A secure framework for protecting IPv6 Neighbor Discovery Protocol

Colin Putman

Technical Report

RHUL–ISG–2019–3

27 March 2019

Information Security Group
Royal Holloway University of London
Egham, Surrey, TW20 0EX
United Kingdom

Student Number: 100885782
Colin Putman

Title: A Secure Framework for Protecting IPv6 Neighbor Discovery Protocol.

Supervisor: Chris Mitchell

Submitted as part of the requirements for the award of the
MSc in Information Security
at Royal Holloway, University of London.

I declare that this assignment is all my own work and that I have acknowledged all quotations from published or unpublished work of other people. I also declare that I have read the statements on plagiarism in Section 1 of the Regulations Governing Examination and Assessment Offences, and in accordance with these regulations I submit this project report as my own work.

Signature:

Date:

Table of Contents

List of abbreviations

6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ARP	Address Resolution Protocol
CGA	Cryptographically-Generated Address
CPA	Certificate Path Advertisement
CPS	Certificate Path Solicitation
CN	Controller Node
DAD	Duplicate Address Detection
DHCP; DHCPv6	Dynamic Host Configuration Protocol; DHCP version 6
DNS	Domain Name System
DNSSEC	DNS security extensions
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
EUI-64	Extended Unique Identifier 64-bit
FFS	Feige-Fiat-Shamir cryptosystem
ICMP	Internet Message Control Protocol
IETF	Internet Engineering Task Force
IDS	Intrusion Detection System
IKE	Internet Key Exchange
IoT	Internet of Things
IP; IPv4; IPv6	Internet Protocol; IP version 4; IP version 6
IPsec	Internet Protocol Security
MCGA	Multi-key Cryptographically-Generated Address
MTU	Maximum Transmission Unit
NA	Neighbor Advertisement
NDP	Neighbor Discovery Protocol
NIST	National Institute of Standards and Technology
NS	Neighbor Solicitation
NUD	Neighbor Unreachability Detection
OCSP	Online Certificate Status Protocol
RA	Router Advertisement
RAP	Router Authorisation Passport
RPK	Request Public Key
RPKI	Resource Public Key Infrastructure
RS	Router Solicitation
RSA	Rivest-Shamir-Adleman cryptosystem
SAVI	Source Address Validation Improvement
SEND	Secure Neighbor Discovery
SHA-1, -2, -3	Secure Hash Algorithm 1, 2, 3
SHA-256	SHA-2 256-bit output
SLAAC	Stateless Address Auto-Configuration
SPK	Send Public Key
SSAS	Simple Secure Addressing Scheme
SSL/TLS	Secure Socket Layer/Transport Layer Security
TAO	Trust Advertisement Option
TBS	Trust Based Security
TSO	Trust Solicitation Option

Executive Summary

One of the major problems to overcome in the transition from IPv4 to IPv6 is the security of the Neighbor Discovery Protocol (NDP) in IPv6, which is used to find neighbouring devices and routers on the same network link and to resolve local IPv6 addresses into link-layer addresses. Because NDP must be used as soon as a network is joined, before an IP address is chosen, it cannot use Internet Key Exchange (IKE) and so cannot be secured using IPsec as the rest of IPv6 can. It is therefore necessary for another system to be developed to secure this protocol, and while numerous proposals have been made, none have yet been found to be completely satisfactory for this purpose.

This project builds upon previous work surveying the proposals in this problem area by closely analysing a selection of the most successful or interesting proposals, focusing on those which directly extend NDP to be less vulnerable to attacks over those which specify reactive solutions based on intrusion detection systems. Particular attention is paid to SEND, currently the most complete and only industry-implemented security extension for NDP, and to the various proposals made to improve upon SEND and mitigate the drawbacks to its design. Two new, brief proposals are also made to add to the current proposals for secure router authentication, these being the use of OCSP queries to confirm that a certificate has not been revoked due to key compromise, and the use of DNSSEC reverse-lookup queries as an alternative trust anchor which does not use certificates.

In addition to providing an analytical survey of the proposals in this area, this project concludes by drawing on selected parts of the examined proposals to formulate a suggestion for a framework of improvements to SEND which aims to cover all of the major drawbacks of the protocol which previous papers have identified. The intention of this framework is to show how aspects of the proposed systems complement each other and can be combined into a more complete and efficient solution than they represent on their own. However, further scrutiny and standards action would be required, likely resulting in changes to the framework, before this solution could be adopted and implemented as an extension to NDP.

1. Introduction

It has been many years now since the Internet Protocol version 6 was first developed to replace IPv4 as the backbone protocol suite of the Internet. Its deployment has been a very slow process, but it is becoming increasingly important to migrate due to the exhaustion of IPv4's address space. It is therefore vital to ensure that the core protocols of IPv6 are secure and practical. Security concerns were a driving force behind the specification and adoption of IPv6 [1, 2, 3], and IPv6 does offer significant security benefits over IPv4. In particular, IPv6 makes support for IPsec mandatory, although it does not require IPsec to be used [1]. IPv6's larger subnet address space also makes some techniques for gathering information on a network impractical. However, some security issues still remain to be solved.

This paper is concerned specifically with the security of the Neighbor Discovery Protocol (NDP) of IPv6. NDP is part of the Internet Control Message Protocol suite for IPv6, and its use in IPv6 is broadly analogous to the Address Resolution Protocol (ARP) in IPv4 with the addition of the router discovery and redirect functions of ICMP [4]. As such, it is susceptible to the same basic threats as ARP [5], and as it is the first protocol a device will need to use when connecting to a network and is needed continually to maintain that connection, it is an extremely important protocol to secure.

However, IPsec is not a satisfactory solution to secure NDP; IKE requires the host device to have an IP address, and one of the purposes of NDP is to acquire an address, leading to a bootstrapping problem, and in many use cases NDP would require too many security associations for practical manual configuration [1, 6, 7]. In the absence of IPsec, NDP assumes that other nodes on the same network link are trustworthy [3, 4, 8], making it vulnerable in some settings such as public wireless networks where this assumption does not hold, as well is in circumstances where another node on the link is compromised by an attacker or an attacker node infiltrates a network. In 2005, the IETF specified a security extension to NDP called Secure Neighbor Discovery (SEND) [6], but in its current state SEND has substantial drawbacks and has not been universally accepted.

Several solutions have been proposed to protect NDP, either by improving SEND or by providing an alternative. These proposals have varying strengths and weaknesses, but none of them are sufficient to defend NDP alone. The aim of this paper is to analyse these proposals and build on them to suggest a framework which achieves as complete a protection of NDP as is feasible, drawing on complementary aspects of different schemes.

The paper begins with a more detailed overview of NDP, its features and its vulnerabilities in section 2. This is followed by an overview of SEND and its drawbacks in section 3, and an analysis of a selection of other proposals both for alternatives to SEND and for upgrades to SEND in sections 3 and 4. In section 5, the most useful of these proposals are organised into a framework which aims to cover all of the major drawbacks of the current security solutions, and the decisions and thought processes going into the design of this framework are explained. The paper concludes by acknowledging some open questions and discussing the next steps in the process of securing NDP.

2. Overview of NDP

The Neighbor Discovery Protocol is a fundamental protocol in IPv6, used to establish contact with other devices on a network, including the network's routers, and to maintain the network's IP-level communication. This chapter summarises the messages and main functions of the protocol, before considering the threats and vulnerabilities in the protocol, as well as some of the challenges that must be faced when trying to secure NDP.

2.1. NDP Messages

NDP specifies five types of message to perform its functions. Each message consists of the IPv6 header followed by an NDP message header which identifies the message type, and a variable options field which is used to contain any necessary additional information depending on the message purpose. The five message types are Neighbor Solicitation, Neighbor Advertisement, Router Solicitation, Router Advertisement, and Redirect.

Neighbor Solicitation (NS) messages are used for several purposes in NDP. An NS message is sent by nodes on a network to request a response from the recipient(s) of the message, but whether it is sent unicast or multicast, and what is sent in the options field, depends on the purpose for which the NS message is being sent. The primary purpose of this message is to carry out address resolution, by sending a multicast NS message onto the network with a set destination IPv6 address and the link-layer address of the sender, awaiting a response from the owner of the destination address which will contain the recipient's link-layer address.

Neighbor Advertisement (NA) messages are required to be sent unicast in response to NS messages, containing the link-layer address of the NA sender, to advertise the sender's presence and role on the network. When this is done for the purpose of address resolution, the link-layer addresses included in the NS and NA messages allow the participants to update their routing tables. Multicast NA messages can also be sent unsolicited to inform other nodes of changes in the sender's link-layer address or role on the network.

Router Solicitation (RS) messages are sent by nodes in a network which are seeking IPv6 routers on the same network link, whether because they have just joined a new network or because they have lost contact with their previous default gateway. This is sent as a multicast message, to ensure that any routers on the link will receive the request, and upon receiving this message routers are required to respond with a Router Advertisement.

Router Advertisement (RA) messages are sent only by routers. A unicast RA message must be sent as a response to any node which sends a RS message, and routers can also periodically send multicast RA messages to advertise their presence to the rest of the network. The options field of the message is used to contain any necessary information about the router and about the network, such as the maximum transmission unit (MTU), the IPv6 address prefix(es) of the network, and a specified minimum amount of time before the sending router can be considered unreachable by receiving nodes.

Redirect messages are sent only by routers, and only processed by host devices. This message is used when a router receives any packet from a host device and its routing table suggests that there is a better first-hop router to use for the packet's destination. The Redirect message includes the link-layer address of the suggested first-hop router, allowing the receiving host device to update its neighbour cache.

These five messages allow NDP to perform all of its intended functions, establishing an interface between IPv6 and the link-layer protocol of the network. The next section describes the functions of NDP, so that subsequent sections can explore how these messages can be abused by an attacker to subvert those functions.

2.2. Functions of NDP

Ahmed et al. [4] identify five main functions provided by NDP, those being address resolution, neighbour unreachability detection, router discovery, duplicate address detection and redirect. Duplicate address detection is part of a feature called stateless address auto-configuration, which I will also summarise here as is a new feature included in NDP which did not exist in IPv4.

Address resolution is a core function of NDP, analogous to the main function of ARP in IPv4, which allows a node to learn the link-layer address corresponding to an IP address on the same network. As described in the previous subsection, this is done by sending a multicast NS message to the IP address in question, containing the sender's link-layer address in the options field. The owner of the target IP address responds with a unicast NA message containing its own link-layer address, and both nodes update their neighbour caches.

Neighbour unreachability detection (NUD) is a simple "heartbeat" function, corresponding to another function of ARP in IPv4, which is sent periodically to each neighbour to confirm whether that neighbour is still connected to the network and able to communicate. This also uses an exchange of NS and NA messages, but in this case the target link-layer address is already known, so the NS message is sent as a unicast message. If no NA message is received after multiple attempts, the target is considered unreachable and the neighbour cache is updated.

Router discovery is the purpose of the RS and RA messages, and as described in the previous subsection, it is achieved using a multicast RS message, to which any receiving routers will respond with a unicast RA message containing network information. This function is used by new nodes joining the network and can also be used to find alternative routers if the sender's default gateway becomes inaccessible.

Stateless address auto-configuration (SLAAC) is a new feature in IPv6 which allows networks to have a more decentralised structure by having each device select its own IPv6 address using the local link's subnet prefix instead of being provisioned one by a central controlling node using DHCPv6. RA messages include a flag in the options field to tell joining nodes whether or not to use SLAAC.

To ensure that auto-configuration does not result in two devices claiming the same IP address, nodes joining the network carry out **duplicate address detection (DAD)**, which consists of the joining node sending a multicast NS message as if performing address resolution, using its own chosen IPv6 address as the target address. Only if no NA response is received will the node initialise the IP address; otherwise it will select a new address and try again.

