Published as: A. AlSa'deh and C. Meinel,"Secure Neighbor Discovery: Review, Challenges, Perspectives, and Recommendations," IEEE Security & Privacy Magazine, vol. 10, no. 4, pp. 26 –34, Aug. 2012.

Secure Neighbor Discovery:

Review, Challenges, Perspectives, and Recommendations

Ahmad AlSa'deh and Christoph Meinel | Hasso-Plattner-Institut

Secure Neighbor Discovery is designed as a countermeasure to Neighbor Discovery Protocol threats. The authors discuss Secure Neighbor Discovery implementation and deployment challenges and review proposals to optimize it.

The Neighbor Discovery Protocol (NDP), one of the main protocols in the IPv6 suite, comprises Neighbor Discovery for IPv6 (Request for Comments [RFC] 4861[1]) and IPv6 stateless address autoconfiguration (SLAAC).[2] It's used for several critical functionalities, such as discovering nodes on the same link, determining link-layer addresses, detecting duplicate addresses, finding routers, and maintaining reachability information about paths to an active neighbor. In addition, NDP plays a crucial role in mobile IPv6 (MIPv6) networks, eliminating the need for foreign agents and allowing mobile nodes to join new foreign networks.

However, NDP is prone to critical attacks.[3] It assumes that all nodes on the link trust each other, but this assumption doesn't hold for several scenarios, such as over a wireless network, in which anyone can join a local link with minimal or no link-layer authentication. Consequently, malicious users could impersonate legitimate nodes by forging NDP messages to generate attacks. As a result, RFC 3971, "Secure Neighbor Discovery (SEND)," became a standard.[4] SEND uses cryptographically generated addresses (CGAs),[5] a digital signature, and an X.509 certification to protect NDP. SEND was designed to ensure message integrity, prevent IPv6 address theft and replay attacks, and provide a mechanism to verify routers' authority.

Although SEND is a promising technique to protect NDP and make IPv6 a safe protocol, its deployment isn't easy. SEND lacks mature implementations by network device manufacturers and operating system developers. It's compute intensive and bandwidth consuming. Moreover, SEND itself can be vulnerable to some attacks. We discuss these implementation and deployment challenges and provide some directions and proposals for facilitating SEND deployment in IPv6 networks, especially for devices with limited resources.

Neighbor Discovery Protocol

NDP is part of the Internet Control Message Protocol for IPv6 (ICMPv6)[6] and uses ICMPv6 message format. It was designed in this position of the IP protocol stack for simplicity and to benefit from IP services, such as security and multicasting. NDP messages consist of an ICMPv6 header, neighbor discovery (ND) message–specific data, and ND message options, which provide additional information, such as link-layer addresses, on-link network prefixes, on-link maximum transmission unit information, redirection data, mobility information, and router specification.

 Copublished by the IEEE Computer and Reliability Societies

NDP Messages and Functionalities

NDP uses the following five ICMPv6 messages:

- router solicitation (RS), type 133—the IPv6 host sends RS to discover the default router and learn network information, such as prefixes and Domain Name Server (DNS) addresses;
- router advertisement (RA), type 134—the router that receives an RS message sends back an RA message (solicited RA) and IPv6 routers send RAs periodically (unsolicited multicast RAs);
- neighbor solicitation (NS), type 135—NS resolves the neighbor node's IPv6 address to its MAC address and verifies that the node is still reachable;
- neighbor advertisement (NA), type 136—the node that receives an NS message sends back an NA message with its own MAC address; and
- redirect message (RM), type 137—the router uses RM to inform other nodes of a better first-hop toward a destination.

These messages achieve various functionalities, including router and prefix discovery, parameter discovery, address autoconfiguration, address resolution, duplicate address detection (DAD), neighbor unreachability detection, next-hop determination, and redirect.

Abbreviations

ADD	Authorization delegation discovery
ARP	Address Resolution Protocol
CGA	Cryptographically generated address
CPA	Certificate path advertisement
CPS	Certificate path solicitation
DAD	Duplicate address detection
DHCPv6	Dynamic Host Configuration Protocol for IPv6
DoS	Denial of service
ECC	Elliptic curve cryptography
ICMPv6	Internet Control Message Protocol for IPv6
IID	Interface identifier
IPsec	IP Security
MITM	Man in the middle
NA	Neighbor advertisement
NDP	Neighbor Discovery Protocol
NS	Neighbor solicitation
RA	Router advertisement
RM	Redirect message
RS	Router solicitation
SEND	Secure Neighbor Discovery
SLAAC	Stateless address autoconfiguration
WinSEND	Windows Secure Neighbor Discovery

NDP Security and Privacy Implications

NDP has some basic protection mechanisms based on its scope. It's a link-local protocol, so the source address must be either unspecified or a link-local address, and the hop limit must be set to 255. Also, the routers don't forward link-local addresses. Thus, NDP messages can't be injected into the network infrastructure from beyond directly connected layer-2 access networks. This shield isn't enough to completely protect IPv6 local networks. Without securing NDP, IPv6 neighbor discovery is vulnerable to spoofing, denial-of-service (DoS), replay, redirect, and rogue router attacks.

