CGA & SEND maintenance
Internet-Draft
Updates: RFC3971
(if approved)
Expires: December 18, 2010

T. Cheneau
M. Laurent
TMSP
S. Shen
Huawei
M. Vanderveen
Qualcomm

June 16, 2010

Signature Algorithm Agility in the Secure Neighbor Discovery (SEND) Protocol
draft-cheneau-csi-send-sig-agility-02

Abstract

This document describes a mechanism to enable the Secure Neighbor Discovery (SEND) protocol to select between different signature algorithms to use with Cryptographically Generated Addresses (CGA).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on December 18, 2010.

Copyright Notice

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The usage scenarios associated with neighbor discovery have recently been extended to include environments with mobile or nomadic nodes. Many of these nodes have limited battery power and computing resources. Therefore, heavy public key signing algorithms like RSA are not feasible to support on such constrained nodes. Fortunately, more lightweight yet secure signing algorithms do exist and have been standardized, e.g. Elliptic Curve based algorithms.

It is then a worthwhile goal to extend secure neighbor discovery to support signing and corresponding hashing algorithm agility. Besides accommodating power-constrained nodes, signing and hashing algorithm agility is also desired as a safety measure over time, to offer alternatives when cryptanalysis of one type of algorithm makes significant progress. Moreover, in some countries or companies, the use of signing and hashing algorithms might be constrained.

The aim of this memo is to outline options for allowing public key signing algorithm and hashing algorithm agility for nodes configured to perform secure neighbor discovery operations. The extent to which these options impact existing specifications [RFC3971] and [RFC3972] is also addressed.

It should be noted that this memo concerns the usage of the signing and hashing algorithms in SEND and therefore solves a different problem than the CGA extension proposed in [RFC4982].

2. Overview

The current SEND protocol specification, [RFC3971], mandates the use of the RSA signature algorithm. Since the time of its writing, different signature algorithms have been shown to be secure and have been adopted by other protocols in an effort to reduce key length, signature generation and verification time, and increase security level. This shift in signature algorithm adoption particularly benefits lightweight devices, which are power and memory-limited but in need of secure signing algorithms support. For these reasons, we feel that the restriction on the signature algorithm for SEND is no longer warranted.

2.1. Potential solutions for Signature Algorithm Agility in SEND

The following section highlights the possible behaviors and capabilities of Signature Algorithm agility of SEND-enabled nodes. Note that this section does not state requirements for this specification, but merely discusses possible solutions.

When implementing a Signature Algorithm Agility solution, a node can possess the following capabilities (ranging from most restrictive to most permissive):

1. A node can verify and sign a message with only one Signature Algorithm that corresponds to the Public Key used to form its CGA address or a known Public Key (e.g. stored in a router's certificate). This is already the case of the nodes that implement [RFC3971] and rely on the RSA Signature Algorithm. In

a near future, it is expected that new nodes will build their CGA addresses using other Signature Algorithms. The CGA address generation and verification process (described in [RFC3972]) is already Signature Algorithm agnostic. Only minor updates are required to the RSA Signature Option. SEND is updated so that if a node receives an unverifiable message (i.e. a message signed with a Signature Algorithm different from the one its Public Key is linked to), the message should be threated as an "unsecure" message (as defined in Section 8 of [RFC3971]).

2. A node can have the above capabilities plus the ability to verify more than one Signature Algorithm. This allows a better interworking of nodes using different Signature Algorithms. In this case, such a node is able to sign SEND messages as above but is also able to verify SEND messages sent by nodes of capabilities as case 1 but not necessarily sharing the same signature algorithm. A fall-back on traditional ND is still an option.

3. A node can have all the above capabilities, plus the ability to sign with multiple Signature Algorithms. This can be achieved in two ways:

    1. The first solution is that the node generates multiple CGA addresses. Each CGA address uses a Public Key linked to a different Signature Algorithm. This solution has two drawbacks: the node is viewed as multiple entities, and the node needs to make a decision on using one address over another when starting a communication (i.e. possibly a negotiation mechanism). While this is an issue for hosts, routers may actually benefit from such a solution, especially in heterogeneous networks. A router may choose to send multiple Router Advertisement from its multiple CGA addresses.

    2. A second approach is that the node builds a CGA from multiple Public Keys. In practice, this can be achieved using CGA Parameter Data Structure extensions to store the extra Public Keys. This solution, while allowing multiple Signature Algorithms at once and thus enhancing interoperability, has several drawbacks: it requires a negotiation algorithm (and

thus is prone to bidding downs attacks), and it increases the size of the packets.

2.2. Agility Requirements

Based on the above discussion, in this specification, we set forth the following requirements:

- A node MUST be able to sign and verify with the Signature Algorithm corresponding to its Public Key. This Public Key can be bound to a CGA address or contained in a certificate (e.g. the node is in fact a router).

- For increased interoperability, a node SHOULD be able to verify more than one type of Signature Algorithm.

- Nodes acting as routers MAY use more than one CGA address. The main purpose is to maximize the number of nodes that can communicate securely with a router within an heterogeneous network. However, this functionality comes with a limitation: each address is seen as a different entity on the network.

For interoperability purposes, we also define the following additional requirements on a signing algorithm agility solution for SEND:

- A Signature-Algorithm-Agility-Node SHOULD be able to communicate with a Non-Signature-Algorithm-Agility-Node. Traditional Neighbor Discovery (ND) should suffice, to accommodate nodes that only support one type of Signature Algorithm, which may not be RSA. Local policy MAY disable this behavior, namely the use of unsecured ND messages when communicating with a node that does not share any common signature algorithm.