Redirect is the function using the Redirect message, which is used by a router when it receives traffic from a host on the network link for which the destination is another neighbour on the link or is closer to another router on the link. A Redirect message is sent to the originating host, with the suggested destination link-layer address included in the options field, and the host updates its routing information accordingly.

2.3. Vulnerabilities in NDP

NDP by default includes very few security measures and does not specify any mechanism for verifying that its messages are being used correctly, allowing them to be manipulated by attackers seeking to subvert the network. Most dangerous is the fact that the protocol provides no method for verifying a message's source address, allowing a range of attacks based on IP address spoofing and message forgery [4].

An attacker can forge messages using the IP address of any node on the network, and by forging NS or NA messages in this way it is possible to poison the neighbour caches of any or all nodes on the network, updating their routing information for the impersonated node to a bogus link-layer address for a blackhole attack or to the attacker's link-layer address for a redirect attack. In the case of a forged multicast NS or NA message, the impersonated node would also receive the message, and so the attack could be detected by an intrusion detection system (IDS) running on each node, but this is not part of the NDP specification. A forged unicast NA would be harder to detect but would have to be sent opportunistically in response to a multicast NS message from a target which had not previously established contact with the impersonated node.

Another use for forged NA messages is to cause failure of the NUD function. If an attacker has taken a node offline as part of an attack, that attacker can forge NA responses to other nodes' unicast NS messages to prevent other nodes on the network from becoming aware that the target has been taken offline [6].

It is also possible to spoof messages from non-existent nodes, and to make attacks based on doing so. In particular, networks which make use of SLAAC are vulnerable to a denial of service attack which prevents new nodes from joining the network. A new node attempting to join will generate its own IPv6 address and will send out a NS message to carry out DAD; an attacker can set up a node to listen for these DAD attempts and respond to each one with a forged NA message claiming to own the chosen address. The joining node will simply continue to retry with new addresses and will be unable to enter the network as long as the attacker remains.

Attackers are also able to impersonate routers, since NDP does not include any mechanism for verifying that the sender of a router message is authorised to send that message. This is a more dangerous vulnerability, as routers are part of the infrastructure of the network and affecting communication with a gateway router will affect communication with the Internet outside the network. The simplest way of attacking this communication is to forge a Redirect message from a target's gateway router to cause that target to send some of its traffic to an arbitrary address, which may be the attacker's address. Alternatively, forging RA messages from the default router with the router lifetime option set to 0 can cause listening nodes to treat the router as unreachable [4].

An attacker can also send RA messages, which are not authenticated in the NDP specification, to manipulate other nodes into treating the attacker's device as a gateway router. This can allow the attacker to make sinkhole and man-in-the-middle attacks [1, 5]. The attacker can also include false network information in the RA message to disrupt communication, for example by falsifying the MTU to cause oversized datagrams to be sent. Alternatively, additional subnet prefixes can be added to the RA message, potentially causing nodes attempting to join the network to select invalid addresses, and nodes already on the network to interpret off-link addresses as neighbours [4] which the attacker may be able to impersonate by responding to the resulting NS messages.

There is a clear common theme of impersonation in these vulnerabilities, and they can largely be attributed to two main problems. The first is that an attacker can spoof messages from any IP address within the network and the spoofed messages will be accepted as genuine; this is most problematic when a message is spoofed from an IP address belonging to another device on the network, and is responsible for the majority of the attacks on host nodes described above, but even spoofing unclaimed IP addresses can be dangerous as in the case of the DAD denial of service attack.

The second and more dangerous problem is that router messages (RA and Redirect) are not authenticated at all, and can be sent from any node on the network, allowing an attacker to influence network traffic by impersonating a router and spreading misinformation about the network topology and settings.

2.4. Challenges with securing NDP

IP spoofing and cache poisoning are far from being new problems, having been known attacks against IPv4 and ARP for decades. However, there are several challenges which any attempt to secure NDP will have to face, some of which make the traditional mitigations for these vulnerabilities unwieldy.

One of these is the far greater maximum size of IPv6 networks; each subnet contains 2^{64} unique possible addresses, and a network may use more than one subnet prefix. This means that larger IPv6 networks will be more difficult to monitor and to protect. Any security solution for NDP will need to scale well with the size of the network. This means, for example, that using static (read-only) neighbour cache entries to protect against cache

poisoning may be an impractical solution, as the number of cache entries to protect increases geometrically with the network size.

IPv6 networks can also come in a very wide variety of structures and topologies, and unlike IPv4 networks they can theoretically be completely decentralised, as SLAAC eliminates the need for a DHCP server. A downside of this is that it removes a potential source of verified information about the network, which could otherwise be used to detect and recover from cache poisoning and other misinformation spread by attackers on the network. However, it also lessens the network's reliance on a specific node which would otherwise cause problems in the case of a failure and would make an attractive target for attackers.

Another challenge with securing NDP, and one of the most difficult, comes from the extreme heterogeneity of the Internet. The devices which will be using NDP are highly varied in their capabilities and resources, from large servers and databases to embedded computers and IoT devices. In many cases the participants in a network will all be of similar types, but there will be many other networks in which a variety of devices are interacting together, meaning that they must all understand and be able to use the same security mechanisms. This results in a difficult situation in which the security mechanisms must be practical for resource-constrained devices to use, but as far as possible still difficult for much more powerful devices to break or circumvent.

These difficulties are further compounded by the fact that host devices need to be able to run this protocol successfully and trust the results they get before they are able to access the Internet securely. This presents an obstacle if any mechanism relies on public information, as the host device's default gateway router may be a malicious node impersonating a router, and therefore unsecured information from the Internet may not be trustworthy. In extreme cases, the host device might not be aware of the (real) time, if it is connecting to the Internet for the first time after being switched off or without power for a long period of time.

Finally, another complication arises as a result of the fact that NDP has already been implemented and is in use by some devices, some of which may be difficult or impractical to update with changes to the protocol. In order to make sure secured devices are still compatible with unsecured devices, it is important to make sure that any security measures do not alter the underlying protocol, but instead run in addition to it. Devices running the security measures must also be able to process NDP messages from unsecured nodes, despite the lower level of assurance the unsecured messages would provide and without compromising their own security to do so.

These challenges are extensive and difficult to overcome, but several proposals have already been made to do so. The next chapter explores the current leading proposal, SEND.

3. Secure Neighbor Discovery

Having discussed the vulnerabilities in NDP and the many challenges inherent in trying to secure it, this paper will now begin to explore some of the numerous solutions proposed over the years to improve the security of the protocol. The most notable of these is Secure Neighbor Discovery (SEND). SEND is a security extension to NDP which was specified by the IETF in RFC 3971 [6] in 2005. Its adoption has been slow, and even now only a few implementations have been deployed [3], but it is nonetheless currently the only industry-implemented standard for securing NDP [4].

This chapter SEND in some detail, beginning with a summary of its features before looking at some of the standard's shortcomings. The chapter then examines some alternative proposals to SEND, comparing them with SEND to determine which would be the best to use as a basis for the framework proposed in chapter 5.

3.1. Features of SEND

There are four main features provided by SEND to mitigate the vulnerabilities described in section 2.3: cryptographically-generated addresses, digital signatures, certificate paths, and replay prevention. For the most part, these functions are delivered through specified additional data to be included in the options field of existing NDP messages, along with new rules for processing messages. Two new messages are also specified, however, to be used in delivering the certificate paths feature. This section examines each of these four features in turn, explaining what they are, how they deliver their services, and how these services address the vulnerabilities in NDP.

3.1.1. Cryptographically-Generated Addresses

Cryptographically-Generated Addresses (CGAs) are a method for generating an IPv6 address bound to an asymmetric key pair, specified in RFC 3972 for use in SEND [9]. This is done by generating a key pair and creating a data structure consisting of a randomly-chosen 128-bit value (used to allow multiple addresses to be generated from a single key pair if needed), the subnet prefix, an eight-bit collision counter and the public key. The specification includes the possibility for further parameters to be added in future versions. This data structure is then hashed using SHA-1, and the output is used as the interface identifier of the CGA, with the exception of the seventh and eighth bits, which are used as local/global and individual/group flags in EUI-64 IPv6 addresses [10], and the first three bits which are reserved for a security parameter which will be explained below. If DAD reveals that the address generated is already in use, a new one is created by incrementing the collision counter and repeating the process. If this occurs three times, the node will stop and report an error.

This allows the private key to be used to prove ownership of the CGA. In order to spoof a CGA successfully, an attacker must either learn the private key associated with the public key used to generate the CGA or generate another key pair and find a 128-bit modifier which can bind that key pair to the same address. Assuming the algorithm and key management are secure enough to protect the CGA owner's private key, this effectively means that an attacker must find a second preimage to the hash used to generate the address.

The specification of CGAs also includes a technique called hash extension, whereby the security of a CGA can be extended beyond that provided by the 59-bit hash used in the interface identifier. To achieve this, a three-bit security parameter (*sec*) is encoded in the first three bits of the interface identifier of each CGA. At the beginning of the CGA generation process, a value from 0 to 7 is chosen for this parameter, and a 112-bit SHA-1 hash is computed using the same set of parameters but with the subnet prefix and collision counter zeroed. For the parameters to be accepted, the first (16 x *sec*) bits of this hash output must be 0. If this condition is not satisfied, the 128-bit modifier in the parameters is incremented and another hash is calculated. This process is repeated until a hash output satisfying the condition is found, requiring on average $2^{16 \times sec}$ hash operations. The final version of the modifier is then used in creating the CGA.

Since the security parameter *sec* is encoded directly into the interface identifier, an attacker attempting to spoof the address must provide parameters which satisfy this same condition. However, the attacker will be using a different public key, as the private key corresponding to the target's public key is unknown. This means that the attacker will have to find a key pair and parameters which both satisfy the hash extension condition *and* result in the target IP address being generated, multiplying the difficulty of the attack.

The specification strongly encourages nodes using SEND to use CGAs, though it states that non-cryptographic addresses may be used for some purposes such as testing and diagnostics. Nodes using CGAs must provide the public key and parameters used as the hash input so that receiving nodes are able to verify that the CGA is valid. These must be included in the options field of any NS and NA messages the node sends, as well as any RS messages which are not sent with an unspecified source address [6].

3.1.2. Digital signatures

Digital signatures are required by SEND on all NDP messages except for RS messages with an unspecified source address. If the sender has a CGA, the digital signature must use the same key pair as was used to generate the CGA. This allows the sender to prove ownership of the private key associated with the CGA, as the digital signature can only be produced using that key; the signature therefore provides assurance that the sender is the rightful owner of the address. The verification key can be authenticated using a copy included in the owner's CGA parameters, using a valid certificate path as described in the next subsection, or both, depending on the message type and the receiver's configuration.

In combination with the CGAs themselves, these provide data origin authentication and minimise the protocol's vulnerabilities to attacks based on impersonating nodes (both host devices and routers) as well as the DAD denial of service attack. This accounts for the majority of the attacks discussed in section 2.3, but it does not address the possibility of an attacker posing as a router without address spoofing, taking advantage of the lack of router authentication in the NDP specification. This vulnerability is addressed by the next feature of SEND, certification paths.

3.1.3. Certificate paths

The method SEND uses to address the matter of router authentication is to specify that any router using SEND should carry a certificate authorising it to act as a router for a specific set of subnet prefixes, corresponding to (or at least including) the network to which the router is connected. This certificate will authenticate the public key of the router, allowing other nodes on the network to verify using the signatures on the router's messages that the messages originate from an authorised router.

The authority which signs the router's certificate can in turn be certified by a higher authority, as is the case with certificate authorities in SSL/TLS, leading to a chain of trust. Each host using SEND is configured with a set of trusted certifying entities, called trust anchors, and their verification keys. However, this is where one of the challenges mentioned in section 2.4 is encountered, as the host requires an Internet connection to find and verify the certificates authenticating the router's certifying authority.

