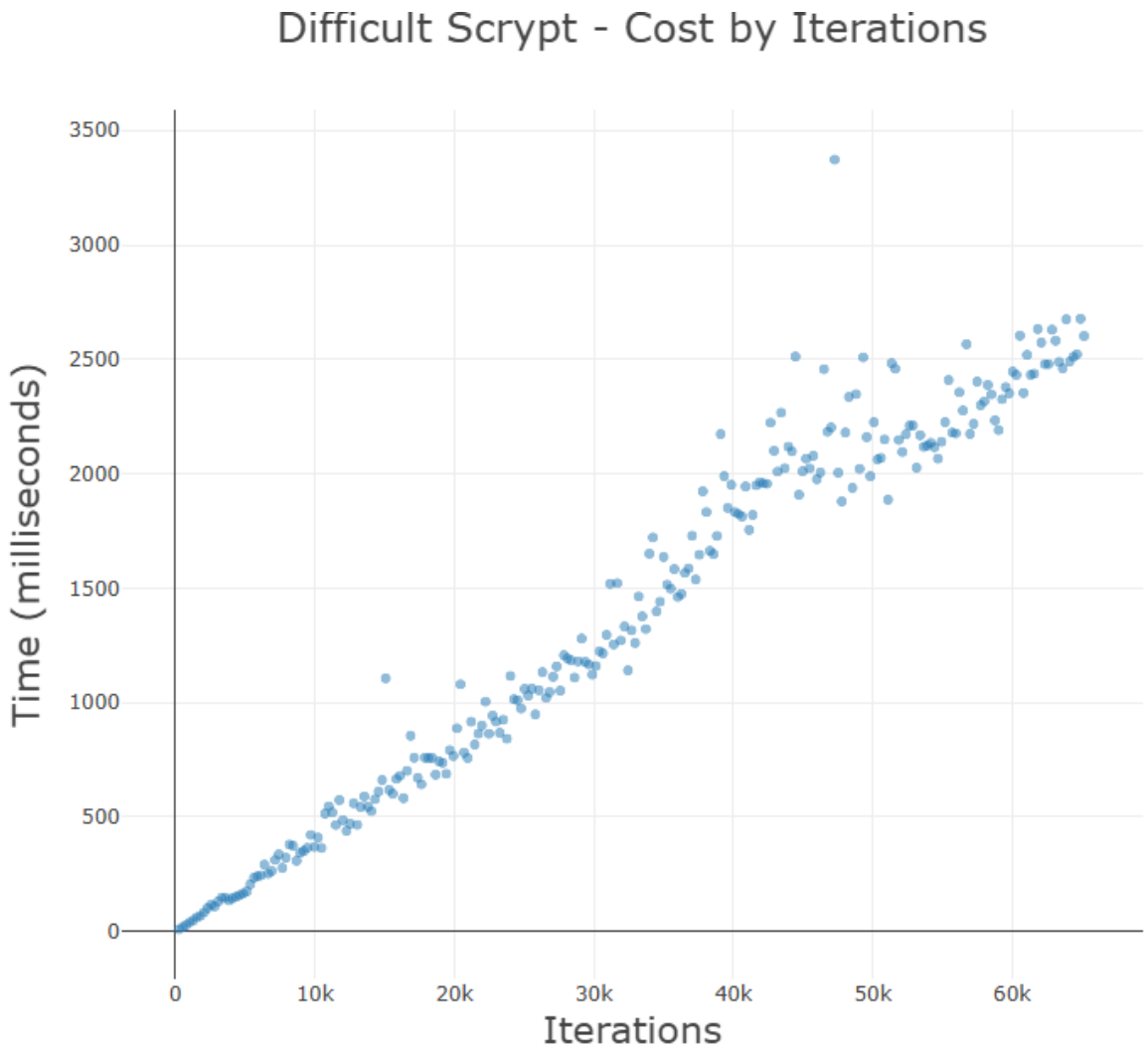


Figure 25. A high difficulty setting with the Scrypt KDF shows a mostly linear relationship between baseline time required to generate a VBA and the input iterations count. Data gathered is not an averaged composite of multiple runs. As the iterations count increases, variations in baseline computation time increases and the linearity of the graph gently curves downwards.

5.2 Discovering VBA Collisions

The problem of hash collision discovery in VBA is of paramount importance for the security of the proposal. According to Figure 26, the Birthday Paradox [63] suggests that in a pool of random 48-bit hash values, the probability of a hash collision reaches 50% around 1.976 x 10^7 (or just under 20 million hashes). Even though the output length of a SHA-256 hash is 256 bits (per its namesake), 48 bits are used in the calculation because only a portion of the generated hash must match to create a VBA collision.

Assuming a 1-millisecond hash generation cost because of some hypothetical LV baseline difficulty settings, and a low iterations-count selection by a target node: a minimum of about 19,760 seconds of computation time would be required to have a 50% chance of discovering a collision by testing random MAC addresses not equal to the target's MAC address. However, notably, the difficulty of collision discovery is greatly increased, if it is possible at all for the target redirection address, because there is a fixed input space from which collision-generating LLID values can be selected.

Collision detection is at the mercy of both the baseline security settings on-link and the iterations count chosen by the genuine VBA holder. Even a relatively meager 20-millisecond generation and verification time for a VBA imparts both a negligible added delay for processing legitimate NDP exchanges and a significant (i.e., 20-fold) delay for collision mining nodes. This small difference in computation time is worthwhile for legitimate nodes to enforce on their neighbors to increase VBA security.