Spoofing. In a spoofing attack, a malicious node successfully uses another node's address or identifier. Attackers can use spoofs to leverage man-in-the-middle (MITM) attacks, create DoS attacks, hide their identities, abuse the trust relation between legitimate nodes, and so forth. Address Resolution Protocol (ARP) spoofing is a well-known attack in IPv4 networks. Instead of ARP, IPv6 relies on NDP to carry out address resolution in addition to the other services. Without the IPv6 authentication mechanism, attackers can generate crafted IPv6 packets with spoofed source addresses and send them across the network.

Denial of service. DoS attacks prevent communication between the legitimate node and other nodes, using significant system resources. For instance, attackers can generate DoS on DAD to prevent a network node from obtaining a network address. DAD ensures that no address collision exists on the same link. A malicious node might hinder the legitimate host from getting a new IPv6 address by responding to every duplicate address detection attempt with a spoofed message saying "I have this address." Thus, victims would find out that every IPv6 address they try to use is in use by other nodes on the link and won't be able to configure an IP address to access the network.

Replay. In replay attacks, attackers capture and change messages exchanged between two nodes. NDP messages are prone to replay attacks. For instance, when host 1 wants to communicate with host 2, it sends an NS message to retrieve host 2's MAC address. Attackers on the same link can capture the NS message and change it to take over the traffic flow between host 1 and host 2. Another example is RA message replay. Attackers receive the RA, change its parameters, and resend this false router information on the link.

Redirect. Redirects are a class of attacks in which a malicious node redirects packets away from the legitimate

receiver to another node. In IPv6, routers use RMs to inform nodes of better first-hop routers. Attackers can fabricate such a message and take over routing from the legitimate router, acting as an MITM and intercepting all messages between the two nodes.

Rogue router. In this type of attack, a malicious node injects rogue information to poison the routing tables, reroute traffic, or prevent victims from accessing the desired network. It's simple for attackers to configure a rogue router on an unsecured link, but it's difficult for a node to distinguish between fake and authorized RAs, especially for a newly connected node, which can't validate the routers without an IP address to communicate. Thus, a malicious node can advertise itself as a router and send a bogus address prefix or advertise itself as a last-hop router to act as an MITM and effectively receive, drop, or replay the packets. Such attacks can cause serious problems because they affect all nodes connected to the same segment.

Combinations. Attackers can use several of these attack types simultaneously. For instance, spoofing and DoS can be used together: attackers can fabricate packets with a fake source IP address to hide their identity, which makes detecting the attack more difficult.

RFC 3756, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," describes and categorizes some possible NDP attacks.[3] In addition, the Hacker's Choice IPv6 (http://thc.org/thc-ipv6) already implemented a toolset to attack IPv6.

NDP Privacy Implications

SLAAC could lead to serious privacy issues. Generating the IPv6 interface identifier (IID) on the basis of the MAC address results in a static IID, which remains constant over time and across networks. Consequently, attackers can correlate the captured traffic from a specific IID to a certain device and track the node. Once attackers determine the user's location and identity, they can target the user for identity theft or related crimes.

RFC 4941, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," introduces a solution by generating global scope addresses from IIDs that change over time.[7] Changing IIDs over time makes it more difficult for eavesdroppers and other information collectors. However, RFC 4941 doesn't protect against IP address spoofing attacks.

CGAs can obscure the node IID to protect privacy and achieve anonymity.

IPsec for Securing NDP

Although the original NDP specification called for the use of IP Security (IPsec) to protect NDP messages, it didn't specify how to use it. IPsec isn't suitable for securing the NDP autoconfiguration process owing to a bootstrapping problem. With the Internet Key Exchange, the nodes must be addressable before IPsec can be used. With IPsec, Internet Key Exchange can automatically perform security associations exchange when a host already has a valid IPv6 address. Therefore, the Internet Engineering Task Force developed SEND as a countermeasure to NDP vulnerabilities.

Secure Neighbor Discovery

RFC 3971 is a set of NDP enhancements.[4] SEND offers three additional features to NDP: address ownership proof, message protection, and a router authorization mechanism. To achieve these enhancements, SEND comes with four new options (CGA, RSA signature, nonce, and Timestamp) and two ICMPv6 messages for identifying the router authorization process.