In order to circumvent this, SEND requires routers to be able to assemble a full chain of certificates and send it directly to the host device. The integrity of the certificates is protected by their digital signatures, so the untrusted state of the router does not undermine this approach. This is accomplished in SEND by specifying two new messages: **Certificate Path Solicitation (CPS)**, which is sent by host devices to inform a router of which trust anchors the host is willing to accept and to request a certificate path leading back to one or more of those trust anchors, and **Certificate Path Advertisement (CPA)**, which is sent by routers in response to CPS messages to deliver a valid certificate chain leading to one of the specified trust anchors, if one exists. The certificate chains are split into multiple CPA messages with one certificate in each message and a component counter to allow reassembly; this is done to avoid IP fragmentation.

The certificates in the chain all include a set of permitted subnet prefixes, just as the router's certificate does, although these sets of prefixes may differ between certificates. Only prefixes included in every certificate in the chain are regarded as being protected by SEND, although a router may still advertise other prefixes which will be considered unsecured. Once a host device has received a certificate path leading back to one of its trust anchors and has verified that the certificates are valid and the subnet prefixes in the certificates match, that host can trust any message signed using the private key corresponding to the router's certified public key as having been sent by an authorised router.

3.1.4. Replay prevention

The last of the main features of SEND is designed to protect NDP against replay attacks. Such attacks were not included among the vulnerabilities I discussed in section 2.3, but the only reason for this is because any NDP message could be forged, making message replay unnecessary. With SEND's addition of cryptographic mechanisms, particularly those providing data origin authentication, it becomes important to prevent protected messages from being replayed by an attacker to subvert the assurance they provide.

SEND achieves this by specifying two new message options which together cover all of the messages of NDP. NS and RS messages in SEND are required to include a nonce in the options field to send a random value with the solicitation. SEND does not specify a length for this nonce, only that it must be at least six bytes and that the total length of the nonce option must be a multiple of 8 octets. NA and RA messages sent in response to a SEND node's NS or RS message must include the nonce sent with the NS or RS message in the options field of the advertisement, thereby proving that the advertisement is a fresh response to the recipient's solicitation and not a replayed earlier message.

The nonce option only provides assurance of freshness when two or more messages are being exchanged, so SEND also specifies a timestamp option for use with unsolicited NA and RA messages and with Redirect messages, providing an assurance of freshness in most cases for these standalone messages. The timestamps given in this option are specified as being 64 bits and based on real time measured in increments of $1/(2^{16})$ seconds.

3.1.5. Compatibility with non-SEND nodes

As noted in section 2.4, it is important for security measures not to cause incompatibility problems with nodes using an unsecured version of NDP, and the specification of SEND does consider this. SEND nodes are able to co-exist on a network link with nodes which do not use or understand SEND. Because SEND does not alter any of the pre-existing components of NDP, non-SEND nodes are capable of treating SEND-protected messages as though they were ordinary NDP messages and ignoring the additional SEND options which are not understood by the non-SEND node.

However, because unprotected messages may have been spoofed by an attacker, it is necessary to specify rules for how SEND nodes process unprotected messages, making a compromise between respecting nodes which do not understand SEND and ensuring that attackers have no opportunity to manipulate the SEND nodes. SEND therefore specifies that while SEND nodes may have an option to ignore unsecured messages, enforcing a SEND-only network, this should not be the default option.

SEND nodes are however instructed to prioritise information from messages protected by SEND over information sent without SEND or with an incorrect implementation of SEND. If information such as network settings or routing information conflicts, it is assumed that the information in an unprotected message is spoofed by an attacker while the

information contained in a SEND message is valid. Only if no SEND-protected information is available may a SEND node trust unprotected information out of necessity.

While performing DAD, SEND nodes accept unsecured messages claiming to own the tested address, but only for the first collision. For the second and third chosen addresses, only SEND-protected messages are accepted. This means it is possible that a node running SEND will choose an address which is already owned by a node which does not run SEND; however, with IPv6's 64-bit interface identifiers, the chances of even one collision are extremely low, so such an occurrence is far more likely to be the result of configuration errors or an attempted denial of service attack [9].

3.2. Drawbacks of SEND

Although SEND's security measures are very thorough and cover all of the major vulnerabilities in section 2.3, the protocol extension has few implementations, and several criticisms have been raised against it, suggesting that some improvements are still needed if it is to be used universally as is desirable for an NDP security extension.

One of the greatest criticisms of SEND is that it can be quite computationally expensive [8, 11, 12], as it requires the generation of an RSA key pair to claim an IP address and the generation and verification of RSA signatures with almost every message. It becomes even more so with a non-zero security parameter; with a security parameter of 0, generating a CGA only requires generating a key pair, selecting a random 128-bit number, and making a single SHA-1 hash computation. Increasing the security parameter to 1 adds an average of 2^{16} more hash computations to the process, and even this can result in troubling delays in some situation such as mobile device handovers [12].

It is worth noting, however, that this additional computation can be done in advance of the need to generate a new CGA [6], as it depends only on the node's public key and the chosen 128-bit modifier, not the subnet prefix or any other network-specific parameters. The generation of a CGA also does not require knowledge of the associated private key, only the public key, meaning it is possible to outsource the computation to other, more powerful devices; however, SEND specifies no mechanism for requesting this.

Another downside to SEND is that it suffers from extreme message expansion. It is required that all messages contain a digital signature and that most contain a public key and 128-bit modifier value as part of the CGA option. Overall, more than a kilobyte must be added to each message [8], which is a very large overhead compared to the small size of NDP messages. This could have serious implications for the bandwidth consumption of NDP messages, as some will be sent very frequently, especially in large networks.

Router authentication can cause similar issues as well, particularly if the certificate path being delivered is a long one. A long certificate path would require several messages, each one containing a full certificate along with a CPA header. In addition to being potentially expensive in terms of bandwidth, such a situation can require a substantial amount of

processing by the recipient, which may be a constrained device. Fortunately, router authentication will not be needed frequently compared to other NDP messages.

Concerns have been expressed about the fact that SEND is bound to the use of specific algorithms for generating CGAs and signatures [11, 13, 14]. The specification of SEND does not outright require the use of RSA; however, RSA is recommended as the default algorithm, and implementations are strongly discouraged from using other algorithms for reasons of compatibility [6]. Meanwhile, the specification for CGAs mandates the use of SHA-1 [9]. Although there are already attacks against the collision-free property of SHA-1, this does not directly affect the security properties needed by SEND in its current state [13]. Even so, if attacks are found against these algorithms in the future, the security of SEND as a whole could be compromised. This is an increasingly important consideration now as the emergence of quantum computers is expected to severely weaken RSA.

Another drawback is that it is possible to launch attacks against the SEND protocol itself. Most of these attacks are denial of service attacks which take advantage of the high processing costs of some SEND features. For example, the process of creating and verifying certificate paths can be very computationally expensive, so an attacker can perform a denial of service attack against a router by making numerous CPS requests involving many diverse trust anchors, or against a host by masquerading as a router and sending a series of CPA messages with a large certificate chain containing many dead-end paths, forcing the host to waste large amounts of time and energy attempting to verify the chain, even if the chain is ultimately invalid [6]. Denial of service attacks such as this are difficult to mitigate, however, except by reducing the processing costs of these messages.

Finally, SEND does not protect against all forms of attack. The specification [6] notes that it does not attempt to provide confidentiality for any messages and cannot compensate for an unsecured link layer allowing attacks such as link-layer address spoofing. AlSa'deh and Meinel [8] also note that CGAs provide no assurance of the node's identity when they are first claimed, and so further measures are still needed if the link is not intended to be open to all devices. Confidentiality and authentication of joining nodes are not necessary to provide the core functions of NDP, however, so while these could be included as configuration options in an NDP extension, it would be equally appropriate to leave these features to be provided by separate systems. Notably, confidentiality cannot be provided without making messages incompatible with nodes running unsecured NDP, as unsecured nodes have no means to decipher confidential messages, so any provision of confidentiality would have to be optional.

Overall, SEND provides good protection against many of the threats against NDP, particularly those involving address spoofing, but it does so with the disadvantage of high processing costs and communication overheads which may be unacceptable on low-power systems such as mobile and IoT devices. However, despite these drawbacks, SEND remains the most complete and developed system to protect NDP and would make an excellent basis for this paper's recommendations. There are numerous proposals to improve SEND by addressing these problems, which are the focus of section 4, but first this paper considers some of the alternatives to SEND, comparing them with SEND in order to justify the decision to build upon SEND instead of adopting an alternative system.

3.3. Alternatives to SEND

Ahmed et al. [4] identify numerous proposals to protect NDP, both those which build upon SEND and those which intend to replace it. Of those which do not build on SEND, several are designs for intrusion detection systems; any compatible IDS could readily be deployed in addition to SEND or a similar extension to NDP, but this paper is more concerned with developing the protocol to be resilient against attacks even in the absence of an IDS, and to be more practical than SEND in its current state.

Some others rely on centralised network architectures, defeating the intentions of SLAAC, and some would be incompatible with nodes using an unsecured version of NDP as they use modified versions of some NDP messages which unsecured nodes would not understand. Still others, such as those proposed by Praptodiyono et al. [15, 16], are partly based on unsound principles, such as using unencrypted hash outputs to detect deliberate message modifications. This section examines some of the proposals with the most merit, out of those focused on directly improving NDP.

3.3.1. Simple Secure Addressing Scheme

The Simple Secure Addressing Scheme (SSAS) [17] is proposed as an alternative to SEND, intending to provide a more lightweight security extension in response to the high computing costs of SEND's address generation and router validation procedures. The measures specified by SSAS cover the same areas of NDP as SEND: address generation, proof of ownership of addresses, and secure router authentication.

Like SEND, SSAS specifies that a node performing SLAAC should generate the interface identifier of its IPv6 address in a specific way using the public key of an asymmetric key pair, so that ownership of the address can be asserted by proving knowledge of the private key. Unlike SEND, however, SSAS does not recommend the use of RSA. Instead, SSAS suggests the use of ECDSA as the default algorithm but accepts the use of any algorithm with a similarly small key size and short generation time. According to the proposal, using ECDSA reduces the length of time required to generate an address by a factor of 4 and very significantly decreases the bandwidth overhead required to send public keys, but increases the time taken to generate and verify signatures, when compared with using RSA for an equivalent level of security.

SSAS uses a somewhat simpler algorithm to generate its addresses than the CGA algorithm used by SEND. First a key pair is generated for the chosen algorithm, for example ECDSA. The public key must be at least 192 bytes long; it is then split into two half-byte arrays each containing half of the key. The first four bytes of each array are taken and concatenated to create the interface identifier, which is combined with the subnet prefix and tested through DAD. If a collision is detected, another interface identifier is generated from the second set of four bytes in each array, and if another collision occurs the third set of four bytes in each array is used.

One problem with this scheme is that the subnet prefix is not used in generating the interface identifier, meaning that any work done to steal an address can then be reused across all networks since the same key pair will bind to the same interface identifier regardless of the subnet prefix. Another potential problem is that each bit in the interface identifier corresponds to a specific, known bit in the public key. This means that a set of all public keys for which a given address would be valid can easily be constructed. This does not directly reveal any private keys to which the address could be bound, but depending on the other parameters under an attacker's control, it may make a spoofing attack easier. Certainly it reveals more information than is necessary, compared with SEND's less revealing approach of using a well-known hash algorithm to generate the interface identifier. It also does not result in a significant increase in efficiency, as it only eliminates one SHA-1 calculation, which is not an intensive computation compared to key generation.

Additionally, SSAS provides no equivalent feature to SEND's hash extension technique. This means that the useful lifetime of SSAS is limited by the feasibility of a brute-force attack on the binding between random key pairs and the target IP address. This attack is less feasible than a similar attack on SEND CGAs, since SSAS has no 128-bit modifier and therefore requires the generation of a new key pair after every three attempts (one for each possible value of the collision count). There are 2^{62} possible addresses in total, since two bits are used as flags and therefore do not depend on the public key, so this brute force attack will require generating on average between 2^{60} and 2^{61} key pairs, divided by the number of viable targets across all networks the attacker has access to. Once such an attack becomes feasible, SSAS in its current state will no longer be effective.