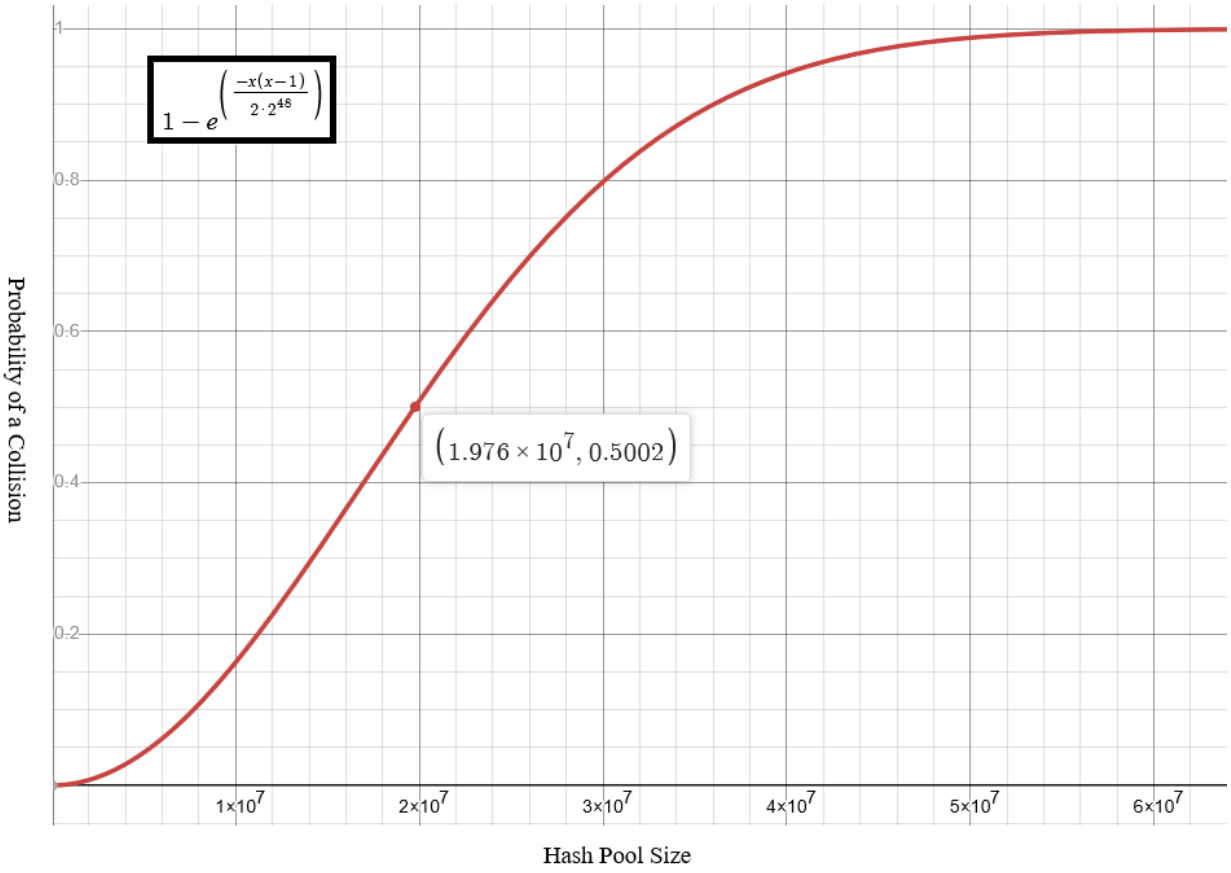


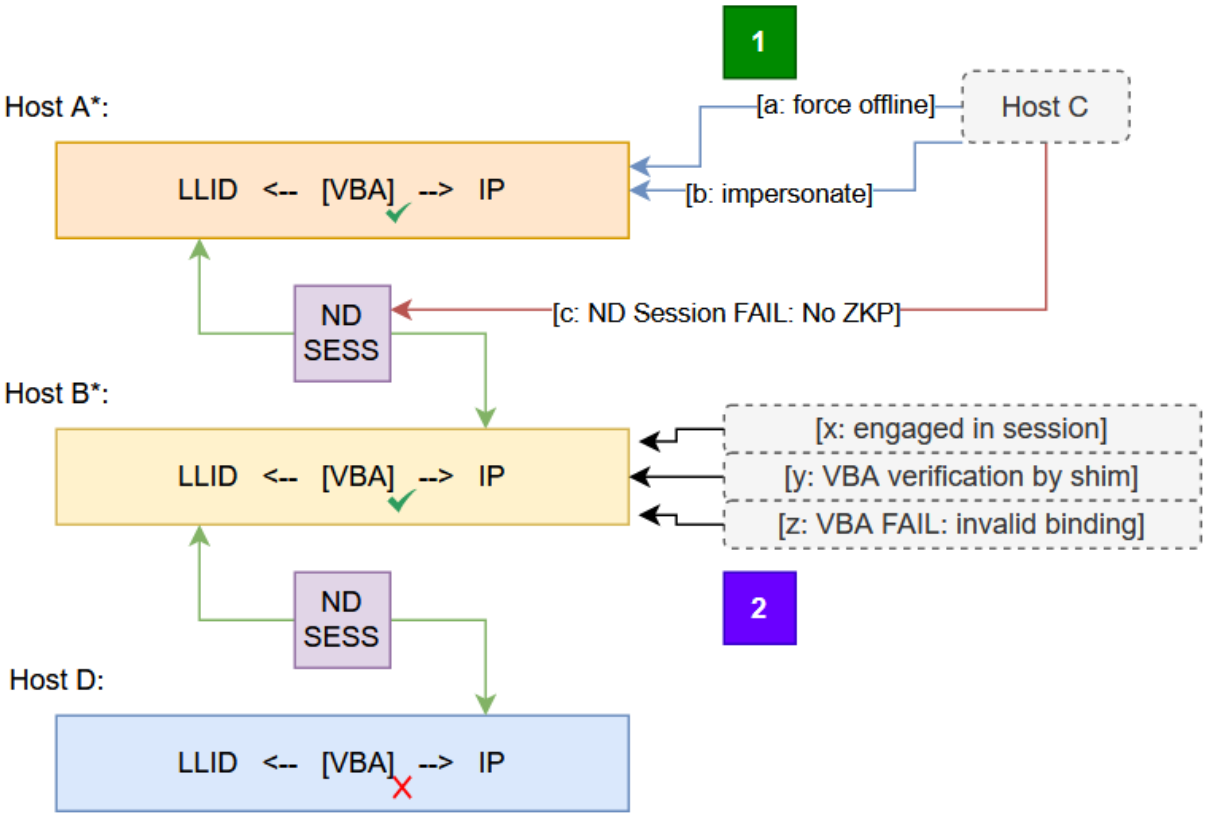
Figure 26. The chance of discovering a hash suffix exactly matching the 48-bit H value within a given VBA suffix is given by the Birthday Paradox formula. Almost 20 million different hashes must be generated to have a 50% chance of a hash collision with an existing VBA.

Chapter 6 Discussion

6.1 Synergistically Integrating Both Protocols

VBA processes dictate the validation of LL2IP bindings based on IEM, typically requiring that NDAR transactions always return the correct LLIDs for specified target IPs. Without VBA, nodes are free to express any NDAR transaction LLIDs and IPs (leading to traffic redirection attacks). Without ND sessions, there is no guarantee of identity beyond claiming and responding from a particular link-layer address. ND Session Options may be integrated harmoniously with VBAs, but they are not a requirement and will in practice add some degree of extra complication to NDAR exchanges.

Any extra complications introduced by these proposals are minimized for the sake of simplicity. The state-heavy nature of hash chaining with sessions is again considered a trade-off for allowing NDAR transactions to be fully guaranteed end-to-end. This is instead of requiring the deployment of more complex implementations or other capable network middleware and infrastructure. If ND Session Options are deployed in parallel with VBA, implementations will always verify or enforce the VBA before checking any session state information or options. If a reported VBA LL2IP binding does not verify through a receiver's shim, its failure should not affect any sessions, active or not. Figure 27 presents the benefits of deploying both VBA and ND sessions together on the same local network.




```
* Hosts A & B use 'Strict' ND Session modes and the AGV IEM for VBA.

(1) Active ND session thwarts a malicious LLID impersonation attack.
  (a) Host C forces Host A off the link.
  (b) Host C then assumes the IP and LLID of Host A for interception
      with Host B.
  (c) Host C does not know the next-in-line ZKP token; ND sessions
      mandate that Host B deny packets after 1 minute without a
      valid Session Reachability Confirmation.

(2) VBA denies a bad LLID-to-IP binding and possible on-path attack.
  (x) Host D successfully engages Host B in an ND session.
      This is not hard to do since sessions do not verify bindings.
  (y) Host D reports a LLID that does not bind properly to its IP.
  (z) Host D is rejected as illegitimate and a possible attacker.
```

Figure 27. A few scenarios are detailed which demonstrate why integrating ND sessions and VBA together in the same IPv6 local network will serve to protect neighbors from falsification and impersonation attacks. Where the shortcomings of one proposal exist, another can assist.

6.2 Precomputing VBA Address Collisions

In networks without any RA Guard capabilities, malicious nodes have a chance to usurp the VB and to introduce their own abusive vouchers. Denial of service notwithstanding, malicious VBs are not able to further any goals aimed at ND Redirection attacks without infeasible time-memory tradeoffs. Control over the key derivation algorithm 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.