Cryptographically generated address. The CGA option carries the associated CGA parameters so the receiver can validate the proper binding between the public key (used to verify the signature) and the CGA.

The CGA is an essential part of SEND, proposed to prevent address stealing. It authenticates IPv6 addresses without requiring third-party or additional security infrastructure. CGAs are IPv6 addresses, in which a one-way hashing of the node's public key and other auxiliary parameters generates an IID. Thus, a node's IPv6 address is bound to its public key. The receiver can verify this binding by recomputing the hash value and comparing it to the sender's IPv6 address's IID.

Figure 1 shows the CGA-generation algorithm (see RFC 3972 for more details[5]). CGA generation begins by determining the address owner's public key and selecting the proper security-level (Sec) value, then continues the hash2 computation loop until finding the final modifier. The hash2 value is an SHA-1 hash value over the entire CGA Parameters data structure (the public key and Collision Counts are zeros). The address generator tries different modifier values until 16 × Sec-leftmost-bits of hash2 equals zero. Once it finds a match, the loop for hash2 computation terminates. The final modifier value is saved and used as an input for hash1 computation. The hash1 value is a hash of combination of the whole CGA parameter data structure. Then, IID is derived from hash1. The Sec value is encoded into the IID's three leftmost bits. The seventh and eighth bits from the left of IID are *u* and *g* bits. The *u* bit (universal/local bit) is set to 1 to indicate universal scope or to 0 to indicate local scope. The *g* bit is the individual/group bit (see RFC 4291 for more details[8]). Finally, DAD ensures that no address collisions are in the same subnet.

A CGA's main disadvantage is its computational cost.

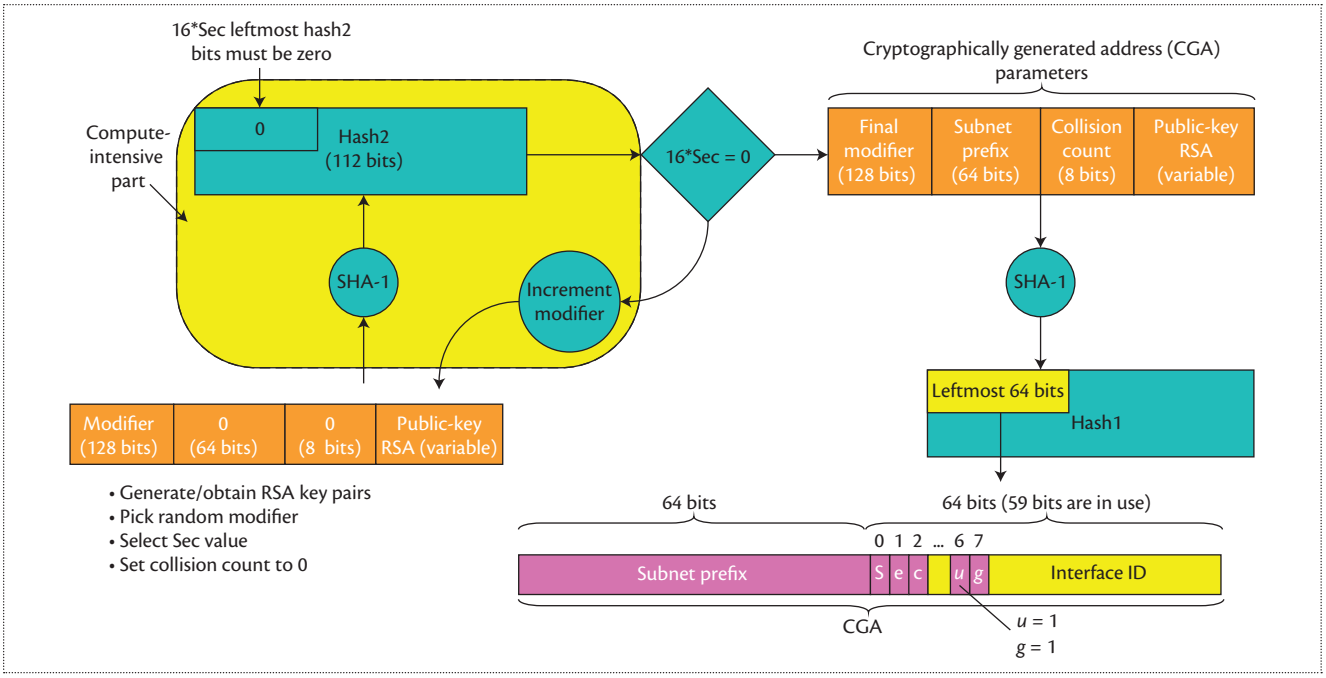

Figure 1. A cryptographically generated address's generation algorithm. The address owner (generator) generates or obtains an RSA key pair and selects the proper security-level (Sec) value, then continues the hash2 computation loop until finding the final modifier, which satisfies the condition that 16 × Sec-leftmost-bits of hash2 equals zero. The hash1 value is a hash of combination of the whole CGA parameter data structure. IID is derived from hash1, and the Sec value is encoded into the IID's three leftmost bits.