- Two Signature-Algorithm-Agility nodes that support signing with a common Signature Algorithm and hashing algorithm SHOULD be able to communicate using SEND and sign messages using the common Signature Algorithm and hash algorithm.

While not a requirement per se, it is also desirable that the current SEND/CGA specifications should incur as few changes as possible.

2.3. Mechanism for Agility Support of CGA and SEND

This document proposes an update to [RFC3971] to allow two SEND nodes to choose an appropriate signature algorithm. This solution encompasses the following:

- A "Supported Signature Algorithm" Neighbor Discovery Protocol option which contains a list of signing and hashing algorithms that the sender node supports for SEND purposes and its interaction with the Neighbor Cache;
- A modification of the "RSA Signature" option defined in the SEND specification;

We define the aforementioned options format and provide processing rules for both senders and receivers of SEND messages employing the new options, as well as example message exchange flows.

Note that the ECC support for SEND is described in document [cheneau-csi-ecc-sig-agility].

3. Supported Signature Algorithm Option

The Supported Signature Algorithm NDP option contains a list of signing and hashing algorithm pairs that the sender node supports. The purpose of the Supported Signature Algorithm option is to diagnose sender's Signature Algorithm abilities. The format of this option is described in Figure 1:

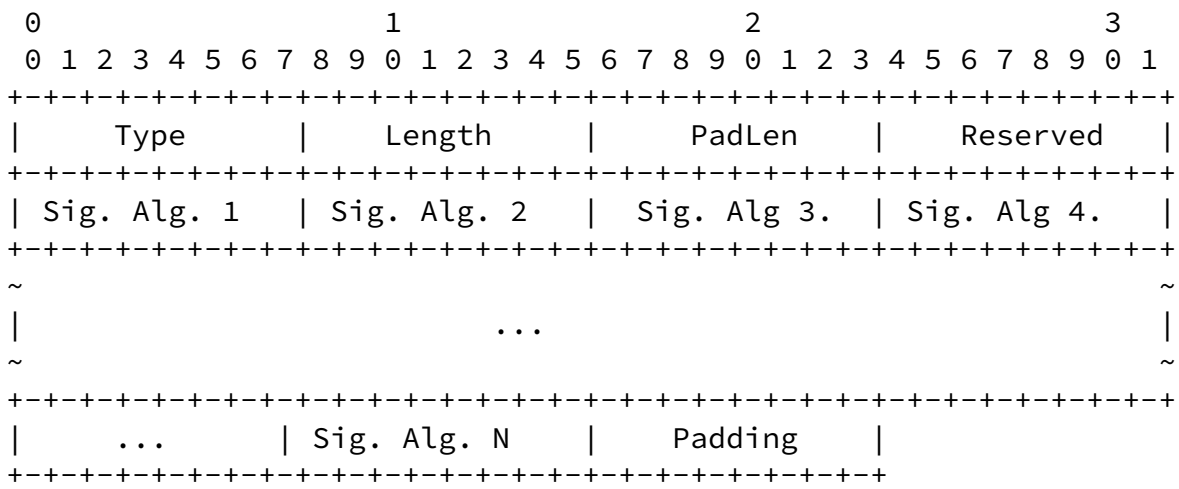


Figure 1: Supported Signature Algorithm option

Type

NDP option type, TBA. See Section 8.

Length

The length of the option (including the Type, Length fields), in units of 8 octets. An 8-bit unsigned integer. Value 0 is invalid.

PadLen

The number of padding octets beyond the end of the Signature Algorithm field(s) but within the length specified by the Length field. This field is 8-bit long.

Reserved

Reserved for future use. This 8-bit field MUST be set to zero by the sender, and MUST be ignored by the receiver.

Signature Algorithm

A one-octet long field indicating a signature algorithm and the corresponding hash algorithm that this node supports; this support implies at least ability to verify signatures of this signature algorithm.

If the first leftmost bit, bit 0, is set to 0, it indicates that the sender is able to perform signature checks only (i.e. no signature generation with this type of signature algorithm). If this bit is set to 1, it indicates that the sender has a public key of this type and can generate signatures (e.g. node's CGA address is generated from this public key). Bit 1 and 2 are reserved. Bit 3 to 7 are named Signature Type Identifier subfield and encode an identifier for the signature algorithm and corresponding hash algorithm. Default values for the Signature Type Identifier subfield defined in this document are taken in part from the IANA-defined numbers for the IKEv2 protocol, i.e. IANA registry named "IKEv2 Authentication Method":

* Value 0 is RSA/SHA-1 (compatible with [RFC3971])

* Value 1 is RSA/SHA-256

* Section 5 of document [cheneau-csi-ecc-sig-agility] provides values for ECDSA signature algorithm

The Signature/Hash Algorithm combinations SHOULD be included in order of preference.

A Supported Signature Algorithm option MAY be built to respect a Local Policy. However, the Supported Signature Algorithm option MUST not indicate Signature Algorithm(s) that the emitting node's CGA does not support and MUST contain at least one Signature Algorithm with the first bit on (i.e. this Signature Algorithm is available for signature generation).

Padding

A variable-length field making the option length multiple of 8, containing as many octets as specified in the PadLength field. Padding octets MUST be set to zero by the senders and ignored by receivers.