Consider a legitimate network using the Argon2d key derivation algorithm with a pseudo-random voucher seed, which is then successfully hijacked by a malicious neighbor. Using a known and static seed, the attacker can pre-compute a set of rainbow tables for all possible link-layer address bindings with a specific set of low-difficulty, computationally trivial Argon2d parameters. Selection of an iterations-count is determined at each network node and is not controllable by the attacker. 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.

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

In the case specific to MAC addresses as link-layer identifiers: if each result stored only the necessary 48 bits extracted from the derived key, 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

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 computation 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 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 scope value of FE80::/64. Any change of the subnet prefix value requires an entirely new set of data at 224 EiB in size.

While pre-calculation is not an issue in a practical sense, it is theoretically possible to generate these tables, even to a partial degree, to pre-compute some of the work required. Likewise, it is not inconceivable to be able to have them readily available in an indexed database at some arbitrary future time. Of course, such a future time must necessarily represent a fundamental difference from the technologies available at the time of this research. In the best

interest of future-proofing the work of Voucher-Based Addressing, and to evade the absurdity of these considerations altogether, all deployments should strongly consider using Router Advertisement Guarding per [44] to assist in preventing these attacks. Guarding against rogue Voucher Bearers disallows all illegitimate voucher hijacking and abates these concerns altogether, because the voucher seed will not be fixed.

6.3 Recognized Gaps & Issues

Voucher-Based Addressing is a straightforward address derivation methodology that produces outwardly random addresses from key components only known to neighbors. Thanks to the certainty of MAC address uniqueness on any functional local network, VBA builds atop a strong foundation for preventing subversive traffic redirection attacks where neighbors attempt to spoof other neighboring link-layer addresses. This same principle applies to the necessary local uniqueness of any link-layer addressing scheme. VBA can also function as specified in environments with a pool of static address assignments, using flexible IEMs and/or static Neighbor Cache entries to remain viable. However, the certainty of VBA security rests upon two important concepts that should be explored in more detail: trust on first use and active-node impersonation protection.

Trust on first use mechanisms, as applied to both ND sessions and VBA, are a considerably weak form of trust. This is especially true since the rendition employed by this research does not require any user interaction or configuration to establish the initial trust (i.e., the traditional manner by which TOFU is generally applied). Neither VBA nor ND sessions can be considered a fully trusted solution to the problem without some form of centralization and authority delegation, such as the SEND/CGA utilization of digital certificates, which this work

has sought to evade. This research consciously introduces two newly proposed NDP amendments with a recognized tradeoff of ‘soft security’, for the sake of increased simplicity and immediate, transparent practicality.

Another concern is inactive node impersonation, where nodes who go offline for one reason or another are directly impersonated. VBA is susceptible to this problem at any time because link-layer address impersonation on an insecure link layer defeats any effectiveness of validating LL2IP bindings during address resolution. ND sessions seek to mitigate this issue by requiring ZKP in the process of using RHCR, but between a receiving node’s REACHABLE and STALE cache states for its peer, anything can theoretically assume the link-layer address of the peer and still be considered valid.

One suggested but prohibitively costly fix for this is to enforce an extremely regular interval for Session Reachability Confirmations; almost to the point that any new data exchanges with session peers should always verify SRCs within ten seconds of one another. Increasing the rate at which NDSOs must be provided back and forth between Session Nodes does increase the randomness (but also the frequency) of the Root Hash revelation in RHCR. Again, it is not without its costs to increase the rate at which hash chain details must be formed and rotated. This is unfortunately an inherent problem with the methodology used by VBA and ND sessions; it should be further examined or improved in future research endeavors.

6.4 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.

Threats due to insecure link layers resulting in direct impersonation of neighbors are addressed by the employment of Neighbor Discovery Sessions. Though a secure link layer is not necessary for VBA to function, threat actors interested in stealthily intercepting packets and data can still take advantage of insecure link layers. Sessions mitigate this problem after each successive reachability confirmation is required and received, by ensuring the corresponding node must persist its known identity (i.e., ZKP information) between reports of NDAR transaction details.

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.

6.5 Simplicity, Privacy, & Flexibility

The introduction of VBA and ND sessions aims to produce a protocol amendment that is specifically tailored to solve the NDP Neighbor Redirection security vulnerability. As the

original goals of this research stated, this effort seeks to ensure the resulting solution maintains three core ideals: simplicity, privacy, and flexibility. In this brief section, the work of this research will be fairly evaluated using the same ‘measuring stick’ instantiated in Chapter 2, in order to determine the adoption potential of VBA and NDSO. A justification is fairly provided to explain why simplicity, privacy, and flexibility are indeed goals achieved by the introduction of these two independent protocols.

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 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. Voucher-Based Addressing is a simple scheme because it does not require huge, mandatory impacts to Neighbor Discovery that cause disruptions in the network or at the neighbor-to-neighbor level. With a mixture of Interface Enforcement Mode selections, a broadcasted Link Voucher, 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, modifiable with one simple per-interface setting (the IEM). The simplicity of ND sessions is arguable, however. While a subjective evaluation might indicate that Reverse Hash Chain Revealing is a convoluted and needless concept, it is in fact easy to implement and maintain once sessions have been initialized successfully. This is thanks to the independence that each hash chain preserves from one another on each side of an ongoing session. Sessions measurably achieve simplicity because they are low-impact overlays on top of ordinary NDP

traffic, only necessary based on the dictation of the Session Node Interface Configuration Modes involved during end-to-end NDP exchanges.

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 unfortunate afterthought of many proposals and research which would otherwise provide excellent protection. VBA manifests privacy in its ability to generate outwardly pseudo-random addresses to nodes off-link, while still providing enough information to direct neighbors to reconstruct and validate the address locally during NDAR transactions. 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 Link Voucher has not rotated, to choose a new interface identifier which shares no correlation to another chosen iterations count, defeating address tracking concerns. 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 current Link Voucher), because a hashing algorithm generates most of the final address suffix. Perhaps less prone to privacy violations, ND sessions are by nature private because NDSOs exist primarily in unicast messages to neighbors on-link. The NDSO messages themselves provide no identifying information about a sending node that cannot otherwise be known by already being a local neighbor.

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, especially if the item is end-to-end. Any proposal which severs that transition capability will never see adoption because forced compatibility cannot be considered ‘compatibility’ in the first place.

VBA and ND sessions both employ IEMs and ICMs to this purpose, respectively. 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 what specifications to adhere to. These same interface modes are defined on a sliding scale of strictness, where more lax solutions behave as though the proposal (VBA or ND sessions) is disabled and more strict solutions have no tolerance for neighbors that do not adhere to its requirements.

Chapter 7 Conclusion

7.1 Future Work

The addition of Voucher-Based Addressing and Neighbor Discovery Sessions to the standardized form of NDP constitutes two decoupled, complementary solutions to only one of the many problems NDP faces. As Chapter 2 has reviewed, NDP is subject to an array of exploitable weaknesses from two major categories: spoofing and flooding [6]. Some more specific issues falling into these categories include RA/RS flooding attacks, rogue routers, NA/NS flooding, illegal or spoofed redirections, RA spoofing, and neighbor spoofing. The application of VBA aims to mitigate neighbor spoofing, which is arguably one of the more threatening vulnerabilities of NDP that requires an end-to-end solution. This is because neighbor caching occurs at each network endpoint rather than being a more distributed process like the seeding of Router Advertisements. ND sessions simply fortify the protections against neighbor spoofing provided by VBA.