CGA computations can take a long time, especially for high Sec values. In fact, satisfying the hash2 condition is the most computationally expensive part of the CGA's generation algorithm. Multiple hashes are computed over the CGA parameter data structure by incrementing the modifier each time (see Figure 1).

RSA signature. SEND uses the RSA signature option to authenticate the sender's identity. Initially, each node must generate or obtain a public/private RSA pair before it can claim an address. The sender signs the outgoing messages with the private key, which corresponds to the public key used in CGA's generation algorithm. This signature prevents attackers from spoofing CGA addresses.

Nonce. The nonce option uses a random number to ensure that an advertisement is a fresh response to a node's solicitation. SEND includes a nonce option in the solicitation message and requires advertisements to include a matching option. This prevents a replay attack in solicited messages, such as NS/NA and RS/RA, which can be used for two-way communication but not for one-way communication messages.

Timestamp. SEND uses the Timestamp option to ensure replay protection against unsolicited advertisements, such as periodic RAs and RMs. Here, the assumption is that all nodes have synchronized clocks, so the node can prevent replay attacks by carrying out a time-stamp-checking algorithm.

Router authorization. SEND uses authorization delegation discovery (ADD) to validate and authorize IPv6 routers to act as default gateways and specifies IPv6 prefixes that a router is authorized to announce on its link. ADD relies on an electronic certificate issued by a trusted third party. Before any node can accept a router as its default, the node must be configured with a trust anchor that can certify the router via certificate paths. So, the node requests that the router provide its X.509 certificate path to a trust anchor, which is preconfigured on the node. The router shouldn't be trusted if it fails to provide the path to the trust anchor. Figure 2 shows a simplified view of the router authorization mechanism.[9]

SEND offers two new ICMPv6 messages to identify the router authorization process: certificate path solicitation (CPS) and certificate path advertisement (CPA). A host sends a CPS message, ICMPv6 type 148, during the ADD process to request a certification path between a router and one of the host's trust anchors. The CPA message, ICMPv6 type 149, is sent in reply to the CPS message and contains the router certificate.

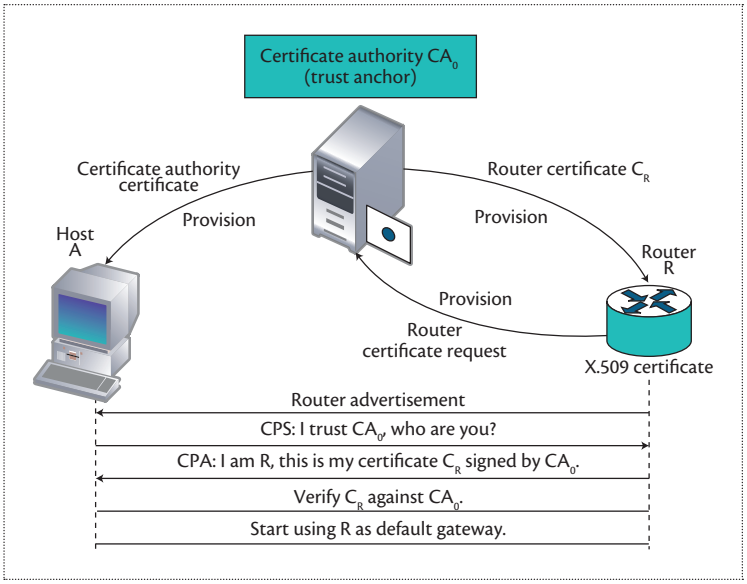

Figure 2. Router authorization process.[9] The host sends a certificate path solicitation (CPS) to ask the router to provide a valid certificate. The router responds with the certificate path advertisement (CPA) that contains the certificate. The host verifies router legitimacy; if it's valid, the host accepts the router as the default router.

SEND Deployment Challenges

SEND faces limitations in several areas, including computation, implementation, deployment, and security, which might prevent CGA and SEND use and leave NDP messages vulnerable to potential attacks.