3.1. Neighbor Cache interactions

The Neighbor Cache SHOULD have the ability to store Supported Signature Algorithm information for each entry (i.e. IPv6 address). Supported Signature Algorithm information for an entry MAY be empty (e.g. entry created by a RFC 3971 node or due to the receipt of an unverifiable SEND message).

3.2. Processing Rules for Senders

If a node has been configured to use SEND, then all Neighbor Solicitation, Neighbor Advertisement, Router Solicitation, Router Advertisement, and Redirect messages it sends MUST contain the Supported Signature Algorithm option. This option MUST contain in the Signature Algorithm field(s) all the signature algorithms it is willing to use in signature generation and verification.

3.3. Processing Rules for Receivers

Upon receiving a SEND packet with a Supported Signature Algorithm option, a receiver performs the following operations:

- When a message is a Neighbor Solicitation or a Router Solicitation, the receiving node computes the intersection between the set of Supported Signature Algorithm indicated by the option and its own. If the set is empty, this means the node will not be able to use a Signature Algorithm that the initiating node can check. Given the local policy, a receiver node MAY still respond to the received message using its "preferred" Signature Algorithm (even if the node knows the receiver will not be able to verify the Signature Algorithm). In any case, a system warning message SHOULD be logged. If the set is not empty, the receiving node will choose one of the algorithms among the set in order to generate an Universal Signature option (see Section Section 4.1).

- If a message passes the SEND verifications (CGA verification, Timestamp, Nonce, RSA Signature Option or Universal Signature Option verification) and contains a Supported Signature Algorithm option, the information of the Supported Signature Algorithm in the Neighbor Cache is updated by the information contained in the Supported Signature Option attached to the message.

- If a message does not pass the SEND verifications because of a unverifiable RSA Signature Option or Universal Signature Option, if it contains a Supported Signature Algorithm option, and the Neighbor Cache entry associated to that node does not contain any

information about the Supported Signature Algorithm, the Neighbor Cache entry SHOULD be updated with the information contained in

the Supported Signature Algorithm option.

4. SEND Universal Signature Option

We propose replacing the RSA Signature Option by a new algorithm-independent signature option. The "Universal Signature" Option is an updated version of the RSA Signature Option, that allows a node to specify which Signature Algorithm it is using. To achieve this, we use the 16-bit reserved field of the RSA Signature Option, and define a new 5-bit Signature Type Identifier field that details the type of algorithms used to generate the Digital Signature.

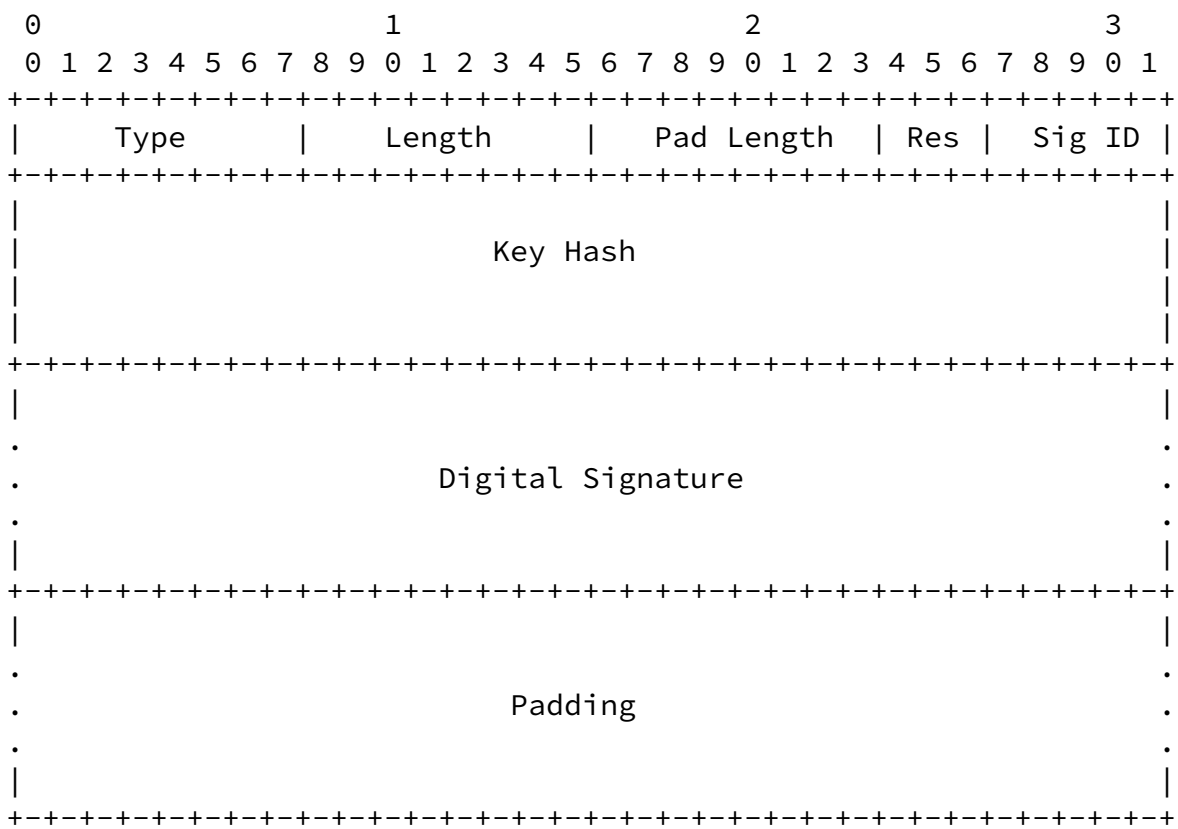


Figure 2: Universal Signature Option format

Type

Same value as in [RFC3971]: 12.