The introduction of these two synergistic works in this research concocts an optional amendment to Neighbor Discovery that is intentionally trivial and simple to implement. In a single introductory specification, however, there is no terse way to iterate all possibilities for the deployment contexts of the two solutions, whether applied together or separately. Future works should aim to evaluate a few key areas of both VBA and ND sessions: how they affect current security in networks using non-standard protocols (e.g., not using MAC addresses at the link layer), how their specifications can be extended to include various other situations and integration with alternative security mechanisms, and how they might overall not be sufficient

for their intended purposes. An analysis of how VBA and ND sessions might apply to the context of current NDP security concerns is also warranted. These topics could represent baseline research questions that might extend into the discovery of a better means by which the goals of this work may be accomplished. The subsections below will serve to initialize this conversation.

7.1.1 Voucher-Based Addressing

Deployments using DHCP

VBAs are designed primarily for SLAAC-based environments and thus no research or work has been done to examine their integration with DHCP servers. Future work might wish to add features into DHCP servers that support VBAs, using something like DHCP Snooping to ensure that only legitimate servers are delegating addresses to neighbors. Because of its centrality and responsibility, a DHCP server would also function well as the link VB if no link router has support available for VBA.

One notable change of generating VBAs on the server-side is that the ability for client nodes to self-determine an IC value dynamically is no longer available. Allowing nodes to choose their own ICs affords them the ability (1) to randomize the value according to their own implementations and (2) to preserve a Preferred Iterations Count. In some future research, DHCP client options might be amended to allow a client to request a certain 'security level' or IC dynamically. Such an option could also present an opportunity to exchange other information about client preferences or other important VBA details.

Neighbor Discovery Proxies

RFC 4389 [64] specifies a Neighbor Discovery Proxy as a network-layer device or software used to provide a presence for nodes who have gone off-link or have always been residents off-link. This is supposed to stand in lieu of a classic link-layer bridge.

Due to link-layer binding, VBA does not support ND proxying in its design unless the proxy is also able to spoof the LLIDs of all the target nodes. These spoofed LLIDs would need to appear on the interface attached to the link on which it must receive and answer ND packets. One solution to enabling ND proxy while keeping the rest of the network secure might be to apply the same strategy used for static addressing and create manual cache entries. Another solution might be to enable the AVGL IEM on the nodes which are required to transact with the proxy. Support for ND proxies is not very well defined by this specification, as it conflicts with one of its primary goals. Future experimentation may wish to uncover ways by which this could be non-disruptively integrated into VBA-enabled networks.

Certifying Link Vouchers

Link Vouchers are susceptible to impersonation despite the use of asymmetric cryptography in signing their details. Once a node acquires an LV and reads the public key details of its newly-active voucher, it becomes 'locked' to this key until some expiration or transition event occurs later. Subsequently, the node will not accept LVs from senders who are not employing the same key information in their option signatures, unless the current LV expires or a VHA is issued. However, the initial exchange between a VB and a neighbor is still vulnerable, because any malicious node on-link could craft a

public key for its own LV and advertise it, if the node is not first blocked by infrastructure-based solutions like RA-Guard.

Section 6 of RFC 3971 [16] dictates the use of Public Key Infrastructure to ensure communication is genuine between hosts and routers, and that each router is authorized to provide router information. Trust anchors are used to determine whether a certificate presented by a router validates its role in SEND. If the router presents a certificate that is trusted by the anchor, then on-link hosts sharing the same trust anchor must consider it as legitimate. The same validation of certification paths can also be used to verify ND RSA Signature options between on-link hosts. Establishing certification paths that validate SEND traffic is done through the use of two new ICMP messages presented by the SEND specification:

- Certification Path Solicitation (ICMP type 148). Solicits routers with a set of trust anchors and expects an advertisement including certificates authorized by one or more of the trust anchors.
- Certification Path Advertisement (ICMP type 149). Routers use these to respond to valid solicitations indicating the need for one or more certificate(s) from a set of specified Trust Anchor options.

Future additions to this research may invoke these ICMP options to integrate with public-key signatures appearing on LVs. This might include amendments to the ND LV option that would extend the field by some extra length in order to convey trust anchor or certification path information. Similar amendments might simply consider adding

certification path information to LVs and letting each neighbor use their own certificate stores to validate them.

While VBA vehemently seeks to dodge the complexities introduced by certificates and trust anchors, this supporting infrastructure might be crucial for first-contact trust assurance wherever RA-Guard or similar mechanisms cannot be used to protect the link from malicious VBs. This is likely a much more performant use of certification paths than SEND, simply because the trust of a public key only needs to be verified one time at each receiver when an initial LV, or LV handoff packet, is received and stored. Such an amendment to LV options is beyond the scope of this work.

It is instead suggested as future work to design an optional certification path amendment for VBA, to make them a more secure, isolated, and end-to-end solution. This is opposed to only relying on automatic Trust on First Use mechanisms. Adding these certification paths will draw VBA closer into the territory of more canonical solutions which already exist for ND security issues. Again, care should be taken to keep them optional to preserve the principles of flexibility and simplicity.

7.1.2 Neighbor Discovery Sessions

The concept of Neighbor Discovery sessions is a novel suggestion that permits cheap proof of knowledge authentication to be applied at a level considerably lower than the transport layer. However, as discussed prior, it comes with many risks that might not outweigh its reward, especially considering a similar assurance is alternatively provided by CGAs and RSA Signature NDP options. Future research may wish to explore various means to secure the transitions used

between each Session Node's Hash Chains. The direct revelation of Root Hashes is risky business that puts the integrity of an established session at risk every time it is revealed, since doing so is essentially done 'in the clear'.

Future research may also wish to explore how Trust on First Use could be a more one-shot transactional experience between two nodes that establishes some baseline for future communication. For example, rather than using Reverse Hash Chain Revealing or Zero-Knowledge Proofs at all, some form of symmetric key exchange might happen during the initial communication between two neighbors. Since Trust on First Use is already accepting the first communication with a neighbor as 'secure' enough to instantiate a new session without further initial proof, the same level of security will apply to the first communication establishing a known symmetric key. Doing so will prevent the need for irregularly revealing a risky 'password' (i.e., Root Hash) when the current token chain is sufficiently depleted, to move to a new set of renewed credentials. However, using a symmetric key could come with risks of mid-session key compromise or session staleness if the key is not regularly rotated somehow. An additional concern is how to recover from session lockouts if each Session Node were to use a symmetric key value. Again, future research might explore the implications of forming such a strategy over choosing to use ZKP at all.

7.2 Towards Robust NDP Security Solutions

The security weaknesses faced by Neighbor Discovery should be a much more pressing concern to security research and practical, security-conscious network administrators alike. Since the typical architecture and model of any computer network rests upon a layering concept, implying there are fundamental network layers which increase in abstraction as the layers

increase, the existence of trivially exploitable vulnerabilities at a lower layer diminishes the security of layers atop it. NDP exists somewhere near the baseline link-layer abstraction to provide a sort of glue between link-layer discovery mechanisms and the Internet Protocol network ‘above’ it. Thus, at such a low level, few mitigations can guarantee any security in more abstract, higher layers such as application- and transport-stack protocols; risk is always present and lurking.