These problems could be solved by employing a hash algorithm as part of address generation, using the subnet prefix as an input, and adding a hash extension as in SEND. However, this results in a very similar design to that of address generation in SEND and seems to offer no benefits over modifying SEND directly to use ECDSA.

Meanwhile, SSAS also specifies an approach to router authentication which is quite different to that used in SEND. Networks supporting SSAS would have a Controller Node (CN) connected to the network, which stores the addresses and public keys of the network routers, as well as those of any nodes which communicate with it. The proposal specifies two new messages, Request Public Key (RPK) and Send Public Key (SPK), which are used by other nodes and the CN respectively in an exchange which allows nodes to obtain the public keys stored by the CN.

This allows nodes to learn the public keys of other nodes from the CN instead of from the owner of the key. This effectively prevents the IP spoofing attack which relies on finding another key pair which can bind to the same address, as even if such a key pair is found it will contradict the information stored by the CN. However, caching public key information from the CN greatly increases the size of each entry in a node's neighbour cache, which may cause problems under tight storage constraints.

This approach also means that the authenticity of routers on the network is vouched for by the CN, but this only defers the problem. SSAS does not address how the CN should be

authenticated. The proposal states that the CN has a known, static IP address, but this can easily be spoofed unless the CN's public key can also be verified. The proposal describes this scheme as a "local centralized Resource Public Key Infrastructure (RPKI)", suggesting it is intended either to be treated as a root authority or to be verified using the RPKI; in the former case, a node would have to be specifically configured for the network to know and trust the CN's public key, and in the latter case a system similar to that used in SEND is still necessary to verify the CN's RPKI certificate.

The presence and role of the CN also enforces a centralised structure on the network and is therefore not desirable in network types which favour decentralisation. It offers very few advantages over a secure CGA scheme, as it does not adequately solve router authentication and CGAs already minimise the IP spoofing vulnerability. For these reasons, it is not preferable to SEND for the purposes of this paper. However, the functionality of the CN is still very useful in centralised networks which do not use CGAs and might therefore make a good addition to DHCPv6 servers.

3.3.2. Source Address Validation Improvement

RFC 7039 [18] proposes a filtering-based security mechanism called Source Address Validation Improvement (SAVI). This mechanism is specifically designed to combat IP address spoofing and is similar in design to ingress and egress filtering. However, since NDP runs solely within a network, SAVI cannot operate on the boundaries of a network but must instead be applied inside the network. In order to minimise the proportion of messages which bypass filtering, it is important for SAVI to be run as close to each host as possible, on switches placed at bottlenecks in the network's physical topology. Multiple filters can be used in a single network, each filtering only the messages on interfaces which are not connected to another filter device to avoid duplicating information.

Each switch running SAVI creates a table in which it stores the authorised IP address of each node once it is learned, associated with a link-layer property of the host device which owns the address, called a binding anchor. Several binding anchors are possible, depending on the underlying technology, but it must be possible to verify them in every packet, they must be as unique as possible to each host, and they must be harder to spoof than the host's IP address. The authorised address of each host can be learned by observing DAD or message exchanges with a DHCP server, depending on which methods for assigning addresses are in use on the network.

Any packet passing through the SAVI device with a specified source address will be dropped by the filter if the address is not in the SAVI device's table or if the binding anchor associated with the message does not match the anchor stored in the address's row in the table. This means that any attempt to spoof an IP address on the network will also require the attacker to spoof the link-layer property of the address's owner in order to succeed, unless the spoofed message does not need to pass through a SAVI filter to reach its destination on the network. The difficulty of doing so will depend on the binding anchor chosen and the link-layer security mechanisms used on the network.

A SAVI device might lose data from its table or fail to include a valid address, and therefore start rejecting legitimate messages. This could happen if, for example, the device running the filter restarts, or if the link topology changes, causing a host to be connected to a new filter device which does not have its IP address stored. To recover from this, the specification includes that each device running SAVI should be able to detect when too many packets are being dropped, and to recover by first checking whether the source address of the dropped packets is unique on the network, either through DAD or by querying a DHCP server, and then add it to the binding table if it is. This check ensures that even if the packets are illegitimate, there is no target for them to spoof, and so there is very little damage they can cause. Note that if DAD is used, this requires the filter device to spoof the address, so this aspect of SAVI is incompatible with SEND-only networks.

The proposal acknowledges, however, that SAVI cannot provide a strong guarantee of security. Under many circumstances, the binding anchors available to be chosen either will be vulnerable to spoofing or will not be unique to each host. It is also possible that the filtering devices themselves will become targets of attack, especially if they are physically accessible, which is likely to be the case if they are placed close to the host devices as is ideal for the digital security of the scheme. SAVI is also not very suitable for networks in which the topology frequently changes, such as wireless networks with multiple access points, as this will cause frequent packet losses as a result of hosts moving to a different filter device which has not learned their addresses and binding anchors.

However, one advantage of SAVI is that it introduces no new protocol, and requires no action by hosts; this means that it could be used to improve the security of networks for devices which do not understand any security protocols for NDP, such as those which were created before any such protocols were available and those lacking the resources to run the specified security mechanisms.

3.3.3. Trust Based Security

Praptodiyono et al. [15] propose another mechanism for verifying the trustworthiness of routers which does not rely on certificates but instead on a distributed trust model. In this proposal, called Trust Based Security (TBS) for IPv6, nodes gain some assurance of the authenticity of a router without requiring any information from outside the network, based on the assumption that a significant number of other nodes in the network will trust the router if and only if the router is trustworthy.

TBS specifies new options to carry its functionality, the Trust Solicitation Option (TSO) and Trust Advertisement Option (TAO). Unfortunately, the proposal does not make the details of how these are used very clear. What is clear is that TSO is used with RS and NS messages and TAO is used in the corresponding RA and NA messages. The two options each include a nonce so that the response cannot be replayed. The specification also makes the mistake of attempting to use a SHA-1 hash to detect deliberate modifications to the message. Neither of these components would be necessary if TBS were to be used in conjunction with the spoof- and replay-prevention features of SEND.

The main feature of TBS, however, is its method of authenticating routers. Once again, the details of how this should be done are not made clear, but the intention of the process has been described. After exchanging RS and RA messages with a router, an inquiring node sends a solicitation to all other nodes on the network to request router information from them. The router information received in the responses to this message is compared with the information received directly from the router, and only if the two sets of information sufficiently match will the router be used; otherwise the information is discarded and the node restarts the process.

This is an attractive option for trust management if it can be made to work reliably, as it relies only on local information which does not have to be delivered through an untrusted router, unlike certificates which must be verified over the Internet. Unfortunately, this approach is much easier for an attacker to subvert than a scheme based on certificates, as it relies on the assumption that the majority of nodes on the network are trustworthy. This means that TBS must be used in conjunction with other security measures to prevent attackers from forging malicious router information from a large number of false addresses. CGAs are not able to prevent this attack except by using a high *sec* value, as it is not necessary to spoof a specific address, only to generate a large number of new addresses. Even if this is prevented, TBS can be subverted if the attacker can infiltrate the network with multiple devices.

There are two main ways in which the scheme can be attacked, and the risk of each will depend on how TBS is configured. Each node will have to set a threshold which will be used in deciding whether or not to trust a router, based on the proportion of other nodes on the network whose solicited router information matches the router under scrutiny. Even in a scenario with no attackers, a threshold of 100% is unlikely to work, since some hosts will be using different routers if more than one is available on the network. This makes it difficult to know where to set the threshold, especially because there is likely to be no way of knowing how many legitimate routers are on the network. Setting the threshold too high also allows an attacker to make a denial of service attack by infiltrating the network with devices which discredit legitimate routers by providing false router information. However, if the threshold is set too low, it becomes more likely that a malicious router will be accepted as the host's default gateway router, especially if the attacker infiltrates the network with nodes which vouch for the false router.

There is therefore a security trade-off in the configuration of TBS, and it gives a lower assurance of security than a certificate-based mechanism. There is also a comparatively high bandwidth consumption associated with this protocol exchange, as it uses a multicast message which solicits a response from every recipient. This problem is exacerbated by using TBS in conjunction with CGAs, as this would require every message to include a digital signature and CGA options. Like SAVI, however, TBS seems to be a reasonable best-effort strategy for when more secure methods are impossible. While the proposal for TBS itself is poorly constructed and not very detailed, this underlying concept could be developed into an acceptable alternative for when certificate-based verification cannot be made to work within a particular network's constraints.

4. Proposals to improve SEND

In the previous section, some of the most promising alternatives to SEND are considered and their own advantages and disadvantages compared with SEND are examined. None of these alternatives provide as much security as SEND, however. SAVI and TBS have significantly lower processing and bandwidth costs, but some of their security shortcomings are difficult to address as they result directly from the design of these systems. The major vulnerabilities in SSAS, meanwhile, result from the choice of cryptographic mechanisms, which is much easier to improve. However, SSAS is very similar in design to SEND, and in creating a compromise between the two it makes most sense to modify SEND, as there are already implementations of SEND with which compatibility should be maintained if possible.

For these reasons, the framework proposed in this paper uses SEND as its basis. However, SEND has numerous drawbacks as discussed in section 3.2, and these must be addressed to ensure that SEND will be practical enough to be widely used. Numerous proposals already exist to improve on these shortcomings, which in this section will be drawn together and discussed in order to determine which improvements would be most useful and would combine well to cover SEND's weaknesses efficiently.

4.1. Alternative algorithms

4.1.1. Elliptic-Curve Digital Signature Algorithm

One of the primary concerns with SEND is the high computation costs associated with creating CGAs and digitally signing messages, and to address this complaint several authors have investigated the possibility of using an algorithm other than RSA in SEND. One of the most popular algorithms suggested is ECDSA, which is compared with RSA in multiple publications. Rafiee and Meinel [17] conducted an experiment, the results of which show that using ECDSA results in much faster CGA generation, but increases the time taken to generate and verify digital signatures. This is a poor trade, as in normal use of SEND signature generation and verification will be taking place far more frequently than CGA generation.

However, it is worth noting that the data gathered by Rafiee and Meinel was from only ten samples, and other studies contradict the results. In a similar experiment by Cheneau et al. [11], the results show ECDSA having a shorter generation time for signatures than RSA, though signature verification remains longer than both RSA signature verification and ECDSA signature generation. The study notes that this is in line with expectations for ECDSA, which is generally known to have slower verification than generation, and that this is a potential disadvantage particularly in multicast messages which require multiple nodes to verify each signature. Messages which are quicker to generate than they are to process may also risk being used in denial of service attacks. On the other hand, Cheneau et al. also call attention to ECDSA's smaller key sizes, which reduce message length and storage requirements. This is likely to outweigh the drawbacks in some circumstances,

such as in wireless networks where data transmission is a key factor in the energy consumption of resource-constrained devices [19].

Some doubt is cast on these results as well, as another study by Qadir et al. [20] displays vast improvements in the signature generation and verification times for RSA by using the RSA implementation in PolarSSL, a lightweight implementation of SSL/TLS written specifically for embedded systems, instead of that in OpenSSL. The results show signature generation speeding up by a factor of more than 1,000 over the data from Cheneau et al., far outclassing the generation time for ECDSA in the earlier study as well. However, this data was also gathered on a different platform, using Nokia 900 instead of Nokia 800 as was used by Cheneau et al. No data is gathered for ECDSA on Nokia 900, so it is unclear how much of the difference in results between these two studies occurs because of the RSA implementation and how much occurs because of the difference in platform.

Overall, it is unclear to what extent ECDSA offers processing advantages over RSA. All sources agree that signature verification takes longer in ECDSA than in RSA, as RSA signature verification is an extremely cheap operation, and this could cause some difficulties in NDP. However, ECDSA has a major advantage in its ability to use much shorter keys for the same level of security, reducing the message expansion caused by SEND and making it a good alternative to RSA in many contexts.