Length

The length of the option (including the Type, Length, Reserved, Signature Type Identifier, Key Hash, Digital Signature, and Padding fields) in units of 8 octets.

Pad Length

An 8-bit integer field, giving the length of the Pad field in units of an octet. For backward compatibility with [RFC3971], if the value of the Signature Type Identifier is 0 (i.e. signing algorithm is RSA and hash function is SHA-1), this field MUST be set to zero by sender and MUST be ignored by the receiver.

Reserved

A 3-bit field reserved for future use. The value MUST be set to zero by the sender and MUST be ignored by the receiver.

Signature Type Identifier

Signature Type Identifier is a 5-bit field. It corresponds to the Signature Type Identifier subfield (bits 3 to 7 of the Signature Algorithm field) in the Supported Signature Algorithm option . It indicates the type of signature contained in the Digital Signature field.

Key Hash

A 128-bit field containing the most significant (leftmost) 128 bits of a hash of the public key used for constructing the

signature. It is computed using the same hash function as used in generating digital signature (indicated in Signature Type Identifier). The hash value is computed over the presentation used in the Public Key field of the CGA Parameters data structure carried in the CGA option. Its purpose is to associate the signature with a particular key known by the receiver. Such a key can either be stored in the certificate cache of the receiver or be received in the CGA option in the same message.

Digital Signature

A variable-length field containing a signature constructed by using the sender's private key associated to the public key. The signature type is determined from the value of the Signature Type Identifier field.

* If the value of the Signature Type Identifier field is 0, the Digital Signature field is computed the same way as the Digital Signature field of the RSA Signature Option described in [RFC3971]. This value is compatible with [RFC3971].

* If the value of the Signature Type Identifier field is 1, then this Digital Signature field is computed the same way as the

Digital Signature field of the RSA Signature Option described in [RFC3971] except that the signature is computed with the RSASSA-PKCS1-v1_5 algorithm (as defined in [PKCS1]) and the SHA-256 hash function.

* Values for ECDSA signature algorithm are defined in Section 5 of [cheneau-csi-ecc-sig-agility].

This field starts after the Key Hash field. The length of the Digital Signature field is determined by the length of the Universal Signature option minus the length of the other fields (including the variable length Pad field).

Padding This variable-length field contains padding, as many bytes long as remain after the end of the signature.

A Neighbor Solicitation/Advertisement, Router Solicitation/ Advertisement and Redirect message MAY contain more than one

Universal Signature option (e.g. using varying hash functions), as long as it does not exceed the MTU. The Universal Signature Options(s) is (are) added as the last option(s). Adding multiple Universal Signature Options is particularly useful for routers operating in heterogeneous networks, where hosts have a disjoint set of supported signature algorithms. For information on how to compute the message size, see Appendix A.

4.1. Processing Rules for Senders

When sending a SEND message spontaneously, a sender node SHOULD choose a signature algorithm of its preference (defined by its local policy) among the corresponding Signature Algorithm available for the Public Key carried in the CGA option. Using this signature algorithm, the node computes the Digital Signature and fills the Signature Type Identifier field with appropriate value.

If the node has been configured to use SEND, then all Neighbor Solicitation, Neighbor Advertisement, Router Advertisement, and Redirect messages MUST contain at least one Universal Signature option. Router Solicitation messages not sent with the unspecified source address MUST contain the Universal Signature option.

A node sending a message with one or more Universal Signature option(s) MUST construct the message as follows:

- If the node has previously received hints (e.g. a NDP message with a Supported Signature Algorithm option or an entry in the Neighbor Cache) on the type of Signature Algorithm it should use, it SHOULD restrict its choice on those Signature Algorithms.

- The message is then constructed in its entirety, without any of the Universal Signature options.
- The Universal Signature option(s) is (are) added as the last option in the message.
- The data to be signed is constructed as explained in [RFC3971], under the description of the Digital Signature field.
- The message, in the form defined above, is signed by using the configured private key associated to the selected Signature

Algorithm, and the result signature is encapsulated into the Digital Signature field.

4.2. Processing Rules for Receivers

Neighbor Solicitation, Neighbor Advertisement, Router Advertisement, and Redirect messages without any Universal/RSA Signature option, or with an unverifiable Universal/RSA Signature option MUST be treated as unsecured (i.e., processed in the same way as NDP messages sent by a non-SEND node). See Section 8 of [RFC3971].

Router Solicitation messages without any Universal/RSA Signature option MUST also be treated as unsecured, unless the source address of the message is the unspecified address.

Redirect, Neighbor Solicitation, Neighbor Advertisement, Router Solicitation, and Router Advertisement messages containing one or more Universal Signature option MUST be checked as follows:

- The receiver MUST ignore any options that come after the last Universal Signature option. (The options are ignored for both signature verification and NDP processing purposes.)

- If the Signature Type Identifier field of the first Universal Signature Option indicates a Signature Algorithm the node is unable to process, the following checks MUST be run on the next Universal Signature Option with a Signature Type Identifier field indicating a verifiable Signature Algorithm (if any).

- The Key Hash field MUST correspond to a known public key, either one learned from the CGA option in the same message or one known by other means.