Transport Layer Security (TLS) and other Secure Socket Layer (SSL) technologies relying on digital certificates and certificate authorities with trust anchors can provide adequate protection against sniffing of higher-layer packets. The same can be said for any type of end-to-end encryption model guaranteeing confidentiality, but not all traffic is guaranteed to be securely encrypted. Any degree of falsification at the link-layer or the network layer can wreak havoc on the fidelity of communication when a malicious threat actor is transparently snooping through unencrypted packets and modifying them. While the attack surface at the link-local scope is decreasing over time as encrypted communications become more commonplace, there is still a significant amount of cleartext data exchange occurring in local networks. And if a neighbor cannot be trusted, who else can be but oneself?

If the non-confidentiality of communication in local networks was not enough risk, there is also the separate risk of protocol abuse leading to denial-of-service attacks. Through various means and prepackaged techniques, a threat actor can exploit either native Neighbor Discovery or SEND systems with ease to lay waste to local networks from a single controlled node. Not only can these attacks cause neighbors to shut down from an overwhelmed processor or to disconnect from the network due to flooding: they are sometimes the direct cause of Neighbor Redirection attacks in the first place.

There is no tangible velocity towards the adoption of a robust security solution that can provide any degree of surety beyond native NDP in local IPv6 networks. It seems as though these security weaknesses have long been covered and patched over 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 is already 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 weaknesses have either remained 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 robust 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 and Neighbor Discovery Sessions seek to establish themselves as two proposals which highlight these attributes and individually target their specific goals, rather than trying to remediate all complex issues of the protocol at once. Until each specific solution provided by future researchers can provide these three facets as headline items, the security of NDP will be left in its current hellish limbo.

7.3 Final Thoughts

Voucher-Based Addresses bind input link-layer addresses to sets of private, deterministic output addresses that can be reconstructed by verifying neighbors on the local link to confirm a

reported binding in NDP transactions. Address generation and verification is constrained to some initial conditions by a Link Voucher, in addition to a window of computation time by usage of key derivation functions, based on parameters agreed upon by peers. This consensus on VBA generation parameters acts as an enforceable handshake between neighbors during the NDP Address Resolution process. This process is not actively falsifiable without introducing obvious network disruptions, thus preventing malevolent attackers on the local network from subversively intercepting or modifying traffic between neighbors.

Likewise, Neighbor Discovery Sessions introduce an element of identity verification to NDP transactions by enforcing an end-to-end session overlay between two nodes. They were conceived to prevent the 'killed node' issue–noted as a glaring weakness of VBA–and they do so by employing a Reverse Hash Chain Revealing authentication mechanism to provide neighbors with on-demand Zero-Knowledge Proofs. The requirement for implementing sessions mandates more caching overhead to accommodate session states, slightly more NDP traffic in the form of NUD NS or NA messages, and a single new NDP Option with an uncomplicated structure, enforceable by neighbors according to their Interface Configuration Modes.

Voucher-Based Addressing and Neighbor Discovery Sessions each describe simple, unique, transparent, privacy-conscious, and flexible standards for use in Neighbor Discovery Protocol transactions. This proposal is best suited for a deployment which cohesively integrates and enables both technologies on capable devices, according to the transition guidelines and implementation samples provided. Both items clearly express the problems they are appointed to solve along with their use cases, and thus may be applied individually on a case-by-case basis. Though the solutions presented in this research maintain a high level of abstraction and do not beget very much concrete and practical evidence of their practicalities, there is a 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 security and privacy issues faced by NDP–though it may perhaps be a suitable baseline–but 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 these concepts and frameworks 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.

There are yet-unfulfilled plans for decoupled implementations of VBAs and NDSOs in a public fork of the Linux kernel. This stands in addition to tentative concrete and technical specifications which will be submitted to the IETF as Internet Drafts for Experimental RFC candidacy. This is a process which will elevate this research to the attention of other field experts directly involved in the development and maintenance of IPv6 and its adjacent protocols. Other researchers are encouraged to critique, reconstruct, or review this work and to explore other more modern approaches to securing the NDP Address Resolution process from on-path attackers. Finally, research should continue to apply alternative approaches to link-layer address resolution and not settle for existing, stale solutions; making full use especially of the generous amount of space IPv6 addresses grant for embedding helpful authentication details.

To conclude, the broad goal of this research is to define an alternative to SEND and CGAs which can work synergistically to solve the address ownership problem in local IPv6 networks. It pragmatically harmonizes three important attributes: privacy, flexibility, and simplicity. This work has set out to molt the NDP security paradigm that has continued to

depend upon sophisticated Public Key Infrastructure, limiting infrastructure-only protocols, unwieldy asymmetric cryptography, and centralized address registration authorities. This research has therefore defined a decentralized and empirical approach that mostly evades the aforementioned tar pits and creates a new perspective on NDP security problems with tangible solutions. Lastly, VBAs and NDSOs measurably solve problems of address ownership in the vast majority of conceivable IPv6 deployment scenarios, in a way that is privacy-focused, wholly transparent to end users, and not at all disruptive to incompatible neighbors.