4.1.2. Feige-Fiat-Shamir

Castellucia [19] proposes another alternative, describing a system based on the Feige-Fiat-Shamir (FFS) signature scheme in place of RSA. Specifically, the proposal uses a small-primes variation of FFS, which reduces the verification time and public key size. The author compares this system with RSA and DSA, though not ECDSA, and shows that when the security of the signature is tuned to the security of the CGA, generating a signature with small-prime FFS requires less than a fifteenth as many modular multiplication operations as RSA uses, resulting in a very significant reduction in the time taken to generate and even verify signatures.

FFS does result in longer key generation, but as the author notes, this can be done offline and is not time-critical, so this is not a major disadvantage. FFS also has larger key sizes than RSA; the private key's size is several kilobytes in size and grows larger with a higher *sec* value in the CGA, but since this does not have to be transmitted it is likely to be acceptable for most if not all devices.

More important is the size of the public keys and signatures, as this directly affects message expansion. With small-primes FFS, the public keys are approximately twice the size of RSA public keys; however, there is a method which allows the same set of primes to be used for every key pair, reducing the public key size to that of the modulus, which is similar in size and effect to an RSA modulus, therefore making the FFS public key no bigger than an RSA public key. FFS signatures are also only a few bytes longer than RSA signatures, so there is no significant disadvantage in message expansion to using FFS compared with RSA; there is also no improvement, however, as there is with ECDSA.

All this suggests that small-primes Feige-Fiat-Shamir would be a preferable signature algorithm to RSA, but its processing requirements have not been directly compared with ECDSA. ECDSA is still likely to be better for wireless networks because of its lower transmission overheads. Another downside of FFS is that it relies on the same hard problem as RSA, that being factorisation of the large modulus. This means that any attack against RSA, including Shor's algorithm, is likely also to affect FFS.

4.1.3. Merkle Signature Scheme

Qadir et al. [21] propose a system built around hash-based one-time signatures, considering the context of mobile networks in particular. This system is suggested because these signatures are expected to be resilient against quantum cryptanalysis. Their proposal suggests using the Winternitz one-time signature scheme, as it includes a way to balance the computational cost against the signature length as needed. The major drawback with this scheme is that, as its name implies, each key pair can only be used to sign a single message. This is completely impractical for SEND, which not only requires a signature for every message but binds the public key into the node's IP address.

To overcome this problem, Qadir et al. suggest a system based on Merkle trees. These are complete binary data trees in which each leaf node is a hashed public key for an underlying one-time signature scheme, in this case the Winternitz scheme. The rest of the tree is then constructed by hashing each pair of nodes and storing the output in the parent node, repeating this process until the root node is reached. This root node is used as the public key for the entire Merkle tree; this hash value depends on all of the one-time public keys, so each signature can be accompanied by its verification key and the hashes of the other verification keys. The verifier can then use this information to reconstruct the Merkle tree and compare the root node with the stored public key to authenticate the verification key sent with the signature.

The Merkle signature scheme has a significantly shorter signature generation time than RSA, but the verification time can be very long, depending on the depth of the Merkle tree. This is because verifying each signature requires reconstructing the entire tree from the leaf nodes, which requires $2^n - 1$ hash operations for a tree of depth n. The scheme also requires that all of the leaf nodes be sent with each signature (except for the one representing the key in use, for which the preimage must be sent instead). If a hash algorithm with 256-bit outputs is used, as [21] recommends for security reasons, this adds $32 \times (2^n - 1)$ bytes to each message, not including the size of the signature itself or the verification key. This alone grows larger than RSA and FFS signatures when n exceeds 2. Increasing the number of verification keys bound to a single public key also increases the number of hashes attackers can find a collision with in order to forge signatures, thus weakening the security of that key.

This suggests that a small Merkle tree is preferable, but the downside to this is that it decreases the number of messages that can be signed using that public key. Generating a new key is not a major problem, since the processing required to do so decreases

proportionally with the number of messages that can be signed between each generation, but since the public key is bound into the CGA, generating a new one necessitates a change in IP address, which could be inconvenient if it must be done very frequently. There are advantages to this, as it decreases the amount of time available to an attacker to steal the node's CGA by finding a collision; however, this scheme is most likely to be used by client hosts rather than routers or servers, since a node with a frequently-changing IP address and public key will be difficult to authenticate, and client hosts are unlikely to be the targets of expensive brute-force attacks [9].

The long verification times of Merkle signatures can also be problematic, particularly in conjunction with the short generation times, as this can give rise to a denial of service attack based on flooding the target with large numbers of signed messages which must be verified. Qadir et al. note that this is prevented by the one-time nature of the signature keys; however, this assumes that the recipient is maintaining state to track which keys have already been used. SEND does not require the verifier to store any key information; the public key must be sent with every message anyway, since there is no guarantee the recipient has observed earlier messages from the sender. In the Merkle scheme, this extends to all of the hashed verification keys as well. However, this same fact means that there is no benefit to condensing the keys into a single hash using the Merkle tree structure, so the verification time problem can be solved by instead simply using the full set of hashed verification keys as the public key in this scheme.

Unfortunately, the underlying one-time signature scheme also presents a problem. The Winternitz one-time signature algorithm has very long verification keys and signatures when compared with RSA; the minimum length quoted by Qadir et al. is 2560 bits for each, but this requires a very large amount of processing, so in practice it would likely be necessary to use longer signatures to reduce the processing. This will cause much greater message expansion than even RSA and may even lead to storage problems if the sender generates and maintains too many keys at once. However, this scheme is believed to be quantum-resistant unlike RSA, FFS and ECDSA, and it would be useful to specify at least one quantum-resistant algorithm for use with SEND as early as possible.

4.1.4. CGA hash algorithms

The signature algorithm is not the only algorithm in SEND which proposals are seeking to replace. Similar concerns exist for the hashing algorithm which is used to produce CGAs, SHA-1, which if broken would undermine the security of SEND's anti-spoofing features. The current attacks against SHA-1 reduce the algorithm's collision resistance and thereby undermine its ability to provide non-repudiation assurance, but this property is not required by SEND, as the security of the CGA against spoofing relies only on second preimage resistance [13]. However, concerns still exist that there is no quick way to change the algorithm used in SEND if SHA-1 is weakened further in future.

RFC 4982 [13] proposes a method to allow SEND to support multiple hash algorithms. The challenge here is that the owner of a CGA must have a way of making other nodes aware

of which hash algorithm was used to generate it, otherwise it cannot be verified and so offers no security. This could be done simply by including a hash algorithm identifier as one of the parameters used as input to the CGA hash, but doing this is not enough to protect the address, as another set of parameters using a different algorithm may still result in the same hash output, so attackers will still be able to use the weaker algorithm to find such a collision.

It is therefore necessary to encode the hash algorithm directly into the address itself. However, repurposing any bits from the CGA hash to encode this will degrade the security of the CGA by effectively shortening the hash used to generate it, and will also cause incompatibility with previous versions of SEND as these repurposed bits will most often cause CGA verification to fail. RFC 4982 therefore proposes repurposing the three bits used to encode the security parameter to instead be an identifier for a combination of *sec* value and hash algorithm.

This approach will mean making several of the higher *sec* values unavailable, but these *sec* values were intended for future use when attacks on the lower values become practical [9]. Currently only the few lowest values of *sec* are in use, as the higher values require too much processing for CGA generation to be practical. It is quite possible that an attack on SHA-1 will invalidate the current scheme before the higher values of *sec* become useful, so repurposing the *sec* bits of the address in this way may extend the lifetime of SEND.

This method allows up to eight of these schemes to be specified at any one time. RFC 4982 suggests continuing to use 000, 001 and 010 to represent SHA-1 with *sec* values of 0, 1 and 2 respectively; this ensures backwards compatibility for these values, as this is the same meaning these parameter values hold in current SEND implementations. The proposal makes no suggestions for subsequent values of the parameter, but it is important to specify at least one alternative algorithm in advance of SHA-1 being deprecated. This is because even if SHA-1 becomes insecure, there will be a transitional period during which some nodes have not been updated, and these nodes will need to be able to interact with updated nodes using the new algorithm.

This comes at the expense of no longer being able to use all eight possible values of *sec* in the same implementation. However, this ability is not a useful one, as only a subset of these values will be both secure and practical at any given time. It is easy to see, for example, that any *sec* value x will be insecure if the *sec* value $x + 4$ is practical. This is because increasing *sec* by 4 multiplies the average number of hash operations needed to generate a CGA by 2^{64}, whereas attempting to spoof a specific address using brute force requires only 2^{58} times as many hash operations on average to find a CGA collision. Considering that a CGA will typically be expected to have a cover time significantly longer than its generation time and that the processing power of an attacker will often be greater than that of the victim, it would be reasonable to extend this result to state that any *sec* value will be insecure if the highest practical *sec* value is at least 3 values higher.

The SHA-2 family and SHA-3 are obvious candidates for alternative hash algorithms. SHA-3 has the advantage of being designed in a fundamentally different way to SHA-1 and SHA-2, meaning it is unlikely that any attack against the earlier algorithms will affect

SHA-3; SHA-3 is a significantly slower algorithm, however [21]. This will significantly affect the efficiency of CGA generation with non-zero *sec* values, potentially making SHA-2 more attractive for mobile hosts, but also reducing the efficiency of brute-force attacks and thereby improving the security provided by higher *sec* values using SHA-3. It might therefore be reasonable to specify configurations for both SHA-2 and SHA-3, using SHA-2 with lower *sec* values for contexts in which efficient CGA-generation is needed, and SHA-3 with slightly higher *sec* values for when security is a more important concern.

If multiple configurations are provided for each of SHA-2 and SHA-3 with different *sec* values, this could easily fill the eight available values for the configuration bits, preventing the use of any other algorithms or higher *sec* values. It will therefore be necessary to recycle old values when it becomes useful to add new configurations [13]. This will cause some compatibility problems during the transitional period when it occurs, and the new configurations should go unused for some time if possible, because a significant proportion of nodes will still be using the old version and will understand the recycled parameter value differently, which will usually mean that new nodes and old nodes will be unable to verify each other using the recycled parameter value and will consider each other to be unsecured as a result. However, by recycling old values in this way, it should be possible to prolong the proposed system far beyond what would be achievable otherwise.

4.2. Multi-key CGAs

A further shortcoming of CGAs, noted in RFC 5909 [22], is that they are incompatible with Neighbor Discovery proxies. Proxies are nodes which advertise the virtual presence of a node which is not physically connected to the network and forward messages to and from the node they are proxying for. Proxies are used in Mobile IPv6 as home agents, allowing a mobile host to reach its home network while it is away, as well as being used as relays in networks which extend a single subnet prefix to multiple physical network links.

In order to establish itself as a proxy for another device, a proxy node must be able to multicast an NA message to update other nodes' neighbour caches to use the proxy node's address in place of the client device's address. However, SEND requires that this message include a signature using the client device's private key, which the proxy node does not possess. This means that the NA message must be sent unsecured, but if the client device is using SEND then this will not update the neighbour caches of any recipients using SEND, as the client's address will be registered as secured information, and the unsecured information in the NA message will not be allowed to update it.

An experimental update for SEND was published in RFC 6496 [23] to allow devices to be certified to act as proxies, similarly to how SEND routers are certified. This makes use of an alteration to SEND certificates in RFC 6494 [24] which allows a certificate to specify the purpose for which a public key is certified to be used, such as for proxying. The update also specifies a new message option, the Proxy Signature option, which proxies use to sign proxied messages, replacing the CGA and RSA Signature options appended by the originator of the message.