CGA and SEND Security Constraints

The CGA mechanism can prevent theft of another node's address, because attackers must find a cryptographic hash value (hash1) collision. However, CGA can't provide assurance about the real node's identity, and it isn't sufficient to guarantee that the CGA address is used by the appropriate node. Because CGAs aren't certified, attackers can create new valid addresses from their own public keys and start the communication. For more security, a certificate authority is necessary to validate the keys. However, address owners use the corresponding private keys to assert their ownership and sign NDP messages sent from the address. Therefore, attackers can impersonate other node addresses from a valid public key but can't sign the others' messages.

CGA verification vulnerabilities. Attackers can perform DoS attacks against CGA-DAD messages using CGAs or non-CGAs. Attackers can use non-CGAs to respond to each CGA-DAD message, with "I have this address." If the collision count reaches 2, CGA generation fails. Therefore, the CGA algorithm might need to be extended to verify the DAD message responses before the increment of the collision count. But in this case, all nodes should support CGA or discard the DAD messages that come from non-CGAs as well as give priority to new CGA addresses rather than non-CGA addresses.

Attackers can capture ND messages and change the sender's CGA parameters, so the CGA verification process at the receiver side will fail. In this scenario, attackers prevent communication between sender and receiver; however, this can happen with other security protocols. For example, if attackers insert a bogus IPsec datagram, the IPsec at the receiver drops the datagram because the integrity check is bogus. Therefore, we don't see this attack as a major drawback of CGAs.

Time-memory trade-off attacks. CGAs are vulnerable to global time-memory trade-off attacks. Attackers can generate a table of valid public/private key pairs in the precomputation phase to reduce the time complexity of the attack on the cost of memory. Researchers proposed a more secure version, called CGA++, to resist this type of attack.[10] In CGA++, the subnet prefix is included in the hash2 calculation, and all the modifier, collision count, and subnet prefix values are signed by the private key corresponding to the public key used. This way, time-memory trade-off attacks can't be applied globally. Attackers could perform a brute-force search for each address prefix separately. However, it's not easy to impersonate a random node in a network owing to the ample storage necessary to carry out this attack. In addition, CGA++ comes with an added cost due to the signature verifications, and it requires much more time than CGA generation for the same Sec value.

Attacks on router authorization. DoS attacks can target router authorization mechanisms. This requires performing many operations for generating, verifying, and signing NDP messages, which can affect the routers' performance. Attackers might target hosts by sending unnecessary certification paths, forcing hosts to spend useless memory and verification resources on them. The host's numerous computations to generate and verify CGAs and the certificates chain make it vulnerable to DoS attacks.

Additional limitations. SEND doesn't guarantee confidentiality or provide protection against address scanning. It doesn't offer information about the node's privileges on the network; if this functionality is necessary, another infrastructure, such as 802.1X, is required.

CGA Privacy Constraints

Owing to CGA's high computation complexity, it's

Table 1. CGA generation time for different security-level values.

Device specifications	Sec = 0	Sec = 1	Sec = 2	Sec = 3
Modern PC (AMD64)[10]	n/a	0.2 s	3.2 hours	n/a (theoretically, 24 years)
Machine with 2.67-GHz CPU[11]	93.41 ms	401.99 ms	1.65 hours	n/a (theoretically, 12 years)
Duo2 (2.53-GHz) workstation[12]	100 μs	60 ms	2,000 s	n/a (theoretically, more than 30,000 hours)

likely that once a node generates an acceptable CGA, it will continue to use it at that subnet. Consequently, nodes using CGAs are still susceptible to privacy-related attacks. However, this issue can be solved by setting a lifetime for CGA addresses. If the lifetime passes, a new CGA with a new CGA parameter should be generated. To avoid the long CGA generation time, we don't recommend choosing a Sec value greater than 1 for current applications. There should be a balance between lifetime and security level.

CGA can achieve privacy protection for a mobile node. When a node moves to a new subnet, hash1 should be recalculated using the new subnet prefix. Consequently, new IPv6 addresses with new IIDs will be generated without needing to recalculate hash2. The process of generating new addresses costs only one hash calculation. The only concern is the possibility of attackers tracking the node on the basis of its public key if it remains fixed. But tracking the node using its public key isn't easy. Normally, tracking nodes over the Internet is done using an IP address.

Computation Exhaustion and Bandwidth Consumption

The average CGA address-generation time depends on Sec bits. However, it's impossible to tell exactly how much time CGA generation will take when Sec isn't zero; it could vary significantly. Theoretically, the computational complexity of hash2 consumption increases by 2^{16} for each Sec value.