Appendix 1. Sample Implementation for VBA Generation & Verification

This sample implementation of VBA generation and verification can be used as a reference to better understand the mechanisms used to service and validate addresses. Note that, despite this implementation's use and assertion of a 'mac_address' array of bytes, VBAs are not strictly coupled to using MAC addresses as bindings from the link layer (Section 3.2.4). As long as both hosts in the NDP exchange are using the same link-layer addressing schema, or have communicated such information prior to the verification process, verification will not fail due to any form of confusion in computation. MAC addresses are chosen for this figure due to their ubiquity, but any length salt is theoretically allowable provided the entirety of the link-layer address from the parsed NDAR transaction is placed into the beginning of the aggregate 'salt' array of bytes.

```
uint64_t compute_address_hash_suffix(uint8_t *voucher_seed,
                                     uint8_t *mac_address,
                                     uint16_t iterations,
                                     VbaAlgorithm algorithm) {
        /*
        * Initialize a storage buffer for the resulting hash.
        *   All hashes requested for generation by VBAs will be 32 bytes in length,
        *   despite the algorithm only using the leading 64 bits.
        */
        size_t res_buffer_size = 32;
        uint8_t res_buffer[32] = {0};

        /*
        * The 'password' is always the 128-bit voucher seed. The salt is a combination
        *   of a LLADDR (MAC in this case) + 'vba' + the 64-bit subnet prefix (or
        *   left-most 64 bits of the unicast address that will be built).
        *   This example application uses "fe80::" for the subnet prefix.
        *   All values are big-endian (network byte order).
        */
        uint8_t salt[VOUCHER_SALT_LENGTH] = {
                /* MAC (6 bytes) */
                0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                /* Static 'vba' (3 bytes) */
                'v','b', 'a',
                /* Subnet prefix (8 bytes */
                0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        };
```

```
        memcpy(&salt[0], &mac_address[0], 6);
        /* memcpy(&salt[9], &subnet_prefix[0], 8); */

        /*
        * Many of the 'fixed' values below are actually disseminated from
        *   voucher details, such as the Argon2 Memory cost and parallelism degree.
        *
        * This sample, per results in experimental data, tries to use values which
        *   result in somewhat consistent computation times across varying levels
        *   of computing power.
        */
        switch (algorithm) {
        case PBKDF2:
                PKCS5_PBKDF2_HMAC((const char *) voucher_seed,
                                VOUCHER_SEED_LENGTH,
                                salt,
                                VOUCHER_SALT_LENGTH,
                                iterations * PBKDF2_ITERATIONS_FACTOR,
                                EVP_sha256(),
                                res_buffer_size,
                                res_buffer);
                break;
        case ARGON2:
                argon2d_hash_raw((iterations >> 8) + 1,
                                ARGON_MEMORY_SIZE,   /* LV parameter; N */
                                ARGON_PARALLELISM,   /* LV parameter; p */
                                voucher_seed,
                                VOUCHER_SEED_LENGTH,
                                salt,
                                VOUCHER_SALT_LENGTH,
                                res_buffer,
                                res_buffer_size);
                break;
        case SCRYPT:
                /* https://www.tarsnap.com/scrypt.html */
                libscrypt_scrypt(voucher_seed,
                                VOUCHER_SEED_LENGTH,
                                salt,
                                VOUCHER_SALT_LENGTH,
                                MAX(1 << (iterations & 0x00FF), 2)
                                << SCRYPT_SCALING_FACTOR,   /* N */
                                MAX((iterations & 0xFF00) >> 4, 16)
                                << SCRYPT_SCALING_FACTOR,   /* r */
                                1,   /* p */
                                res_buffer,
                                res_buffer_size);
                break;
        default:
                fprintf(stderr, "Unknown algorithm type.\n");
                return -1;
        }

        /* Always use the first 8 bytes (64 bits) of the resulting hash. */
        uint64_t hash_H = *((uint64_t *) &res_buffer[0]);

        /* Get the first two bytes of the current Link Voucher seed. */
        uint16_t first_seed_hextet = (voucher_seed[0] << 8) | voucher_seed[1];
        printf("FH: %04x /// IC: %04x\n", first_seed_hextet, iterations);

        /* Compute the 'Z' value and overwrite the first hextet of hash 'H'. */
        return
            ((uint64_t)(~(iterations ^ first_seed_hextet)) << 48)
                | (0x0000FFFFFFFFFFFF & hash_H);
}

/* A method used to verify a VBA. */
bool verify_address_suffix(uint64_t suffix,
                           link_voucher_t *lv,
                           uint8_t *mac_address
                           uint8_t *subnet_prefix)
{
```

```
    /* First extract the 'L' value from 'Z' in the input SUFFIX. */
    uint16_t first_seed_hextet = (lv->seed[0] << 8) | lv->seed[1];
    uint16_t iterations =
        (uint16_t)((~(suffix ^ first_seed_hextet) >> 48) & 0xFFFF);

    /* Now independently compute SUFFIX like the generator does. */
    uint64_t computed_suffix = compute_address_suffix(lv,
                                                      mac_address,
                                                      subnet_prefix,
                                                      iterations);

    /* If the independent calculation yields the same suffix as the
        input, then the binding for this LV is legitimate. */
    return computed_suffix == suffix;
}
```

Appendix 2. Sample Implementation for VBA Collision Detection

A sample implementation for attempting to find VBA collisions is given in this section. The “pthread” library is used to generate up to any precompiled number of threads to operate simultaneously, attempting to discover collisions with one of two different methods: random and ordered. As its name suggests, the random method blindly selects a random sequence of 48 bits to input into the VBA generation algorithm. Conversely, the ordered method assigns blocks of MAC addresses to attempt for each thread, with no MAC address overlap, thus allowing the entire 48-bit space to be slowly and simultaneously walked by the program.

The data structure of type *worker_ctx_t* holds all necessary context to be passed to child threads upon instantiation. Based on the operation type, some fields of the context object may not be used. Any thread discovering a collision-generating input MAC address will set the reference to the *match_found_sync_bool* in the main thread of the program to a True value, indicating that some worker thread has successfully mined and produced a collision-generating MAC address.

```
/* Thread context data passed to each started thread. */
typedef struct _worker_ctx {
        pthread_t thread_handle;
        unsigned int id;
        uint64_t starting_mac;
        uint64_t ending_mac;
        uint8_t* voucher_seed;
        uint64_t legitimate_suffix;
        uint16_t iterations;
        VbaAlgorithm algorithm;
        bool* match_found_sync_bool;
} worker_ctx_t;


/*
 * This is it. Attack the protocol and see how long it takes to find a collision.
 *   Since iteration counts are fixed into a node's address, that value must
```

```
 *   remain static throughout the duration of the test.
 *
 * Therefore, if a node chooses a high iteration count, it is prohibitively costly
 *   for attackers to determine a 48-bit collision in any reasonable time.
 *
 * Optimizations to this cannot come from time-memory tradeoffs, but can possibly
 *   come from restricting the input set of MAC addresses to those allowed for
 *   normal network nodes (e.g., skipping link-layer Multicast and Broadcast MACs).
 *
 * For the sake of attack completeness, invalid MAC ranges will be SKIPPED when
 *   attempting to find a collision through the "Ordered" methodology.
 *   See: https://www.rfc-editor.org/rfc/rfc7042#section-2
 */
static inline void
_find_collisions(VbaAlgorithm algorithm)
{
    bool is_random_match = false;
    bool is_ordered_match = false;

    for (int i = 0, j = 0; i < FIXED_ITERS_COUNT; ++i, j += 2) {
        uint16_t iterations = _fixed_iter[i];

        printf("For '0x%04x' (%d) iterations.\n", iterations, iterations);

        uint64_t legitimate_hash =
                compute_address_hash_suffix(_stable_voucher_seed,
                                            _stable_mac_address,
                                            iterations,
                                            algorithm);

        uint64_t legitimate_suffix = build_address_suffix(iterations,
                                                           legitimate_hash,
                                                           _stable_first_seed_hextet);

        printf("\tGot address: ");
        print_lladdr_from_suffix(legitimate_suffix);

        printf("\n\tNow searching for a collision...\n\t\tRandom... \n");

        for (unsigned int x = 0; x < THREAD_COUNT; ++x) {
                worker_ctx_t thread_ctx = {
                        {0},
                        x + 1,
                        0,
                        0,
                        &_stable_voucher_seed[0],
                        legitimate_suffix,
                        iterations,
                        algorithm,
                        &is_random_match
                };
                memcpy(&_thread_contexts[x], &thread_ctx, sizeof(worker_ctx_t));

                pthread_create(&(_thread_contexts[x].thread_handle),
                               NULL,
                               _thread_routine_collision_random,
                               &_thread_contexts[x]);
        }
        for (int x = 0; x < THREAD_COUNT; ++x)
                pthread_join(_thread_contexts[x].thread_handle, NULL);

        if (is_random_match)
                printf("\n\n=== A RANDOM MAC ADDRESS FOUND A COLLISION! ===\n\n");

        /* ================================= */
        printf("\n\t\tOrdered... \n");

        for (unsigned int x = 0; x < THREAD_COUNT; ++x) {
                worker_ctx_t thread_ctx = {
                        {0},
                        x + 1,
```

```
                        x * MACS_PER_THREAD,
                        ((x + 1) * MACS_PER_THREAD) - 1,
                        &_stable_voucher_seed[0],
                        legitimate_suffix,
                        iterations,
                        algorithm,
                        &is_ordered_match
                };
                memcpy(&_thread_contexts[x], &thread_ctx, sizeof(worker_ctx_t));

                pthread_create(&(_thread_contexts[x].thread_handle),
                               NULL,
                               _thread_routine_collision_ordered,
                               &_thread_contexts[x]);
        }
        for (int x = 0; x < THREAD_COUNT; ++x)
                pthread_join(_thread_contexts[x].thread_handle, NULL);

        if (is_ordered_match)
                printf("\n\n=== AN ORDERED MAC ADDRESS FOUND A COLLISION! ===\n\n");

        printf("\n\n");
        }
}
```

Appendix 3. Sample Outputs from VBA Generation

Sample implementation code was run from Appendix 1 to run various bench tests through experimentation with Voucher-Based Addressing. This appendix outputs some select snippets of raw logging information from the compiled and optimized runtime, showing how addresses are generated based on their various inputs. Cross-sections of the final VBA can be analyzed to reveal a few key ideas from the work in this research.