Use of the Proxy Signature option notifies recipients that the verification key must have a known and verified certificate; the key is identified with a 128-bit hash included in the Proxy Signature option, but not directly included in the message. Recipients must therefore be aware of the proxy node and its certificate before they can verify proxied messages; this could be a disadvantage, as recipients which are not aware would need to solicit and verify a certificate path, introducing a substantial delay to any response they might be required to make. Cheneau and Laurent [14] also argue that this method gives too much power to the proxy node, making it an attractive target for attackers to compromise.

Kempf et al. [25] suggest a solution which makes use of CGAs generated using the public keys of more than one node, called Multi-key CGAs (MCGAs), to establish proxies. Messages sent from MCGAs can then be signed using a group signature algorithm, which only requires the private key of one participant to create the signature but requires the public keys of all participants for verification. It is impossible to determine which participant produced a given signature, so when this method is used to establish ND proxies it also conceals whether the client is in its home network or communicating via proxy at any given time.

The algorithm suggested by Kempf et al. is the Rivest-Shamir-Tauman (RST) ring signature algorithm, which has the advantage over other group signature algorithms of not requiring one participant to act as a 'group manager', which in other algorithms has the power to break the anonymity of the signature. It is also very easy to set up an ad-hoc ring signature group, and a node can form a group without the active participation of the other members, provided the other members' public keys are known.

The main downside of this MCGA scheme is message expansion, as multiple public keys must be used as input to the CGA and provided in the CGA option, and the size of the group signature increases with the number of participants in the group. The need to use the public keys of all group participants to generate the MCGA also means that participants cannot be added or removed without needing to generate a new address. It is also necessary to ensure that all participants generate good keys, as the security of a ring signature group is equal to that of the weakest of the key pairs involved. Finally, the proposal by Kempf et al. only introduces the signature scheme, and does not specify any method for establishing trust for proxy nodes.

Cheneau and Laurent [14] build on the ideas expressed in RFC 6496 and by Kempf et al., drawing them into a single scheme whereby a node can learn that a router is authorised to act as a proxy from its certificate as in RFC 6496 and can subsequently use the router's public key, acquired during router discovery, to create a MCGA with the router as a participant. The node can then use this router as a proxy at any time during the lifetime of this address if it wishes to remain connected to the network after physically moving to a different network link. It is worth noting that this still relies on using a certified router as a proxy, which may be compromised by an attacker to allow impersonation and man-in-the-middle attacks.

Cheneau and Laurent advocate the use of ECDSA instead of RST ring signatures for this scheme, as ECDSA's much shorter keys make up for the need to include multiple keys in the CGA. ECDSA is not a group signature scheme, so it allows a fixed signature size regardless of the number of participants in the MCGA group, but it sacrifices the benefit of the signatures being identical regardless of which participant generated them. Cheneau and Laurent argue that this feature was not as beneficial as it seemed since the client host's location privacy was undermined by the unconcealed link-layer address, which would be different depending on whether the message was sent onto the network by the host itself or by the proxy router.

Cheneau and Laurent also note that MCGAs can be used to include public keys for multiple algorithms, for the purpose of negotiating in heterogenous networks in which different nodes may have different sets of signature algorithms they are able to verify. Receivers could process this identically to proxy keys, as all that is needed in either case is to identify which key applies to the signature on the message and use it. However, the usefulness of this is debateable, as SEND does not have any mechanism for making this negotiation, and many NDP messages must be multicast. It would therefore be necessary to send many messages with multiple signatures, in addition to sending multiple public keys with each message, leading to a very large amount of message expansion. This would negate the main advantage of using ECDSA and would generally be an undesirable outcome. It is useful to have this capability in case it becomes necessary, but it would be better to keep the number of signature algorithms in use to a minimum and ensure as far as possible that all implementations can verify all standard signatures.

4.3. CGA++

Another small proposal to change the generation of CGAs is given by Bos et al. [26], who identify two methods of attacking SEND across multiple networks and suggest changes to the generation of CGAs to prevent these, calling this new method CGA++. The first attack identified is a time-memory trade-off attack which can be used to eliminate the effect of the hash extension technique by generating a list of parameter sets which pass the hash extension test in advance and storing this list for future attacks. The particular problem Bos et al. are concerned with is that this database of parameter sets would be applicable in any network, since the subnet prefix is specifically excluded from the computation of the hash extension.

This attack would require a very large amount of storage, as there would need to be enough entries in the database to generate the addresses for a brute force attack on the 59 hash bits of the CGAs themselves. Most of these entries would only need to be 128-bit modifier values, with only occasional changes to the other parameters, and each entry can generate three unique addresses by varying the collision count in the CGA parameters. An attempt to impersonate a specific node therefore requires on average $2^{59}/3$ entries, each 128 bits long, with some comparatively negligible additional data for other parameters, for a total of $2^{63}/3$ bytes or approximately 2.7 exabytes. This is currently an infeasible amount of storage for such an attack, but it becomes far easier if the attacker does not need to

impersonate a specific node, in which case the average number of entries needed will be divided by the number of nodes on the network; for example, to impersonate a random node in a network with 2^{16} nodes, the database only needs approximately 43 terabytes of storage (this differs from the figure given in by Bos et al. as their calculations do not include the effect of varying the collision counter).

This is a very expensive attack, but still a viable one, especially because the same database can be used for all networks. For this reason, Bos et al. recommend changing the hash extension to include the subnet prefix in the calculation of the hash, ensuring that different networks will require different parameters to pass verification with a non-zero *sec* value, and therefore that a different database will be needed for each network. One drawback to this is that it will be incompatible with pre-existing SEND implementations whenever a non-zero *sec* value is used, as the new version will fail verification of the hash extension by the old version and vice versa.

Another downside to using CGA++ is that it will no longer be possible for a device to reuse the parameters from its old address when moving from one network to another, assuming a non-zero *sec* value is to be used, or to calculate the parameters for the hash extension test in advance without knowing the subnet prefix of the network it will be joining next. This could be a problem for mobile devices which need to be able to perform quick handovers from one network to another and could otherwise perform the bulk of the processing to find a new IP address before it becomes necessary to hand over. All but the lowest *sec* values are too impractical to use in this context; according to Qadir et al. [21], in 2015 a CGA with a *sec* value of 1 took approximately 400ms to generate on a mobile device, while a CGA with a *sec* value of 2 took more than 90 minutes. This is not a major vulnerability yet, as CGAs with security parameters of 0 and 1 are still difficult to spoof; however, RFC 3972 [9] notes that increasing the computing power of mobile devices is often not a priority, so higher *sec* values may remain impractical even when lower ones become insecure.

One possible method to avoid this problem when it arises is to select a temporary address with a *sec* value of 0 when joining the network, and to change to a more secure address when possible. A device could also request that the processing necessary to generate the new address be done by another, more powerful device on the network [6], although there is currently no mechanism for doing this in SEND. To do this, the mobile device would need to send a message containing its public key and other CGA parameters, and the recipient would reply with a suitable modifier value which the mobile device should verify before use.

Theoretically, if CGA processing is outsourced in this way, a malicious device could deliberately select a CGA for the host for which a colliding set of parameters is known. This attack would require less processing than attempting to spoof a specific address with the same security parameter, but it would also have to be carried out in a much smaller amount of time. It might be affected by any weaknesses in the collision resistance of the hash algorithm, such as those which exist in SHA-1. A collision could also be found in advance if the victim's public key is known, so any CGA for which generation is outsourced should use a new key pair.

The other attack discussed by Bos et al. [26] is a form of replay attack which crosses network boundaries by copying signed messages sent by a node on one network, then constructing an address using the same public key under a different subnet prefix and replaying the signed messages in order to feign knowledge of the private key and spoof the address. This attack does not work, however, because digital signatures in SEND are computed over not only the NDP message but also the source and destination addresses, including the subnet prefix. A signature which is valid on one network is therefore not automatically valid for the same message on a different network, and the attack fails. This means that knowledge of the private key is still necessary to sign messages, including DAD messages, so the other change proposed to combat this, which is to sign the hash of the CGA parameters before using it to construct the CGA, is not necessary to prove knowledge of the private key when claiming an address.

4.4. Improving router verification

The other major drawback to SEND identified in section 3.2 is the expensive process of authenticating routers, which can require significant amounts of processing for routers to assemble certificate chains, which they may need to do frequently, as well as for low-power devices to verify these chains. This creates the potential to make denial-of-service attacks in both directions by abusing the system to require much more processing than would be required by normal use. To prevent these attacks, host devices should place a limit on the lengths of certificate paths they will expect, thereby setting a maximum on the amount of processing they will have to do. However, the lengths of certificate paths may be difficult to predict, so this risks rejecting some legitimate paths. Routers can also minimise their processing by caching assembled certificate paths to reuse later, but this will only significantly affect processing costs from frequently-used trust anchors.

In the original specification of SEND, there was no particular system of trust anchors specified, only suggestions such as a centralised model under IANA or a decentralised model similar to what exists for SSL/TLS. In 2012 SEND was updated with RFC 6494 [24], which included specifying the use of the Resource Public Key Infrastructure (RPKI) certificate profile to ensure the RPKI can be used as a certificate path. RPKI has a well-defined hierarchy which makes it far easier to predict how long a valid certificate path should be, helping hosts to set an appropriate limit on accepted certificate path lengths.

Zhang et al. [27] propose an extension to SEND to streamline the process of verifying the authenticity of a router. When a node inquires about the certificates of a router, that node only needs to know that the router is indirectly authorised by a trust anchor known to the node and is not interested in the chain of trust in between. Zhang et al. therefore suggests mechanism by which the task of verifying the chain of trust instead falls on the router and other entities along the chain. The result of this is that the certificate path is condensed into a single token, called a Router Authorization Passport (RAP) by the proposal, which specifically authenticates the router's public key and is signed by the trust anchor. This in effect works as a single certificate provided directly by the trust anchor to the router, which can efficiently be verified by nodes on the network link.

This does not provide complete assurance that the router is legitimate, however. As noted in section 2.4, the inquiring node may not be reliably aware of the time, and so may not be certain that the RAP is still within its validity period. It is also possible that the certified public key will be compromised and the certificate will be revoked. If complete assurance is desired, the inquiring device will need a way to verify that the certificate or RAP is still valid. If the router has a RAP, it should include identifying information for each of the certificates involved, but only the last certificate (which vouches for the router's public key) needs to be verified, since the existence of the RAP guarantees that the other public keys in the chain had not been compromised when it was issued.

Fortunately, it is possible for a node to make some online queries before its gateway router is trusted, provided the responses are protected in such a way that any replay attacks or fabrication by an intermediate party can be detected, and provided that the node does not make itself vulnerable if no reply is received. These queries would require an IP address, which cannot reliably be chosen without an authenticated router from which to obtain the subnet prefix. A temporary address could be generated for this purpose, however, as using a false subnet prefix would not render the node vulnerable to attack. Alternatively, the router could be used as a proxy.

It would therefore be possible to make a query using the Online Certificate Status Protocol (OCSP) to receive a signed assurance that the certificate is still valid, using a nonce to prevent replay attacks. The OCSP response also contains a timestamp [28], allowing the node to verify that the certificate is within its validity period. This would require servers owned by each trust anchor to handle OCSP requests and provide RAPs. This means some infrastructure will need to be added to the Internet to support this scheme, and these servers would need to be able to support a high volume of requests.

A possible alternative to the use of certificates for this purpose would be the use of DNSSEC to make a reverse-lookup query into the router's IP address to learn whether the address has a public key registered with DNS. Like OCSP, DNSSEC responses are digitally signed [29], assuring the inquiring device that they cannot be tampered with by an untrustworthy router. This allows the node to securely and independently learn the router's public key if it is registered with DNS, or to obtain an authenticated denial of existence if it is not. In the latter case, this provides no assurance that the node's gateway router is legitimate, only that the network owner has not registered the public key. The node will therefore need to authenticate the public key using other means (such as certificates) or reject the network.