- The Digital Signature field MUST have correct encoding and MUST not exceed the length of the Universal Signature option minus the Padding.

- The Digital Signature verification MUST show that the signature has been calculated as specified in the previous section.

- If the use of a trust anchor has been configured, a valid

certification path (see Section 6.3 of [RFC3971]) between the receiver's trust anchor and the sender's public key MUST be known.

Messages that do not pass all the above tests MUST be silently discarded if the host has been configured to accept only secured ND messages. The messages MAY be accepted if the host has been configured to accept both secured and unsecured messages but MUST be treated as unsecured messages. The receiver MAY also otherwise silently discard packets (e.g., as a response to an apparent CPU exhausting DoS attack).

5. Basic message exchange

5.1. Overview

This section describes different configurations of SEND-enabled nodes with varying signing capabilities and their interaction during a message exchange.

Case 1: when both nodes support the same two Signature Algorithms, they can pick the Signature Algorithm they prefer for signing and are able to verify each others signature. Figure 3 is an example of such a message flow.

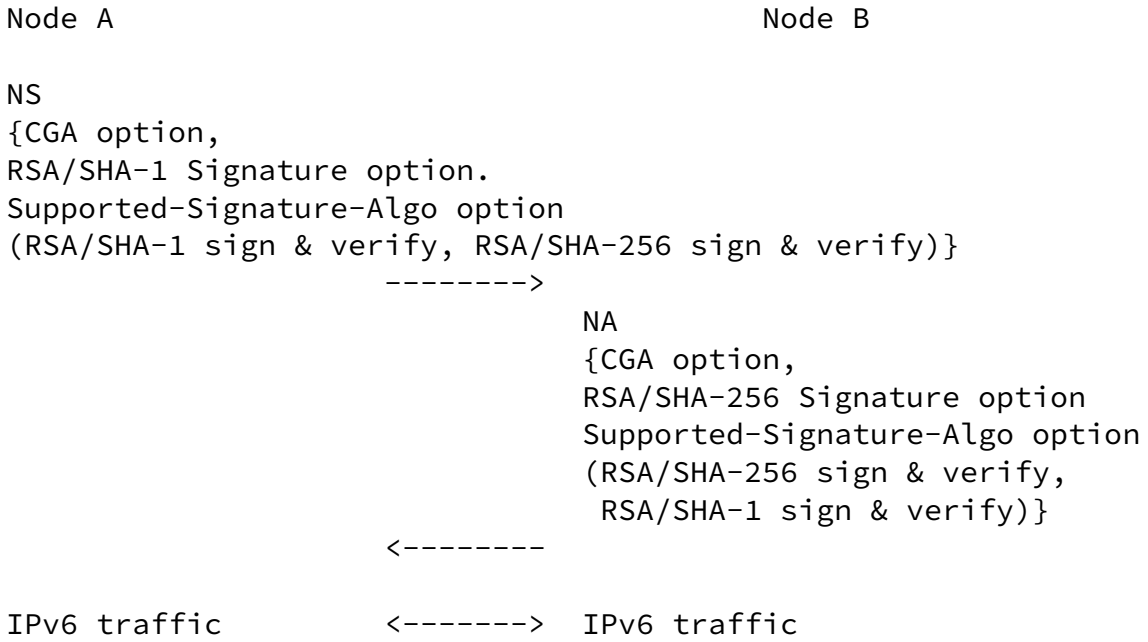


Figure 3: Basic message exchange - Case 1

Case 2: two nodes sharing at least one common Signing Algorithm must be able to securely communicate. Figure 4 is an example of such a message flow.