By performing 1,000 tests on an Intel Duo2 2.67-GHz CPU, we found that the average CGA generation time is 402 milliseconds for Sec = 1. For Sec = 2, our test on an unrepresentative set of five samples gave an average CGA generation time of 5,923,857 milliseconds (1 hour and 39 minutes). Accordingly, we don't recommend Sec values higher than 1 because they require a lot of processing power and time using current technologies. If the generation and verification values exceed 0.5 second, DAD would fail because it expects to send and receive two ND messages in less than 1 second, as RFC 4861 indicates.[1]

Table 1 presents a summary of CGA generation times reported in several studies.[10–12] Each study measured CGA on different device specifications. Some of these studies considered the whole CGA generation time, and others considered only hash2 calculation time.

Router authorization and certificate validation can be heavyweight and complicated. Any node trying to verify router authorization must be prepared beforehand with the anchor certificate. In addition, routers are provisioned with certificates that prove their identity. Because this relationship can be a long chain of trust, ADD requires an end node to store and retrieve all the certificates in the certification path to verify the authorized router. Thus, the node requires verification of the lengthy certificates chain, and transporting all the certificates and keys between routers and hosts increases local network traffic. Also, the certification path information transformation requires many CPS/CPA messages, especially if an error occurs on the network and retransmission is required.

SEND requires each node to include the public key and other parameters with the message and to affix its signature with every signaling packet it generates, which means that more than 1 Kbyte is added to each packet. This increases communication overhead and consumes network bandwidth and computational resources.

Lack of Sophisticated Implementation

Most operating systems support NDP but lack support for SEND. Even though some major vendors, such as Cisco and Juniper, have various levels of support for SEND in their routers, no major operating system provides a good level of support. Current SEND implementations for specific OS distribution, such as Debian Linux, are basically proofs of concept rather than production-ready software. Some of these implementations—send-0.2, NDprotector, Easy-SEND, and Windows Secure Neighbor Discovery (WinSEND)—are done in the user space, and others—send-0.3 and ipv6-send-cga—at the kernel level.

DoCoMo's SEND implementation (send-0.2). NTT DoCoMo USA Labs implemented the first open source SEND. Its implementation (send-0.2) works on Free Berkeley Software Distribution (FreeBSD), using the Berkeley Packet Filter interface embedded in a netgraph node for transferring packets between the kernel and the user-space daemon. The user-space daemon handles

the SEND options. The communication between the NDP stack in the kernel and the SEND daemon flows through the chain of netgraph nodes rather than passing through normal layer processing.

The send-0.2 implementation has some limitations. First, all network traffic must traverse through packet-filtering hooks, which introduce significant processing overhead. Second, send-0.2 depends on a netgraph subsystem, which is available for the FreeBSD operating system family, making it importable for other operating systems. Users need to have good knowledge of firewall rules (ip6tables), because it's essential for the application's configuration. Finally, DoCoMo USA Labs is no longer maintaining the SEND project—source code is no longer available for download and support has been canceled.

> **Reducing the threats against the IPv6 network by enhancing SEND security and making it a lightweight and deployable authentication mechanism is important.**

NDprotector. NDprotector (http://amnesiak.org/NDprotector) is another implementation of CGA and SEND for Linux based on Scapy6 and is limited to the Linux platform owing to its dependency on iproute2, ip6table, and netfilter queue. Implementation can be divided into initialization and runtime phases. During initialization, CGA addresses are set up and ip6table rules are defined. These rules route all NDP messages to specific netfilter queues. NDprotector's runtime component takes the NDP messages out of these queues and secures the outgoing NDP messages or verifies the incoming ones. NDprotector supports elliptic curve cryptography (ECC) keys in addition to standard RSA keys. It's implemented in Python and therefore uses a Python wrapper to access netfilter. For packet manipulation, NDprotector uses a modified version of Scapy6 (scapy6send).

Easy-SEND. Easy-SEND is another Linux user-space implementation of SEND developed in Java.[13] Easy-SEND is an open source project developed for educational purposes. Easy-SEND doesn't implement router authorization.

WinSEND. WinSEND, a user-space implementation developed in Microsoft .NET, is the first SEND implementation for Windows.[14] WinSEND works as a service for Windows families with a user interface to set security parameters for the proper network interface card.

The user-space implementations have some advantages, such as avoiding kernel crashing. However, this means emulating the behavior of a kernel structure, which isn't always possible. Therefore, there are some trial implementations to integrate the SEND with NDP code at the kernel.

Native SEND kernel API for BSD (send-0.3). This implementation aims to overcome the major drawbacks of DoCoMo's implantation by implementing a new kernel-user-space API for SEND and eliminating the use of netgraph and the Berkeley Packet Filter.[15] Avoiding netgraph reduces overhead, speeds up packet processing, and makes the implementation more portable to other operating systems. Send-0.3 uses routing control sockets to exchange messages between the kernel and the user space. It handles the packets that might be affected by SEND rather than processing all packets, and lets the existing kernels handle the other packets. This implementation uses a kernel module that acts as a gateway between the network stack and the user-space interface.