Both of these authentication methods rely on the validity of public keys which are stored by each device and trusted implicitly; those of the root trust anchors in the RPKI and the OCSP server in the case of certificate validation, and that of the DNS server in the case of DNSSEC authentication. However, over time these keys will expire or occasionally be compromised, and as a result authentication using the affected key will become either impossible or insecure for nodes which are unaware of the change.

A device could improve its security against the compromise of these keys by requiring routers to be authenticated by two independent trust anchors, whether through a

certificate chain or through DNSSEC, before accepting it as genuine. This would require more processing to authenticate routers, but it would mean that a false router would need two separate stolen keys to trick the device. The decision of whether or not to do this can reasonably be left to each host device implementation, as it does not affect the behaviour required of other nodes. Meanwhile, keys which expire peacefully and are replaced are less likely to cause problems if neither the host device nor the router relies entirely on a single trust anchor.

5. Proposal for a secure framework

The preceding sections examine a broad range of proposed systems for securing some or all of NDP's features, comparing them and analysing their advantages and disadvantages. However, apart from SEND itself and SSAS, these proposals are all made with a specific feature or vulnerability in NDP in mind, and none cover the full protocol, so it is necessary to use multiple proposals together for a complete solution.

It is therefore important to consider these proposals not only in isolation but also in terms of how they interact with each other, which ones work well together, and how best to combine them into a full framework which efficiently protects NDP as a whole. In this chapter, a suggestion for one such possible framework is described, along with the reasons for choosing each element used to create it.

The framework this paper proposes uses SEND as its foundation. SEND is already a very intricate and thorough extension to NDP and offers more protection than any of the other standalone solutions presented in this paper. It is also the only solution to have been implemented in industry, although its deployment is not widespread. Its main downside is the amount of processing and transmission overheads it incurs; many of the improvements adopted by this framework are therefore directed towards increasing the efficiency of SEND. There are also some notable vulnerabilities that remain in SEND, which are addressed by elements of this paper's proposal.

5.1. Address and signature generation

One of the most notable features of address generation in SEND is the hash extension technique, which allows the processing cost of generating a CGA, and consequently that of spoofing one, to be arbitrarily increased according to a parameter encoded in the address itself. The intention of this technique is to allow CGAs in their current state to remain secure even when it becomes practical to make brute-force attacks to spoof them, and thereby to extend the lifespan of the SEND protocol. However, as argued in section 4.1, SEND's reliance on the RSA and SHA-1 algorithms is likely to render it insecure long before the full potential of the hash extension is needed. Therefore one of the most important changes to make is to allow SEND to support the use of multiple algorithms, making the protocol more resilient to attacks against specific algorithms and allowing the selection of more efficient options to improve the protocol's performance.

Allowing SEND to support multiple signature algorithms is a simple change, as all that is needed is a way for devices to identify which algorithm is in use. This could be done by adding a new field to the CGA parameter structure, containing an identifier for the algorithm the public key belongs to. The specification of the CGA parameters includes extension fields reserved for future versions to add new parameters [9], so this will not cause incompatibility with previous implementations if RSA is still used.

Section 4.1 discusses three proposed algorithms to use as alternative signature algorithms to RSA. Of these, ECDSA carries the clearest advantages over RSA in its ability to use shorter keys, reducing the message expansion caused by SEND, and is the algorithm most widely regarded as a desirable alternative to RSA. It would be wise also to consider a quantum-resistant algorithm, as these are likely to become necessary in the near future. One such candidate is the Winternitz hash-based signature scheme discussed in section 4.1.3; however, at the time of writing, NIST is holding a contest to decide on standard quantum-resistant algorithms [30], and one or more of the algorithms standardised as a result of this contest should also be considered. Alternatively, the question of which algorithms to support can be left up to the implementation, but this could cause problems with interoperability if not enough algorithms are common to all implementations.

This framework will also include the implementation of MCGAs as discussed in section 4.2, allowing nodes to specify additional public keys beyond the first when creating a CGA and thereby creating a way in which proxies can be established in SEND. The version of MCGAs supported by this proposal is that described by Cheneau and Laurent [14], in which multiple independent key pairs are associated with the address and any one of them can be used to sign a message. The signature option in SEND already includes a truncated hash of the verification key [6] which can be used to identify which of an MCGA's keys is required for a given signature.

As noted in section 4.2, this allows not only proxies but also the use of multiple signature algorithms by a single node, which may become necessary to ensure compatibility between all nodes on a network if a single algorithm cannot reliably be understood by all implementations. If an MCGA is used for this purpose and one of the keys is an RSA key, that key should be specified as the first key in the CGA parameters to ensure backward compatibility with versions of SEND which do not support multiple algorithms, as such versions would only know to check the first key and would expect an RSA key.

Supporting multiple hash algorithms for the generation of the CGA is somewhat more complicated than supporting multiple signature algorithms, since it is necessary to encode the algorithm used in such a way that an attacker cannot use a different, weaker algorithm to find a collision. Section 4.1.4 describes a way of doing this, proposed in RFC 4982 [13], which works by repurposing the security parameter in the CGA. This proposal does not affect the hash extension technique but alters the meaning of the security parameter so that it refers to a combination of a hash algorithm and a *sec* value. For example, a security parameter of 000 may refer to using SHA-1 with a *sec* value of 0 while a parameter reading 010 refers to using SHA-256 with a *sec* value of 0.

As discussed in section 4.1.4, this method comes with very few drawbacks. It does mean that at some point it will be necessary to recycle parameter values in order to add new algorithms or higher *sec* values. However, if the specified algorithms and *sec* values are chosen well, there should usually be an insecure configuration available to be replaced when a new one must be specified. For example, if 000, 001 and 010 denote SHA-1 with *sec* values of 0, 1 and 2 respectively and it becomes practical to add a configuration using SHA-1 with a *sec* value of 3, SHA-1 with *sec* 0 should will have become insecure and can be replaced with the new configuration.

Whenever possible, a new configuration should replace an older one which uses the same algorithm with a lower *sec* value for reasons of compatibility. Because of the way the hash extension technique is specified, a set of parameters which passes verification with a high *sec* value will also pass verification with a lower *sec* value, so when a configuration is replaced in this way any nodes which have not been updated will still be able to verify updated nodes using the new configuration. This means that nodes using the new configuration will still be considered secure by outdated nodes. Nodes using the old configuration will fail verification by updated nodes and will be treated as unsecured, but this too is desirable, as the old configuration should be considered insecure.

Some incompatibility is unavoidable when specifying a new hash algorithm to replace one which has become insecure, but when this occurs there will still be other algorithms specified which remain secure and are understood by both the old and the new standard, so secure communication will not be impossible. For this reason, the newly-specified algorithm should not be used until it can confidently be assumed that the majority of devices on the network will be using the new standard.

Another proposal which ties in closely with this, as it pertains to the process of generating address, is the CGA++ proposal described in section 4.3. The global time-memory trade-off attack discussed in that section is a very real possibility and will only become more practical and more damaging as storage costs decrease and SEND begins to use higher *sec* values. It therefore makes sense to take steps against it as early as possible, and while CGA++ does not eliminate the attack, it does limit its applicability to the point where it must be carried out separately against each network.

A notable downside is that including the subnet prefix in the hash extension test will make new implementations incompatible with old ones except when a *sec* value of 0 is used. Here again, however, the security parameter can be used to specify whether to use CGA++ for each configuration independently. It can therefore be specified that configurations using SHA-1, for which backward compatibility is a concern, will not use CGA++ while configurations which use a different algorithm, which are already incompatible, will. Table 1 proposes a possible full specification for the repurposed security parameter, which follows on from the reasoning given here and in section 4.1.4. However, this is only an example, and a more detailed analysis of the options and requirements should be done before anything is standardised.

Note that CGA++ here refers only to the proposal that the subnet prefix be included in the hash extension process. As discussed in section 4.3, the other part of the CGA++ proposal, that the CGA parameters be signed before being used to generate the CGA, is not needed for the purpose suggested by Bos et al. [26]. The proposal might provide protection to nodes requesting that a CGA be generated for them by a more powerful device, as it adds an extra processing step between hash extension and the final address which attackers cannot calculate both ways, but it would also be cumbersome for MCGAs as all key owners would have to actively participate in generating the address and all of the related signature algorithms would be needed to verify the CGA. For this reason, this part of the CGA++ proposal is excluded from the proposed framework.

Parameter	Algorithm	*sec*	Use CGA++?
000	SHA-1	0	N/A
001	SHA-1	1	No
010	SHA-1	2	No
011	SHA-256	0	N/A
100	SHA-256	1	Yes
101	SHA-3	1	Yes
110	SHA-3	2	Yes
111	SHA-3	3	Yes

Table 1. A suggested specification for a repurposed security parameter.

5.2. Router authentication

The other main feature of SEND which requires improvements is the process of authenticating routers to ensure they are not malicious devices impersonating the router to set up a man-in-the-middle attack. This was ie focus of section 4.4, and the proposals made here follow on from the discussion in that section.

This framework makes use of two main methods of establishing the authenticity of a router. The first of these is certificate paths, as already used in SEND, whereby each router is certified to act as a router for a set of subnet prefixes, and the certifying entity is in turn certified, leading to a chain of trust which should ultimately lead back to a trust anchor which the inquiring device implicitly trusts. The second method is a reverse lookup using DNSSEC, which can be used to establish whether the network has a public key registered in the DNS database, and whether the registered public key is the same as the key used by the advertising router.

These two methods differ in how they must be used, but they offer similar levels of assurance, with the DNS server functioning as a trust anchor vouching directly for the network's registered public key(s). This direct relationship makes the option of querying the DNS server simpler to use, but certificate paths have the benefit of being able to use multiple trust anchors, which is important to avoid over-relying on the security and continued validity of a single public key.

To reduce the processing costs of using certificate paths to authenticate a router, this framework supports the proposal by Zhang et al. [27] that routers acquire condensed tokens (RAPs) which confirm the existence of valid certificate paths leading back to frequently-used trust anchors and are signed by those trust anchors directly. This can only be done if the trust anchors implement a service which provides RAPs; this should include verifying the certificate path including checking the status of all of the certificates to ensure none have been revoked. When a router receives a CPS message indicating that the sender can process RAPs, relevant RAPs should be given precedence over full certificate paths in constructing the reply to minimise processing requirements. Only if not enough RAPs are available to satisfy the request should a certificate path be assembled.

A node attempting to authenticate a router will also need to verify that the certificates used to do so are still valid. This can be achieved by making an OCSP query to the relevant server, but since the node will not yet have an IP address or a trusted Internet connection at this point, it will be necessary to use the router as a proxy to make this request, and so routers will need to support this functionality. Although the router is not trusted, the OCSP response is authenticated, so the only ways the router can manipulate the exchange are by making a replay attack or by suppressing the response. The query should therefore use a nonce to prevent replays, and if no response is received after multiple attempts verification should fail. Note, however, that the node must have the public key of the OCSP server in order to verify that the response is authentic. Trust anchors will therefore need to maintain OCSP databases which track the certificates they issue RAPs for.

This verification should be done for every certificate in a certificate path; however, if the node is verifying a RAP, only the final certificate needs to be verified using OCSP. This is because the existence of the RAP guarantees that all of the certificates were valid when the RAP was generated, so it can be assumed that the public keys were secure when they were used to certify the router. There is a chance that a key had been compromised but not yet revoked when the RAP was generated; this risk can be mitigated if the RAP service forces a delay between the request and the provision of the RAP.

The process of authenticating a router using DNSSEC is very similar to the process of verifying a certificate using OCSP, as it involves sending a reverse-lookup query using the router as a proxy and receiving an authenticated reply from the DNS server. The response to this query is all that is needed to authenticate the router using this method, but if the network has not registered public keys with DNS or if the node requires two independent verifications to trust the router it will be necessary to use certificates.