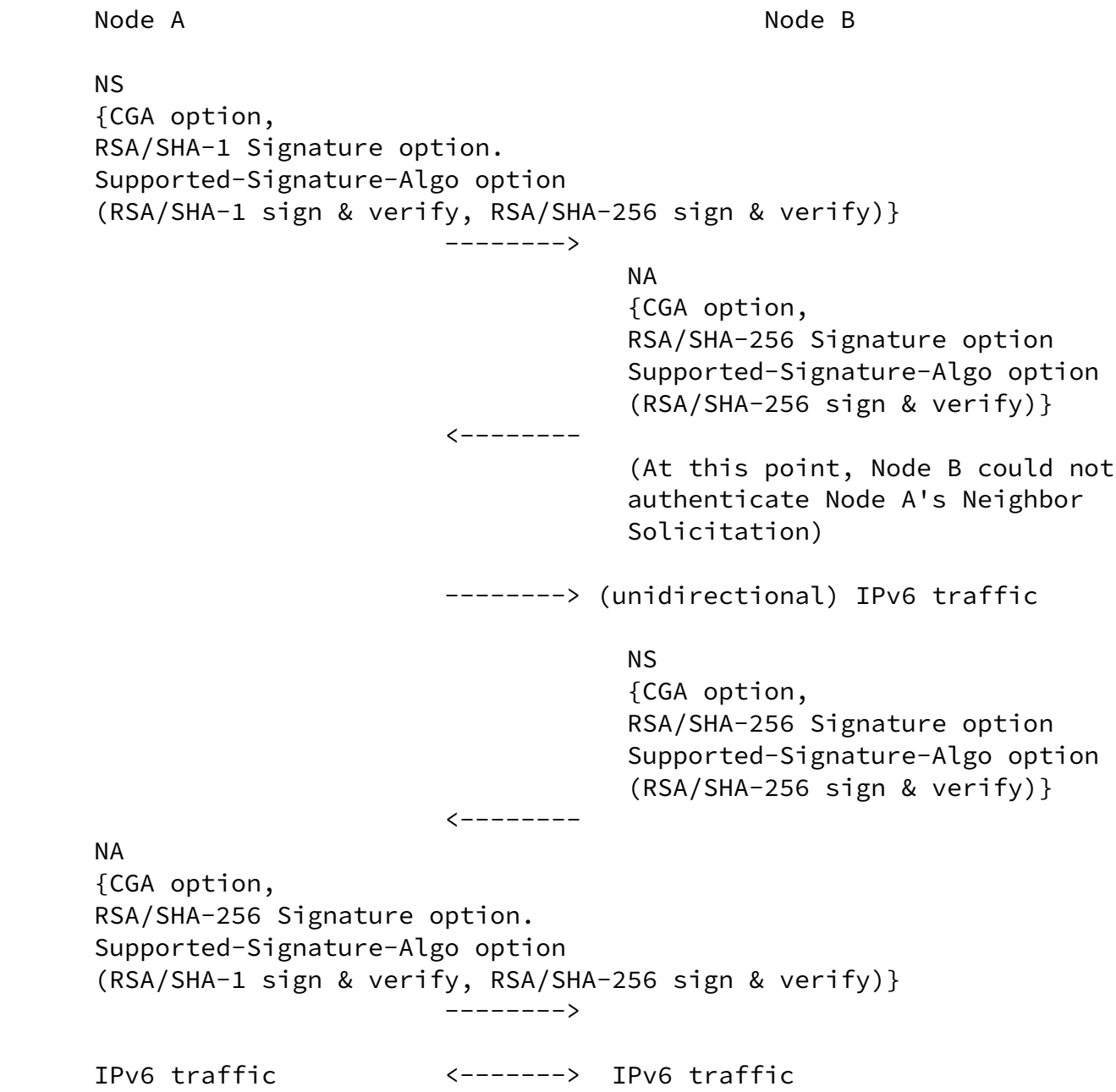


Figure 4: Basic message exchange - Case 2

Case 3: when two nodes have a disjoint set of Signature Algorithm support for signing, but the two nodes are able to verify each others, a communication is possible. Figure 5 is an example of such

a message flow.

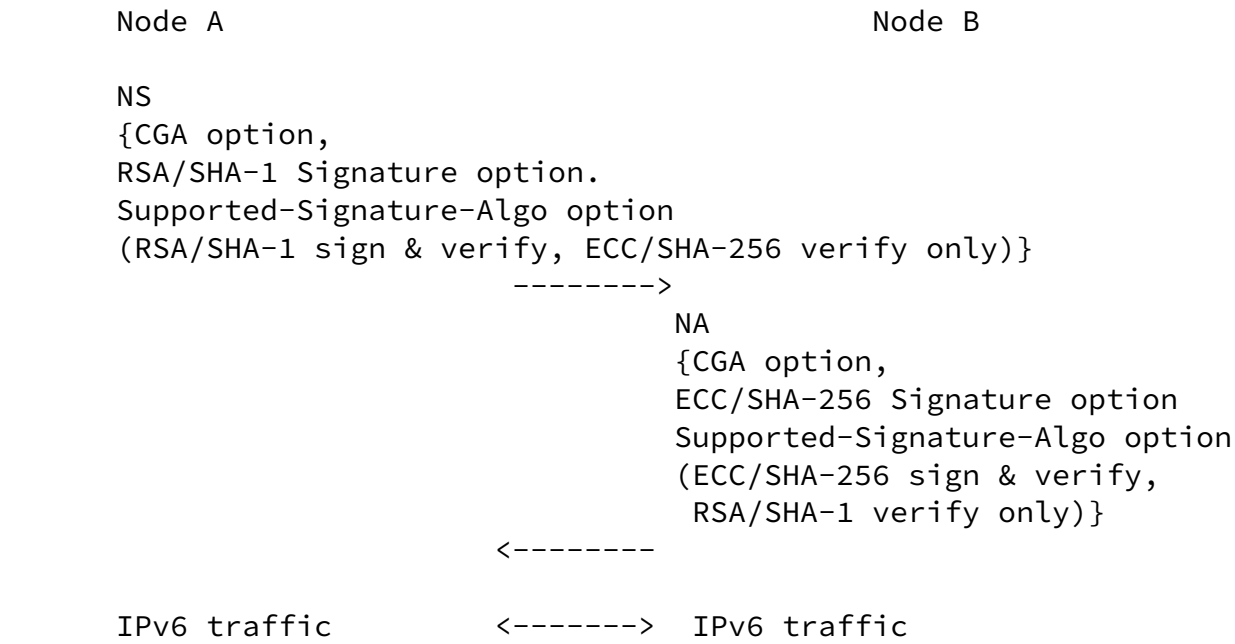
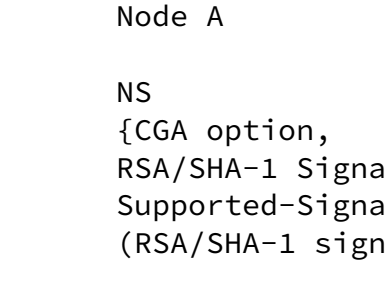


Figure 5: Basic message exchange - Case 3

Case 4: when two nodes have a disjoint set of Signature Algorithm support for signing, but one node is able to verify, a partial communication is possible. Figure 6 is an example of such a message flow.