Huawei and BUPT (ipv6-send-cga). Huawei and Beijing University of Post and Telecommunications (BUPT) introduced a SEND implementation (www.ohloh.net/p/ipv6-send-cga) in the Linux kernel IPv6 module. The C code is partially built into the real NDP code at the kernel, where direct access to neighboring caches and the routing table is available. However, cryptographic processing remains in the user space. This work is a research prototype under development and lacks interoperability testing. Bugs that could cause the kernel to crash are expected.

Proposals to Facilitate SEND Deployments

Researchers have proposed ways to optimize and facilitate SEND deployment. In "Significantly Improved Performances of the Cryptographically Generated Addresses Thanks to ECC and GPGPU," Tony Cheneau and his colleagues proposed an improved method for generating CGA by using an ECC key instead of a standardized RSA key.[16] For the same security level, ECC has shorter keys, which leads to smaller packet sizes. Accordingly, ECC use is more suitable in resource-limited environments.

Another potential idea is delegating the expensive computation to a powerful machine. In "A Quick CGA Generation Method," a key server performs the computation on behalf of the nodes in advance or offline.[17] The work in "Configuring Cryptographically Generated Addresses (CGA) Using DHCPv6" proposed using the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) server to manage CGA.[18] DHCPv6 is extended to propagate the parameters that a host

needs to generate CGA. A host might send a request to a DHCPv6 server to compute the CGA for it. However, these approaches return to a centralized model in which the server might be the target of DoS attacks.

Other solutions use cryptographic accelerator cards to compute CGA faster or use parallelized algorithms to compute the CGA modifier. "Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating Systems" proposed implementing a parallelized CGA-generation algorithm to speed up CGA computation.[19] With the parallel approach, the speed-up time increased substantially by increasing the number of cores in the computing device.

To avoid unexpected delays in CGA generation, using a time termination condition can be a practical approach.[20] For example, on the basis of the CPU speed, the algorithm recommends a proper value for Sec or a time termination condition. The termination condition also depends on the application requirement; for example, MIPv6 needs to finish the address generation within hundreds of milliseconds. "Stopping Time Condition for Practical IPv6 Cryptographically Generated Addresses" proposed a modified CGA-generation algorithm called *time-based CGA*, which limits the maximum time that the user/application can invest for CGA generation.[11] Time-based CGA takes the upper bound of the CGA running time as an input and determines the Sec value as an output of the brute-force computations. This paper also suggested reducing security-level granularity from 16 to 8 to increase the chances of having a better Sec value within the time limit.

Other approaches tried to avoid SEND's deployment complexity by using alternative protection models. RFC 6105, "IPv6 Router Advertisement Guard," proposed using filters at layer 2 to avoid router authorization and certificate validation complexity.[21] It counters the rogue RA problem by applying a layer-2 switching device to identify and block invalid RAs. However, IPv6 RA-Guard still can't prevent IP address theft if deployed alone. A combination of CGA and RA-Guard might be a solution to secure the IPv6 network.

Other approaches, such as NDPMon (http://ndpmon.sourceforge.net) and RAmond (http://ramond.sourceforge.net) tools, protect the local network by monitoring the NDP messages to detect malicious activity. However, monitoring techniques can't prevent attacks; moreover, an attack might damage a network before the administrator can respond. Also, determining the attack node isn't easy, because attackers can use spoofed addresses.

We offer the following recommendations and comments regarding CGA and SEND deployments:

- Don't use CGA with a Sec value greater than 1, because higher values are impractical for most users and applications with current CPU speeds.
- A parallelizing CGA algorithm can alleviate the required time for CGA generation by speeding up CGA computations and investing the available CPU in CGA computation, especially when the device has multiple cores.
- Generate key pairs on the fly using a CGA-generation program to make it easier for the user and to avoid storing the keys in a particular path before starting the application. This way, the keys won't be vulnerable to theft.
- Because there's no guarantee of stopping CGA generation after a certain time for a Sec value greater than zero, we recommend using time-based CGA to limit the CGA generation time.[11]
- Because of the router authorization mechanism's complexity, it's necessary to find an easy certificate deployment model for SEND. We recommend using a CGA compound with complementary approaches—for example, using CGA with another detection or monitoring mechanism, such as NDPMon or RAmond. Using CGA with RA-Guard prevents address theft and detects fake RAs—CGA protects the local link from address theft, and the RA-Guard protects the local link from fake RAs.
- For a very secure IPv6 local network, we recommend that all nodes use SEND. Therefore, we ask the operating systems' vendors to implement SEND.