The requirements for a router to be trusted may vary slightly between implementations, but to mitigate the risk that a trust anchor's private key may be compromised, nodes should require validation by at least two independent trust anchors unless it is very important to be able to authenticate a router quickly. Validation by a specific trust anchor should also not be required, as this will become impossible if the public key held for that trust anchor expires. Finally, devices should place a limit on the lengths of certificate chains they will accept, and routers should limit the number of certificate paths that can be requested from them by a single node, to prevent denial of service attacks taking advantage of this feature.

These changes will improve the security of router authentication in SEND by taking account of the possibility that a signing key will be compromised and its certificate revoked, and the proposals to use RAPs and DNSSEC reverse-lookup queries as authentication tools have the potential to increase the efficiency of this SEND feature greatly. However, it is worth noting that a substantial amount of infrastructure will be needed to support this. Each trust anchor will need an OCSP server which can verify the status of any certificates linked to the trust anchor, and most should also implement RAP provision services for routers with a valid certificate path. Using DNS reverse lookup as an authentication method will also introduce a significant amount of load to the DNS servers.

5.3. DHCPv6

For the most part, this paper has been assuming a decentralised network using SLAAC, as this is a new capability in IPv6 and a challenging one to secure, whereas the challenges of centralised networks are comparatively well-understood and DHCPv6 already includes security measures such as an authentication framework using digital signatures [31]. SLAAC is also assumed to be the default setting in IPv6 [6]. However, while DHCPv6 does authenticate its own messages, it seems to specify no mechanism for proving or verifying ownership of the addresses assigned by a DHCP server.

This could be done quite easily by appending a signature to each message which can be verified using a public key sent to the DHCP server with the initial address request. This requires that DHCP servers allow other nodes to request a copy of the public key associated with an IP address. This mirrors the functionality provided by the controller node in SSAS, discussed in section 3.3.1.

However, SEND requires nodes to use CGAs in order to be considered secure by other nodes [6]. This requirement could be relaxed when there is a DHCP server on the network which has been advertised by a certified router; alternatively, the DHCP could construct a CGA for each host on the network, using a parameter set supplied by the device making the request, as described in section 4.3. An advantage of this would be that nodes would not have to request public keys from the DHCP server, as the CGA would verify the key just as it would in a decentralised network. However, the DHCP server would still be able to maintain a central table of addresses in use on the network, along with associated public keys if desired, and this would remove the need for duplicate address detection.

5.4. SEND and IoT

One of the most important aims of this paper is to ensure that the security mechanisms chosen to protect NDP are suitable and practical for use by as many Internet-connected devices as possible, including mobile, low-power and resource-constrained devices. However, it should be clarified that this proposal is not intended to be used to secure Internet-of-Things (IoT) networks, as SEND is incompatible with 6LoWPAN [32]. The reason for this is that numerous changes are made to the way NDP operates in 6LoWPAN networks in order to optimise the protocol for low-power and lossy communications and for devices with regular sleep cycles [33] which neither SEND nor the improvements proposed in this paper take into account.

Since NDP is an entirely link-local protocol, the devices running NDP over 6LoWPAN will not be communicating directly with non-6LoWPAN devices using NDP, so these devices can be secured using a different protocol extension without needing to be concerned about compatibility. A draft proposal exists for a lightweight version of SEND for 6LoWPAN networks [34], although this has not been updated for several years. Such proposals may benefit from some of the ideas described in this paper, particularly algorithm agility and router authentication methods. However, a detailed investigation into NDP security for 6LoWPAN is beyond the scope of this paper.

6. Conclusion

This paper has examined a wide range of proposed systems for protecting NDP from abuse, from the currently implemented standard, SEND, to the various suggestions to improve the efficiency and robustness of SEND, to a selection of proposals intended to replace SEND altogether. The majority of these proposals address only a part of the problem of securing NDP, and so this paper takes a broader view and suggests a way in which aspects of several of the considered proposals could be combined to form a complete solution for safeguarding the protocol, considering the ways in which they work together and complement each other. Using SEND as a foundation, the proposed solution focuses on improving the efficiency and accessibility of SEND, as well as making SEND more adaptable to new attacks which will be discovered over time.

The solution proposed in this paper makes use of previous proposals to allow SEND devices a choice between multiple algorithms both for signing messages and for computing the hashes used in address generation. This not only allows SEND the flexibility to last beyond the deprecation of the algorithms currently in use, but also provides a way of improving its efficiency by using faster algorithms such as ECDSA, all while maintaining backward compatibility with current versions by continuing to support RSA and SHA-1. A proposal for including multiple public keys in CGAs is included, which helps to support both proxies and signature algorithm agility, and a small change to the process of generating CGAs is included to protect against a time-memory trade-off attack, provided using the algorithm agility mechanisms to ensure it does not break backward compatibility.

Router authentication is also made more efficient and secure by including a proposal which allows certificate paths to be condensed into a single token signed by the trust anchor, and by formulating two new proposals, one which takes into account the possibility of certificate revocation and recommends making an OCSP query using the router as a proxy, and one which suggests the DNS as an additional 'trust anchor' which can be contacted directly through a DNSSEC reverse-lookup query, bypassing the cumbersome certificate path mechanism entirely. However, the feasibility of these new proposals has not been extensively investigated.

The solution proposed in this paper requires several changes to the way in which SEND operates, including altering the meaning of the security parameter in CGAs, specifying new extensions to the CGA parameters, and requiring routers to act as proxies for OCSP and DNSSEC queries. Standards action would therefore be required before these changes could be implemented, during which the feasibility of this proposal will be subjected to more thorough investigation and the framework will likely be changed as a result. However, it is hoped that by providing a broad perspective on how these proposals can be used to improve the security of NDP as a whole, this paper can help to create a more robust and efficient solution than could be achieved by considering each proposal in isolation.

7. Bibliography

[1] C. E. Caicedo, J. B. D. Joshi and S. R. Tuladhar, "*IPv6 Security Challenges*", *Computer*, vol. 42, no. 2, February 2009, pp. 36–42.

[2] S. Sotillo, "*IPv6 Security Issues*", *Scanning*, 2006.

[3] T. Zhang and Z. Wang, "*Research on IPv6 Neighbor Discovery Protocol (NDP) Security*", 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China (2016), pp. 2032–2035.

[4] 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, 2017, pp. 18187–18210.

[5] D. J. Tian, K. R. B. Butler, J. I. Choi and P. McDaniel, "*Securing ARP/NDP From the Ground Up*", *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, September 2017, pp. 2131–2143.

[6] Internet Engineering Task Force RFC 3971 (2005): *SEcure Neighbor Discovery (SEND)*, *http://www.ietf.org*

[7] F. Xiaorong, L. Jun and J. Shizhun, "*Security analysis for IPv6 neighbor discovery protocol*", 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA), Toronto, Canada (2013), pp. 303–307.

[8] A. AlSa'deh and C. Meinel, "*Secure neighbor discovery: Review, challenges, perspectives, and recommendations*", *IEEE Security & Privacy*, vol. 10, no. 4, July-August 2012, pp. 26–34.

[9] Internet Engineering Task Force RFC 3972 (2005): *Cryptographically Generated Addresses (CGA)*, *http://www.ietf.org*

[10] Internet Engineering Task Force RFC 3513 (2003): *Internet Protocol Version 6 (IPv6) Addressing Architecture*, *http://www.ietf.org*

[11] 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, June 2010, pp. 419–431.

[12] S. Qadir and M. U. Siddiqi, "*Cryptographically generated addresses (CGAs): A survey and an analysis of performance for use in mobile environment*", *International Journal of Computer Science and Network Security*, vol. 11, no. 2, 2011, pp. 24–31.

[13] Internet Engineering Task Force RFC 4982 (2007): *Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)*, *http://www.ietf.org*

[14] T. Cheneau and M. Laurent, "*Using SEND signature algorithm agility and multiple-key CGA to secure proxy neighbor discovery and anycast addressing*", Conference on Network and Information Systems Security (SAR-SSI), La Rochelle, France (2011).

[15] S. Praptodiyono, R. K. Murugesan, A. Osman and S. Ramadass, "*Security Mechanism for IPv6 Router Discovery based on Distributed Trust Management*", IEEE International Conference on RFID Technologies and Applications, Johor Bahru, Malaysia (2013).

[16] S. Praptodiyono, R. K. Murugesan, I. H. Hasbullah, C. Y. Wey, M. M. Kadhum and A. Osman, "*Security Mechanism for IPv6 Stateless Address Autoconfiguration*", International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT), Bandung, Indonesia (2015), pp. 31–36.

[17] H. Rafiee and C. Meinel, "*SSAS: A simple secure addressing scheme for IPv6 autoconfiguration*", Eleventh Annual International Conference on Privacy, Security and Trust (PST), Tarragona, Spain (2013), pp. 275–282.

[18] Internet Engineering Task Force RFC 7039 (2013): *Source address validation improvement (SAVI) framework*, *http://www.ietf.org*

[19] C. Castelluccia, "*Cryptographically generated addresses for constrained devices**", *Wireless Personal Communications*, vol. 29, no. 3–4, June 2004, pp. 221–232.

[20] S. Qadir, M. U. Siddiqi and F. Anwar, "*A study of CGA- (cryptographically generated address) signature based authentication of binding update messages in low-end MIPv6 node*", International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia (2012), pp. 510–514.

[21] S. Qadir, M. U. Siddiqi and W. F. M. Al-Khateeb, "*An investigation of the Merkle signature scheme for cryptographically generated address signatures in mobile IPv6*", *International Journal of Network Security*, vol. 17, no. 3, May 2015, pp. 311–321.

[22] Internet Engineering Task Force RFC 5909 (2010): *Securing neighbor discovery proxy: Problem statement*, *http://www.ietf.org*

[23] Internet Engineering Task Force RFC 6496 (2012): *Secure Proxy ND Support for SEcure Neighbor Discovery (SEND)*, *http://www.ietf.org*

[24] Internet Engineering Task Force RFC 6494 (2012): *Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)*, *http://www.ietf.org*

[25] J. Kempf, J. Wood, Z. Ramzan and C. Gentry, "*IP Address Authorization for Secure Address Proxying Using Multi-key CGAs and Ring Signatures*", 2006. In: H. Yoshiura, K. Sakurai, K. Rannenberg, Y. Murayama and S. Kawamura (eds) *Advances in Information and Computer Security,* IWSEC 2006, *Lecture Notes in Computer Science*, vol. 4266, Springer, Berlin, Heidelberg.

[26] J. W. Bos, O. Özen and JP. Hubaux, "*Analysis and optimization of cryptographically generated addresses*", 2009. In: P. Samarati, M. Yung, Ardagna C. A. (eds) *Information Security*, ISC 2009, *Lecture Notes in Computer Science*, vol. 5735, Springer, Berlin, Heidelberg.

[27] J. Zhang, J. Liu, Z. Xu, J. Li and X. Ye, "*TRDP: A trusted router discovery protocol*", International Symposium on Communications and Information Technologies, Sydney, Australia (2007), pp. 660–665.

[28] Internet Engineering Task Force RFC 6960 (2013): *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, *http://www.ietf.org*

[29] Internet Engineering Task Force RFC 4033 (2005): *DNS Security Introduction and Requirements*, *http://www.ietf.org*

[30] National Institute of Standards and Technology, *Post-Quantum Cryptography Round 1 Submissions*, 2018, *https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions*

[31] Internet Engineering Task Force RFC 3315 (2003): *Dynamic host configuration protocol for IPv6 (DHCPv6)*, *http://www.ietf.org*

[32] A. Rghioui, M. Bouhorma and A. Benslimane, "*Analytical study of security aspects in 6LoWPAN networks*", 5th International Conference on Information and Communication Technology for the Muslim World (ICT4M), Rabat, Morocco (2013).

[33] Internet Engineering Task Force RFC 6775 (2012): *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, *http://www.ietf.org*

[34] Internet Engineering Task Force Draft (2012): *Lightweight Secure Neighbor Discovery for Low-power and Lossy Networks*, *https://tools.ietf.org/html/draft-sarikaya-6lowpan-cgand-03*