DNSCurve: Link-Level Security for the Domain Name System
draft-dempsky-dnscurve-01

Abstract

This document describes DNSCurve, a protocol extension that adds
link-level security to the Domain Name System (DNS).

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF), its areas, and its working groups. Note that
other groups may also distribute working documents as Internet-
Drafts.

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."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at

This Internet-Draft will expire on August 30, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
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 BSD License.

Table of Contents

1. Introduction ......................................................... 3
   1.1. Terminology .................................................. 3
2. Overview ............................................................. 3
3. Base-32 Encoding .................................................... 4
   3.1. Examples ..................................................... 5
4. Encoding Public Keys in Name Server Names .......................... 5
5. Nonce Generation .................................................... 5
6. DNSCurve Expanded Formats .......................................... 6
   6.1. Streamlined Format ........................................... 6
   6.2. TXT Format .................................................. 7
7. UDP and TCP .......................................................... 8
8. Security Considerations ............................................ 9
9. IANA Considerations ................................................ 9
10. Acknowledgements ................................................ 10
11. References ........................................................ 10
   11.1. Normative References ....................................... 10
   11.2. Informative References ..................................... 10
Author’s Address ...................................................... 10
1. Introduction

DNS Curve adds link-level security to the Domain Name System (DNS). It includes a key distribution mechanism compatible with today’s name server software and registry services, and two packet formats: a simple streamlined format requiring minimal space and processing overhead and a mostly backwards-compatible format intended for use with strict firewalls and DNS proxies.

DNS Curve packets include a cryptographic MAC (aka authenticator) to provide integrity and availability. Clients can be confident that verified responses came from the appropriate server and were not forged by a blind or even sniffing attacker, while servers can be confident that responses will not be replayed against other unintended clients. Additionally, DNS Curve packets are encrypted to provide some confidentiality.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Overview

DNS Curve uses Curve25519XSalsa20Poly1305, a particular combination of the Curve25519, Salsa20, and Poly1305 primitives as described in [naclcrypto]. In particular, it is a cryptosystem featuring 256-bit public and secret keys, 192-bit nonces, and 128-bit authenticators.

Each DNS Curve client and server has a secret key and a corresponding public key. DNS Curve servers distribute their public keys by encoding them in name server names embedded in standard DNS NS records (as described in Section 4), while DNS Curve clients distribute their public keys by including them in their query packets. (Additional mechanisms for key distribution like DNSSEC’s trust anchors and DLV are possible, but not defined by this document.)

When a DNS Curve client is about to send a DNS query to a name server, if the client knows a DNS Curve public key for that name server, it MAY instead use that public key along with its own DNS Curve secret key and a nonce to protect its query in a "cryptographic box" as described in [naclcrypto]. The client then encodes this cryptographic box along with the nonce and its own public key as an expanded DNS Curve query packet, which it sends to the DNS Curve server instead of the original DNS query.
Upon receiving a DNS query packet, a DNSCurve name server should first treat the packet as a DNSCurve query packet by extracting the client’s DNSCurve public key, nonce, and boxed query and trying to open the box using the extracted public key and its own secret key. However, if this fails (i.e., the packet is not formatted as an expanded DNSCurve query packet or the box’s authenticator is invalid), then the server responds to the packet as a normal DNS packet.

Assuming the unboxing succeeds, then the server discovers the client’s original query packet. To send a response, the server chooses a nonce extension to append to the client-chosen nonce, and protects its response packet in a cryptographic box using the extend nonce and same keys used to unbox the client’s DNS query. The server then encodes this cryptographic box as an expanded DNSCurve response packet, which it sends to the DNSCurve client instead of the original DNS response.

Meanwhile, the DNSCurve client waits for an expanded DNSCurve response packet. If it receives a non-DNSCurve response packet, an expanded DNSCurve response packet with an invalid nonce (i.e., not an extension of its original nonce) or an invalid cryptographic box (i.e., cannot be opened using the same keys and the extended nonce), then it discards the packet and continues waiting. Once it receives a valid expanded DNSCurve response packet, it opens the cryptographic box to discover the server’s original DNS response.

3. Base-32 Encoding

Sometimes DNSCurve communicates arbitrary byte strings inside domain names. While the DNS protocol is 8-bit safe for names and labels (except for case-insensitive handling of ASCII alphabetic characters), many tools have trouble with arbitrary characters in domain names, in particular domain registrar software. To cope with this limitation, DNSCurve encodes byte strings using a set of safe alphanumeric characters.

In DNSCurve’s base-32 encoding, a byte string is interpreted as a number in little-endian form. Each 5-bit sequence of this number, from least significant to most significant, is encoded as one of the standard "digits" "0123456789bcdfghjklmnprstuvwxzy". A final sequence of fewer than 5 bits is zero-extended before encoding. Decoders MUST accept "BCDFGHJKLMNPQRSTUVWXYZ" as synonyms for "bcdfghjklmnprstuvwxzy".