The Z value of each address prefix can be easily reversed to find the input Z value by following the Z' function mentioned in Chapter 4. The changing seed shows how the input L value is masked to a more privacy-focused value despite the iterations-count values being the same across the three different algorithms. Each generated address is significantly different from adjacent addresses using almost the exact same inputs, as a testament to VBA's promised privacy and randomness. Lastly, input MAC addresses and final VBA prefix details can be inserted into each KDF algorithm (with the iterations count at each step) to calculate and prove the VBA generation process by hand.

```
Algorithm #1 iterations '0x0100' (actual [x256]: 65536) ...
Result: 0x3657223f6626464f
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::3657:223f:6626:464f
    Duration:          16616 us   (          16.616000 ms)

Algorithm #1 iterations '0x0200' (actual [x256]: 131072)
Result: 0x3557e4070700f192
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::3557:e407:0700:f192
    Duration:          29192 us   (          29.192000 ms)

Algorithm #1 iterations '0x0300' (actual [x256]: 196608) ...
Result: 0x3457aa0fad5b66cd
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::3457:aa0f:ad5b:66cd
    Duration:          63230 us   (          63.230000 ms)
```

```
. . . . .

Algorithm #1 iterations '0xfd00' (actual [x256]: 16580608) ...
Result: 0xca57c836e7269b60
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::ca57:c836:e726:9b60
    Duration:          4873968 us   (         4873.968000 ms)

Algorithm #1 iterations '0xfe00' (actual [x256]: 16646144) ...
Result: 0xc957218fc8d92d00
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::c957:218f:c8d9:2d00
    Duration:          4742346 us   (         4742.346000 ms)

Algorithm #1 iterations '0xff00' (actual [x256]: 16711680) ...
Result: 0xc85780b7de0a9fd3
    MAC Address: 11-22-33-44-55-66    Seed: 0xc8a87936a71bdca82bce74c9c2c54648
    Final IPv6 Addr (lladdr): fe80::c857:80b7:de0a:9fd3
    Duration:          4680291 us   (         4680.291000 ms)

//////////////////////////////////////////

Algorithm #2 iterations '0x0100' (actual: 256) ...
Result: 0x675f436c9960b49f
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::675f:436c:9960:b49f
    Duration:          29723 us   (           29.723000 ms)

Algorithm #2 iterations '0x0200' (actual: 512) ...
Result: 0x645f15d18526c18c
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::645f:15d1:8526:c18c
    Duration:          43377 us   (           43.377000 ms)

Algorithm #2 iterations '0x0300' (actual: 768) ...
Result: 0x655fc3b3f959a15a
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::655f:c3b3:f959:a15a
    Duration:          58880 us   (           58.880000 ms)

. . . . .

Algorithm #2 iterations '0xfd00' (actual: 64768) ...
Result: 0x9b5fb16138b25099
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::9b5f:b161:38b2:5099
    Duration:          1137549 us   (         1137.549000 ms)

Algorithm #2 iterations '0xfe00' (actual: 65024) ...
Result: 0x985f70cd7f4aac5c
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::985f:70cd:7f4a:ac5c
    Duration:          1197668 us   (         1197.668000 ms)

Algorithm #2 iterations '0xff00' (actual: 65280) ...
Result: 0x995fba07c7bac059
    MAC Address: 11-22-33-44-55-66    Seed: 0x99a03c53d49ec845e0a0e113788c517a
    Final IPv6 Addr (lladdr): fe80::995f:ba07:c7ba:c059
    Duration:          1172262 us   (         1172.262000 ms)

//////////////////////////////////////////

Algorithm #3 iterations '0x0100' (actual: 256) ...
Result: 0xe858206d745e95f6
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::e858:206d:745e:95f6
    Duration:          8530 us   (           8.530000 ms)

Algorithm #3 iterations '0x0200' (actual: 512) ...
Result: 0xeb58c77fb102b5dd
```

```
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::eb58:c77f:b102:b5dd
    Duration:          17814 us   (          17.814000 ms)

Algorithm #3 iterations '0x0300' (actual: 768) ...
Result: 0xea58582fa5d0b1e5
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::ea58:582f:a5d0:b1e5
    Duration:          27097 us   (          27.097000 ms)

. . . . .

Algorithm #3 iterations '0xfd00' (actual: 64768) ...
Result: 0x1458137871909b3c
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::1458:1378:7190:9b3c
    Duration:          2521439 us   (         2521.439000 ms)

Algorithm #3 iterations '0xfe00' (actual: 65024) ...
Result: 0x1758496315177377
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::1758:4963:1517:7377
    Duration:          2677193 us   (         2677.193000 ms)

Algorithm #3 iterations '0xff00' (actual: 65280) ...
Result: 0x1658ae0dbe516501
    MAC Address: 11-22-33-44-55-66    Seed: 0x16a78a4fc33f152504ef7f1a26077f65
    Final IPv6 Addr (lladdr): fe80::1658:ae0d:be51:6501
    Duration:          2601904 us   (         2601.904000 ms)
```

References

[1] 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.

[2] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating MAC Address Randomization Through Timing Attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, in WiSec '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 15–20. doi: 10.1145/2939918.2939930.

[3] J.-C. Zúñiga, C. J. Bernardos, and A. Andersdotter, "Randomized and Changing MAC Address state of affairs," Internet Engineering Task Force, Internet-Draft draft-ietf-madinas-mac-address-randomization-12, Feb. 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-madinas-mac-address-randomization/12/

[4] T. Aura, "Cryptographically Generated Addresses (CGA)," in *Information Security*, C. Boyd and W. Mao, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 29–43.

[5] 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.

[6] F. Najjar, Q. Bsoul, and H. Al-Refai, "An analysis of neighbor discovery protocol attacks," *Computers*, vol. 12, no. 6, p. 125, Jun. 2023, doi: 10.3390/computers12060125.

[7] 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.

[8] P. Sumathi, S. Patel, and A. Prabhakaran, "A Survey on IPv6 Secure Link Local Communication Models, Techniques and Tools," 2017.

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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.

[19] A. S. A. Mohamed Sid Ahmed, R. Hassan, and N. E. Othman, “IPv6 Neighbor Discovery Protocol Specifications, Threats and Countermeasures: A Survey,” *IEEE Access*, vol. 5, pp. 18187–18210, 2017, doi: 10.1109/ACCESS.2017.2737524.

[20] J. W. Bos, O. Özen, and J.-P. Hubaux, “Analysis and Optimization of Cryptographically Generated Addresses,” in *Information Security*, P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 17–32.

[21] A. Alsadeh, H. Rafiee, and C. Meinel, “Cryptographically Generated Addresses (CGAs): Possible Attacks and Proposed Mitigation Approaches,” in *2012 IEEE 12th International Conference on Computer and Information Technology*, 2012, pp. 332–339. doi: 10.1109/CIT.2012.84.