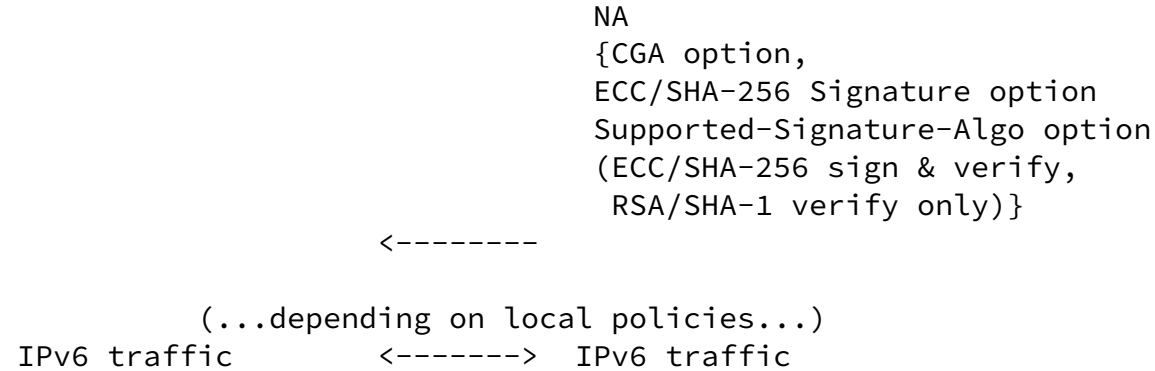


Figure 6: Basic message exchange - Case 4

Upon receiving the Neighbor Solicitation message, node B determines, through the Supported Signature Algorithm option, that node A will

not be able to verify any of its signature algorithm. However, based on their local policy, node B may answer and node A might decide to trust the insecure Neighbor Discovery (thus being vulnerable), see Section 4.2.

5.2. Sending Unsolicited Messages

When sending unsolicited message, a node MAY have to rely on the entries of its Neighbor Cache. The Neighbor Cache will provide hints concerning the Signature Algorithm supported by the neighbors. Neighbor Cache can assist the node in the Signature Algorithm selection process when:

- A router advertises unsolicited Router Advertisement message to the All-Nodes multicast address (e.g. to indicate a prefix lifetime is going down to 0). The router needs to know which signature algorithm(s) to use in order to send verifiable messages to hosts. To do so, the router MAY rely on the Neighbor Cache and compute an intersection of the set of all Supported Signature Algorithms. The router will then be able to advertise a Router Advertisement signed multiple times with the resulting subset of Supported Signature Algorithms or advertise multiple Router Advertisements, each signed with a single Signature Algorithm part of the intersection.

- A node sends unsolicited Neighbor Advertisement (e.g. when

changing its Link-Layer address). This is similar to the previous problem and can also be solved using the Neighbor Cache the same way.

- A router sends a Redirect message to a host. Choosing a supported signature algorithm without probing the node can be difficult. However, Neighbor Cache will most likely contain an entry for the host, prior to the decision to send a Redirect message, because of the Address Resolution process. This entry should contain information on the Supported Signature Algorithm(s) and thus provide hints concerning the Signature Algorithm to choose to sign the Redirect messages.

Note that the information on the neighbors with which a communication has occurred recently or is ongoing are in the Neighbor Cache and are maintained up to date through the Neighbor Unreachability Detection procedure.

5.3. Interaction with legacy RFC 3971 nodes

This section discusses the interaction between the nodes implementing this specification and the node that are not updated (referred here a

"legacy RFC 3971 nodes"). Following mechanism ensures backward compatibility:

- When the receiver is a legacy RFC 3971 node, the Supported Signature Algorithm option is an unknown option and MUST be ignored, as stated in Sections 6, 7 and 8 of document [RFC4861].

- The Universal Signature option is backward compatible with the RSA Signature Option. That is, Signature Type Identifier value 0 in the Universal Signature option indicates the use of the Digital Signature field and the Key Hash field as defined in [RFC3971]. Also, note that legacy RFC 3971 nodes fill the Reserved field of the RSA Signature options with zeros. When nodes implementing this specification receive messages emitted from legacy RFC 3971 nodes, their RSA Signature Option will be interpreted as an Universal Signature option and Signature Type Identifier will be read as the value 0.

The two above statements ensure that node implementing this

specification and using the RSA/SHA-1 Signature Algorithm interacts with legacy RFC 3971 nodes.

6. Authorization Delegation Discovery

This document advises that the Certificate Path MAY only contain certificates signed with a given signature algorithm. That is, if a node support this specific signature type, he will be able to verify all the certificates composing the Certificate Path. However, when only one Certificate Path composed of certificate signed by different signature algorithm is available, a node is expected to be able to process certificates signed with Signature Algorithm advertised by its Supported Signature Algorithm option. Thus, a router can verify when responding to a Certificate Path Solicitation message if the Certificate Path is verifiable by the requester.

7. Security Considerations

Section 4 presents a new Universal Signature option. A recommended use of this option is to allow signatures of equivalent security level (i.e. Public Keys with equivalent key lengths) for a same

node.

Usage of SHA-1 for signature is strongly NOT RECOMMENDED, and when available should be preferred by the usage of SHA-256. SHA-1 security has been proved to be flawed in the light of recent attacks [Recent_SHA-1_Attack] [NIST-st].