For example, the two-byte string with bytes {0x64,0x88} (i.e., (100,136) decimal) is interpreted as the integer 0x8864 (i.e.,
34916). The bits 1000100001100100 of this integer are divided into 5-bit parts 00100, 00011, 00010, 00001, which in turn are encoded as "4", "3", "2", "1". The original string is therefore encoded as the string "4321".

N.B., this is not the same encoding as defined in [RFC4648]. In particular, the byte string is chunked into 5-bit sequences differently, and a different alphabet is used. The first allows DNSCurve public keys to be encoded slightly more compactly (see Section 4), and the second helps to further prevent false positives when searching for base-32 encoded strings in domain names.

3.1. Examples

<table>
<thead>
<tr>
<th>Byte string</th>
<th>Base-32 encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>{0x88}</td>
<td>&quot;84&quot;</td>
</tr>
<tr>
<td>{0x9f,0xb0}</td>
<td>&quot;zw20&quot;</td>
</tr>
<tr>
<td>{0x7d,0x3c,0x14,0x31}</td>
<td>&quot;rs89f&quot;</td>
</tr>
<tr>
<td>{0x2a,0x99,0x13,0x7e}</td>
<td>&quot;b9b71z1&quot;</td>
</tr>
<tr>
<td>{0x7e,0x69,0x3a,0xef,0xac}</td>
<td>&quot;ycu6urmp&quot;</td>
</tr>
<tr>
<td>{0xe5,0x3b,0x60,0xe8,0x15,0x62}</td>
<td>&quot;5zg06nr223&quot;</td>
</tr>
<tr>
<td>{0x72,0x3c,0xef,0x3a,0x43,0x2c,0x8f}</td>
<td>&quot;13hygxd8dt31&quot;</td>
</tr>
<tr>
<td>{0x17,0xef7,0x35,0x09,0x41,0xe4,0xdc,0x01}</td>
<td>&quot;rsxcm44847r30&quot;</td>
</tr>
</tbody>
</table>

4. Encoding Public Keys in Name Server Names

DNSCurve public keys are encoded in name server names as a 54-byte label consisting of the magic string "uz5" followed by the first 51 bytes of the base-32 encoding of the public key. (Curve25519 public keys are actually 255-bit integers in little-endian, so the 52nd byte of the base-32 encoding will always be "0".)

When a DNSCurve client is searching a name server name for a DNSCurve public key, it MUST check every label for an encoded public key. If multiple public keys are found, the left-most label MUST be chosen. String comparison with "uz5" MUST be performed case-insensitively.

5. Nonce Generation

For every request, DNSCurve clients generate a 96-bit nonce, and for every response, DNSCurve servers generate a 96-bit nonce extension. Nonces MUST be unique for distinct packets for the same client-server
key pair. A simple way to achieve this is to choose a unique nonce for each packet and for each retransmission. Additionally, servers MUST use a non-zero nonce extension (because nonces are zero extended in query packets). However, subject to these constraints, clients and servers may generate nonces however they choose.

Two recommended ways to generate a 96-bit nonce or nonce extension are

1. a 64-bit counter (starting at 1) followed by a 32-bit random number and
2. a 64-bit timestamp (e.g., nanoseconds since 1970) followed by a 32-bit random number.

In either case the 64-bit value MUST NOT decrease even if the software restarts or the system clock jumps backwards.

If multiple clients or multiple servers share a DNSCurve secret key, then they MUST make sure no two separate clients or servers generate the same nonce. A simple way to achieve this is to use nonce separation; e.g., if two servers share a DNSCurve key pair, one server could use only even nonces and the other could use only odd nonces.

6. DNSCurve Expanded Formats

DNSCurve defines two expanded formats: "streamlined" and "TXT". Each includes a format for expanded queries and a format for expanded responses. DNSCurve clients may send DNSCurve expanded queries using whichever format it chooses, but they are encouraged to use the streamlined format when possible. A DNSCurve server MUST support DNSCurve expanded queries in either format and MUST send expanded responses using the corresponding format.

6.1. Streamlined Format

An expanded query packet in streamlined format has the following bytes:

- 8 bytes: the magic string "Q6fnvWj8".
- 32 bytes: the client’s DNSCurve public key.
- 12 bytes: a client-selected nonce for this packet.
A cryptographic box containing the original DNS query packet.

An expanded response packet in streamlined format has the following bytes:

- 8 bytes: the magic string "R6fnvWJ8".
- 12 bytes: the client’s nonce.
- 12 bytes: a server-selected nonce extension.
- A cryptographic box containing the original DNS response packet.

Note that this streamlined response format does not repeat the client’s query name, and in particular does not repeat the client’s public key. However, it does repeat the client’s nonce.

6.2. TXT Format

The "TXT" format receives its name from the fact that expanded query and response packets in this format appear to casual inspection to be standard DNS packets with two possible exceptions: 1) the query name may exceed 255 bytes and 2) the total packet may exceed 512 bytes.