[22] J. L. Shah and J. Parvez, “IPv6 Cryptographically Generated Address: Analysis and Optimization,” *AICTC 16 Proc. Int. Conf. Adv. Inf. Commun. Technol. Comput.*, Jan. 2016, doi: 10.1145/2979779.2979781.

[23] L. Group and others, “arpwatch, the Ethernet monitor program; for keeping track of ethernet/ip address pairings,” *Last Accessed Sept.*, vol. 17, 2016.

[24] F. Najjar, M. M. Kadhum, and H. El-Taj, *Detecting Neighbor Discovery Protocol-Based Flooding attack using machine learning techniques*. 2016. doi: 10.1007/978-3-319-32213-1{_.

[25] N. Omar and S. Manickam, “Rule-Based SLAAC Attack Detection Mechanism,” in *Advances in Cyber Security*, M. Anbar, N. Abdullah, and S. Manickam, Eds., Singapore: Springer Singapore, 2021, pp. 435–449.

[26] A. Kukec, M. Bagnulo, and M. Mikuc, “SEND-based source address validation for IPv6,” in *2009 10th International Conference on Telecommunications*, 2009, pp. 199–204.

[27] M. Bagnulo and A. Garcia-Martinez, “SEcure Neighbor Discovery (SEND) Source Address Validation Improvement (SAVI),” no. 7219. in Request for Comments. RFC Editor, May 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7219

[28] J. Wu, J. Bi, M. Bagnulo, F. Baker, and C. Vogt, “Source Address Validation Improvement (SAVI) Framework,” no. 7039. in Request for Comments. RFC Editor, Oct. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc7039

[29] F. Gont, S. Krishnan, D. T. Narten, and R. P. Draves, “Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6,” no. 8981. in Request for Comments. RFC Editor, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8981

[30] 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

[31] J. Ullrich and E. Weippl, “Privacy is Not an Option: Attacking the IPv6 Privacy Extension,” in *Research in Attacks, Intrusions, and Defenses*, H. Bos, F. Monrose, and G. Blanc, Eds., Cham: Springer International Publishing, 2015, pp. 448–468.

[32] A. Cooper, F. Gont, and D. Thaler, “Security and Privacy Considerations for IPv6 Address Generation Mechanisms,” no. 7721. in Request for Comments. RFC Editor, Mar. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7721

[33] F. Gont, A. Cooper, D. Thaler, and W. (Shucheng) LIU, “Recommendation on Stable IPv6 Interface Identifiers,” no. 8064. in Request for Comments. RFC Editor, Feb. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8064

[34] S. Groat, M. Dunlop, R. Marchany, and J. Tront, “The privacy implications of stateless IPv6 addressing,” *CSIIRW 10 Proc. Sixth Annu. Workshop Cyber Secur. Inf. Intell. Res.*, Apr. 2010, doi: 10.1145/1852666.1852723.

[35] 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.

[36] S. Praptodiyono, I. H. Hasbullah, M. Anbar, R. K. Murugesan, and A. Osman, “Improvement of address resolution security in IPv6 local network using trust-ND,” *TELKOMNIKA Indones. J. Electr. Eng.*, vol. 13, no. 1, pp. 195–202, 2015.

[37] I. H. Hasbullah, M. M. Kadhum, Y.-W. Chong, K. Alieyan, A. Osman, and Supriyanto, “Timestamp utilization in Trust-ND mechanism for securing Neighbor Discovery Protocol,” in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016, pp. 275–281. doi: 10.1109/PST.2016.7906974.

[38] A. Al-Ani, A. K. Al-Ani, S. A. Laghari, S. Manickam, K. W. Lai, and K. Hasikin, “NDPsec: Neighbor Discovery Protocol Security Mechanism,” *IEEE Access*, vol. 10, pp. 83650–83663, 2022, doi: 10.1109/ACCESS.2022.3196028.

[39] A. S. Ahmed, R. Hassan, F. Qamar, and M. Malik, “IPV6 Cryptographically Generated Address: Analysis, Optimization and Protection,” *Comput. Mater. Contin. Mater. Contin. Print*, vol. 68, no. 1, pp. 247–265, Jan. 2021, doi: 10.32604/cmc.2021.014233.

[40] A. S. Ahmed, R. Hassan, and N. E. Othman, “Secure neighbor discovery (SeND): Attacks and challenges,” in *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, 2017, pp. 1–6. doi: 10.1109/ICEEI.2017.8312422.

[41] T. Zhang and Z. Wang, “Research on IPv6 Neighbor Discovery Protocol (NDP) security,” in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2032–2035. doi: 10.1109/CompComm.2016.7925057.

[42] J. L. Shah, “Secure Neighbor Discovery Protocol: Review and Recommendations,” *Int. J. Bus. Data Commun. Netw.*, vol. 15, no. 1, pp. 71–87, Jan. 2019, doi: 10.4018/ijbdcn.2019010105.

[43] 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

[44] 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

[45] J. Linkova, “Gratuitous Neighbor Discovery: Creating Neighbor Cache Entries on First-Hop Routers,” no. 9131. in Request for Comments. RFC Editor, Oct. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9131

[46] N. Moore, “Optimistic Duplicate Address Detection (DAD) for IPv6,” no. 4429. in Request for Comments. RFC Editor, Apr. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4429

[47] R. Asati, H. Singh, W. Beebee, C. Pignataro, E. Dart, and W. George, “Enhanced Duplicate Address Detection,” no. 7527. in Request for Comments. RFC Editor, Apr. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7527

[48] N. H. Walfield and W. Koch, “TOFU for OpenPGP,” *EuroSec 16 Proc. 9th Eur. Workshop Syst. Secur.*, Apr. 2016, doi: 10.1145/2905760.2905761.

[49] 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.

[50] 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

[51] 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

[52] 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

[53] 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

[54] 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

[55] S. Venaas and T. Chown, “Rogue IPv6 Router Advertisement Problem Statement,” no. 6104. in Request for Comments. RFC Editor, Feb. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6104

[56] J. Postel, “User Datagram Protocol,” no. 768. in Request for Comments. RFC Editor, Aug. 1980. [Online]. Available: https://www.rfc-editor.org/info/rfc768

[57] 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

[58] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, “Recommendation for password-based key derivation, part 1: storage applications,” National Institute of Standards and Technology, Jan. 2010. doi: 10.6028/nist.sp.800-132.

[59] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, “Moderately hard, memory-bound functions,” *ACM Trans. Internet Technol.*, vol. 5, no. 2, pp. 299–327, May 2005, doi: 10.1145/1064340.1064341.

[60] Q. H. Dang, “Recommendation for applications using approved hash algorithms,” National Institute of Standards and Technology, Jan. 2012. doi: 10.6028/nist.sp.800-107r1.

[61] L. Lamport, “Password authentication with insecure communication,” *Commun. ACM*, vol. 24, no. 11, pp. 770–772, Nov. 1981, doi: 10.1145/358790.358797.

[62] P. Nikander, *Denial-of-Service, address ownership, and early authentication in the IPV6 world*. 2002. doi: 10.1007/3-540-45807-7{_.

[63] K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota, “Birthday Paradox for Multi-collisions,” in *Information Security and Cryptology – ICISC 2006*, M. S. Rhee and B. Lee, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 29–40.

[64] M. Talwar, D. Thaler, and C. Patel, “Neighbor Discovery Proxies (ND Proxy),” no. 4389. in Request for Comments. RFC Editor, Apr. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4389