The Universal Signature option is vulnerable to downgrade attacks. That is, given that a node can employ multiple signature types (by varying the hash function), an attacker may choose to use a flawed one. To mitigate this issue, nodes are allowed, on a local policy, to refuse to check certain types of signature (i.e. those which are know to be flawed) and will treat the associated messages as unsecured. When trying to completely mitigate downgrade attacks, an administrator MAY deploy SEND-secured nodes only authorizing a single signature algorithm scheme. This comes at a price of a reduced interoperability.

8. IANA Considerations

Section 3 defines a Signature Type Identifier subfield containing new values corresponding to different Signature Algorithm. This document requests creation of a new registry to the IANA.

9. Acknowledgments

The authors gratefully acknowledge Jean-Michel Combes and Julien Laganier for their reviews. We also acknowledge Marcelo Bagnulo, Gabriel Montenegro, Greg Daley, Dave Thaler, Stephen Kent, Jari Arko, and Francis Dupont for their helpful feedback.

10. References

10.1. Normative References

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

10.2. Informative References

[cheneau-csi-ecc-sig-agility] Cheneau, T., Laurent, M., Shen, S., and M. Vanderveen, "ECC public key and signature support in Cryptographically Generated Addresses (CGA) and in the Secure Neighbor Discovery (SEND)", draft-cheneau-csi-ecc-sig-agility-02 (work in progress), June 2010.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4982] Bagnulo, M. and J. Arkko, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)", RFC 4982, July 2007.

[NIST-st] National Institute of Standards and Technology, "NIST

Comments on Cryptanalytic Attacks on SHA-1",
<http://csrc.nist.gov/groups/ST/hash/statement.html>.

[PKCS1] RSA Laboratories, "RSA Encryption Standard, Version 2.1", PKCS 1, November 2002.

[FIPS.180-2]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", September 2000,

<http://secg.org>.

[Recent_SHA-1_Attack]
McDonald, C., Haukes, P., and J. Pieprzyk, "SHA-1 collisions now 2^52", May 2009, <http://eurocrypt2009rump.cr.yp.to/837a0a8086fa6ca714249409ddfae43d.pdf>.

Appendix A. On the number of Universal Signature Options supported per CGA

RSA key length (bits)	Public exponent	Size of the DER-encoded Public Key (bytes)
384	3 or 17	76
384	65537	78
512	3 or 17	92
512	65537	94
1024	3 or 17	160
1024	65537	162
2048	3 or 17	292

2048	65537	294
3072	3 or 17	420
3072	65537	422
7680	3 or 17	996
7680	65537	998
15360	3 or 17	1956
15360	65537	1958

Table 1: Common sizes for DER-encoded RSA Public Key

RSA Key Length (in bits)	Size of the Digital Signature field without padding
384	48
512	64
1024	128
2048	256
3072	384

7680	960
15360	1920
+---+---+

Table 2: Common sizes of the Digital Signature field when using RSA

When using multiple Universal Signature options, one may reach before each Neighbor the Maximum Transfer Unit (which must be at least 1280 octets according to [RFC2460]). This section aims to approximate this limit.

Numerous factors (presence and number of option, size of public key, etc) influence the size of the Neighbor Discovery message. For example, when sending a SEND-secured Router Advertisement message:

- The IPv6 header is 40 bytes long. Described in [RFC2460].
- The bare Router Advertisement message (without any option) is 16 bytes long. Described in [RFC4861].
- A Prefix Information Option (can appear more than once) is 32 bytes long. Described in [RFC4861].
- A Source Link-Layer Option, when a IEEE 802 address is used, is 8 bytes long. Described in [RFC4861].
- A MTU Option is 8 bytes long. Described in [RFC4861].
- The CGA Option is the size of the CGA Parameters data structure plus 4 bytes rounded up to the closest multiple of 8 value. This option is defined in [RFC3971]. The CGA Parameters data structure (defined in [RFC3972] size depends on the following fields:

  * Modifier: 16 bytes long.
  * Subnet Prefix: 8 bytes long.
  * Collision Count: 1 byte long.
  * Public Key: variable size. Table 1 provides size of the

commonly used DER-encoded RSA Public Keys.

* Extension(s): variable size.

- The Timestamp Option is 16 bytes long. Defined in [RFC3971].

- The Nonce Option minimum size is 8 bytes long. Defined in [RFC3971].

- The Universal Signature option depends on the size of the Digital Signature. The fixed part of the option is 20 bytes long. This option is updated in this document. Table 2 presents common sizes for usual Digital Signature field when using RSA. This option size must be a multiple of 8 bytes.

A Router Advertisement message, carrying a Prefix Information Option and a Source Link-Layer Option, without Nonce, with one 1024-bits long RSA Public Key and a Public Exponent of 3 in the CGA Option is 456 bytes long.

Authors' Addresses

Tony Cheneau
Institut TELECOM, TELECOM SudParis, CNRS SAMOVAR UMR 5157
9 rue Charles Fourier
Evry 91011
France

Email: tony.cheneau@it-sudparis.eu

Maryline Laurent
Institut TELECOM, TELECOM SudParis, CNRS SAMOVAR UMR 5157
9 rue Charles Fourier
Evry 91011
France

Email: maryline.laurent@it-sudparis.eu

Sean Shen
Huawei
4, South 4th Street, Zhongguancun
Beijing 100190
P.R. China

Email: sean.s.shen@gmail.com

Michaela Vanderveen
Qualcomm

Email: mvandervn@gmail.com