When encoding an expanded query packet in TXT format, a DNSCurve client MUST create a DNS standard query packet with the AA, TC, RD, RA, Z, and RCODE bits cleared, a single entry in the question section, and no records in the answer, authority, or additional records sections. The one question MUST ask for Internet-class TXT records for the query name constructed from the concatenation of the following labels:

- One or more labels, each label except the last being exactly 50 bytes, with the last label being at most 50 bytes. The concatenation of these labels is the base-32 encoding of a 96-bit client-selected nonce for this packet followed by a cryptographic box containing the original DNS query packet.

- One 54-byte label: the client’s DNSCurve public key, encoded as described in Section 4, except with the magic string "x1a" instead of "uz5".

- Zero or more additional labels specifying the name of the zone served by this server; i.e., the owner name of the relevant NS record.

A DNSCurve server SHOULD be lenient in decoding expanded query packets in TXT format. In particular, it MUST allow the RD bit to
either be set or clear, MUST allow records in the answer, authority records, and additional records sections, and MUST allow any labels to follow the DNSCurve public key in the query name. However, it MUST discard packets with the QR bit set.

When encoding an expanded response packet in TXT format, a DNSCurve server MUST create a DNS standard response packet copying the ID, RD bit, and questions section from the expanded query packet, setting the AA bit, leaving the TC and RA bits cleared and Z and RCODE values set to 0, containing one record in the answer section, and no records in the authority records or additional records section. The record in the answer section MUST be an Internet-class TXT record for the query name from the questions section with a TTL of 0. The RDATA of this record is the 96-bit server-selected nonce extension followed by a cryptographic box containing the original DNS response packet, encoded as a sequence of one or more strings of at most 255 bytes in standard DNS TXT RDATA format.

Similarly, a DNSCurve client SHOULD be lenient in decoding expanded response packets in TXT format. In particular, it MUST allow the server to alter the case of the query name when repeating it in the questions section.

7. UDP and TCP

If a normal DNS response packet is larger than 512 bytes then the server replaces it by an explicitly truncated packet. The client then tries again through TCP. Servers are not required to support TCP if no responses are above 512 bytes; clients are permitted to try TCP only if the server has explicitly indicated truncation.

DNSCurve does not require TCP support from servers that were not already supporting TCP. If the original DNS response packet is at most 512 bytes then the server is permitted to send the expanded response packet as a UDP packet. DNSCurve clients are required to set aside a 4096-byte buffer for receiving a UDP response packet.

If the original DNS response packet is larger than 512 bytes then it is replaced by an explicitly truncated packet and the truncated packet is protected by DNSCurve. In this case the client tries again by TCP, sending its DNSCurve query packet through TCP and receiving the DNSCurve response through TCP.

TCP is considerably more expensive for clients and servers than UDP is, and TCP has no protection against denial of service, so server administrators are advised to stay below 512 bytes if possible. DNSCurve adds some denial-of-service protection for UDP but cannot do
anything to help TCP.

If a protected DNS query includes an EDNS0 OPT record, then the payload size field refers to how large the original DNS response packet can be before encoding as a DNSCurve response packet. Clients MUST reduce the payload size they advertise to account for overhead from encoding the response as an expanded response packet. If a server builds a response within the payload size limit, but then cannot fit the encoded response in 4096 bytes, it MAY silently discard the response.

Even when DNSCurve transactions take place over UDP, they may still be vulnerable to denial-of-service attacks due to spoofed IP fragments if response packets are large enough to require IP fragmentation. Therefore, servers SHOULD try to keep response packets within the path’s MTU limits.

8. Security Considerations

The security of the Curve25519XSalsa20Poly1305 cryptosystem and its underlying cryptographic primitives is discussed in [naclcrypto]. In summary, it is designed to meet the standard notions of privacy and third-party unforgeability for a public-key authenticated-encryption scheme using nonces.

DNSCurve only provides link-level security between a client-server pair. It does not attempt to ensure end-to-end security for queries and responses relayed by untrusted DNS proxies and caches.

DNSCurve clients are free to choose whether or not to use DNSCurve on a per query basis; e.g., a client may decide to fallback to standard DNS after a few failed DNSCurve queries. Of course, DNSCurve cannot make any security guarantees for transactions that do not use DNSCurve, so clients are encouraged to use DNSCurve if possible.

DNSCurve adds some confidentiality by encrypting DNS packet contents but does not attempt to hide the length of the original DNS packet nor the source or destination of the packet. Additionally, the TXT format requires clients to reveal the zone they are querying.

9. IANA Considerations

This document has no actions for IANA.
10. Acknowledgements

The DNSCurve protocol was first introduced by Dan Berstein. Thanks also to Adam Langley and George Barwood for their contributions to early DNSCurve implementations.

Much thanks for feedback regarding this draft from George Barwood, Sjoerd Langkemper and Nikos Mavrogiannopoulos.

11. References

11.1. Normative References


11.2. Informative References


Author’s Address

Matthew Dempsky
OpenDNS, Inc.
410 Townsend St, Suite 250
San Francisco, CA 94107
US

Phone: +1 415 680 3742
Email: matthew@dempsky.org