Reducing and eliminating the threats and attacks against the IPv6 network by enhancing SEND security and making it a simple, lightweight, and deployable authentication mechanism is important. Without these security measures, IPv6 network will be left vulnerable to IP spoofing-related attacks. ■

References

1. T. Narten et al., "Neighbor Discovery for IP Version 6 (IPv6)," RFC 4861, Sept. 2007; http://tools.ietf.org/html/rfc4861.
2. S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862, Sept. 2007; http://tools.ietf.org/html/rfc4862.
3. P. Nikander, J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," RFC 3756, May 2006; http://tools.ietf.org/html/rfc3756.
4. J. Arkko et al., "SEcure Neighbor Discovery (SEND)," RFC 3971, Mar. 2005; http://tools.ietf.org/html/rfc3971.
5. T. Aura, "Cryptographically Generated Addresses (CGA)," RFC 3972, Mar. 2005; http://tools.ietf.org/html/rfc3972.
6. A. Conta, S. Deering, and M. Gupta, "Internet Control

Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 4443, Mar. 2006; http://tools.ietf.org/html/rfc4443.
7. T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC 4941, Sept. 2007; http://tools.ietf.org/html/rfc4941.
8. R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291, Feb. 2006; http://tools.ietf.org/html/rfc4291.
9. "IPv6 Secure Neighbor Discovery: Protecting Your IPv6 Layer 2 Access Network," Cisco Systems, 2009; www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/whitepaper_c11-602135.html.
10. J.W. Bos, O. Özen, and J.-P. Hubaux, "Analysis and Optimization of Cryptographically Generated Addresses," LNCS 5735, Springer, 2009, pp. 17–32.
11. A. AlSa'deh, H. Rafiee, and C. Meinel, "Stopping Time Condition for Practical IPv6 Cryptographically Generated Addresses," *Proc. 26th IEEE Int'l Conf. Information Networking* (ICOIN 12), IEEE, 2012, pp. 257–262.
12. S. Jiang, "Analysis of Possible DHCPv6 and CGA Interactions," draft, 12 Mar. 2012; http://tools.ietf.org/html/draft-ietf-csi-dhcpv6-cga-ps-09.
13. S. Chiu and E. Gamess, "A Free and Didactic Implementation of the SEND Protocol for IPv6," *Machine Learning and Systems Engineering*, vol. 68, S.-I. Ao, B. Rieger, and M.A. Amouzegar, eds. Springer, 2010, pp. 451–463.
14. H. Rafiee, A. AlSa'deh, and C. Meinel, "WinSEND: Windows Secure Neighbor Discovery," *Proc. 4th Int'l Conf. Security of Information and Networks* (SIN 11), ACM, 2011, pp. 243–246.
15. A. Kukek and B.A. Zeeb, "Native Send Kernel API for BSD," 2010; http://people.freebsd.org/~anchie/SeND_AsiaBSDCon_2010.pdf.
16. T. Cheneau, A. Boudguiga, and M. Laurent, "Significantly Improved Performances of the Cryptographically Generated Addresses Thanks to ECC and GPGPU," *Computers & Security*, vol. 29, no. 4, 2010, pp. 419–431.
17. S. Guangxue et al., "A Quick CGA Generation Method," *Proc. 2nd Int'l Conf. Future Computer and Communication* (ICFCC), IEEE, 2010, pp. V1-769–V1-773.
18. S. Jiang and S. Xia, "Configuring Cryptographically Generated Addresses (CGA) Using DHCPv6," 11 Apr. 2012; http://tools.ietf.org/html/draft-ietf-dhc-cga-config-dhcpv6-02.
19. H. Rafiee, A. AlSa'deh, and C. Meinel, "Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating Systems," *Proc. 26th IEEE Int'l Conf. Information Networking* (ICOIN 12), IEEE, 2012, pp. 269–274.
20. T. Aura and M. Roe, "Strengthening Short Hash Values," 2009; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.7681.
21. E. Levy et al., "IPv6 Router Advertisement Guard," RFC 6105, Feb. 2011; http://tools.ietf.org/html/rfc6105.

Ahmad AlSa'deh is a PhD student at the Hasso-Plattner-Institut at the University of Potsdam, Germany. His research interests include networking security, particularly IPv6 security. AlSa'deh has an MS in scientific computing from Birzeit University in Palestine. Contact him at ahmad.alsadeh@hpi.uni-potsdam.de.

Christoph Meinel is a professor and director of the Hasso-Plattner-Institut at the University of Potsdam, where he leads the Internet Technologies and Systems research group. His research interests include security and trust engineering, Web 3.0, and eLearning. Meinel has a PhD computer science from the Humboldt-University in Berlin. Contact him at christoph.meinel@hpi.uni-potsdam.de.