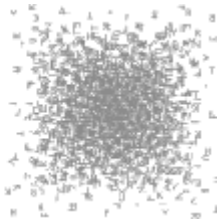
Fachhochschule Frankfurt am Main
University of Applied Sciences



Bachelorthesis

Implementierung und Einsatz von DNSCurve mit dem Protokoll Ipv6 unter DJBDNS

Fachbereich 2 Informatik und Ingenieurswissenschaften
Studiengang Informatik

1. Prüfer: Prof. Dr. Dipl.-Phys. Erwin Hoffmann
2. Prüfer: Prof. Dr. Erich Selder

vorgelegt von:

Thomas Elinkmann
Mat.-Nr.: 830050

Abgabetermin am 02.01.2012

Inhaltsverzeichnis

1. Einleitung....1
2. Allgemein....4
2.1 Ipv6....4
2.2 DNSCurve....6
2.3 Daemontools....10
2.4 DJBDNS....13
2.4.1 Tinydns....14
2.4.2 Dnscache....17
2.5 CurveDNS....17
3. Elliptische Kurven....19
3.1 Mathematische Grundlagen zu elliptischen Kurven....19
3.2 Elliptische Kurven in der Kryptographie....20
3.3 Schlüsselaustausch....22
3.4 Derzeitige Verwendung von elliptischer Kurven Kryptographie....23
3.5 Schwachstellen in der elliptischen Kurven Kryptographie....23
3.6 Weitere Entwicklung hyperelliptische Kurven....24
4. NaCl....25
4.1 Curve25519....27
4.2 Curve....27
5. DJBDNS+IPV6+DNSCurve....28
6. Installation und Konfiguration....31
6.1 Daemontools....32
6.2 Libev....33
6.3 NaCl....33
6.4 CurveDNS....33
6.5 ucspi-Tcp....35
6.6 Djbdns....36
6.8 Installation von DJBDNS+IPV6+DNSCurve....38
7. Aufbau der Kommunikation von DJBDNS+IPV6+DNSCurve 41
7.1 Ablauf der Kommunikation in Paketen....43
7.2 Leistung....50
8. Anwendung von CurveDNS und DNSCurve....52

9. Fazit.....53
10. Literaturverzeichnis.....54

1. Einleitung

Mit steigender Nutzung und der Verbreitung des Internets musste eine Möglichkeit gefunden werden eine einfachere Kommunikation mit Servern aufzubauen, ohne die genauen IP-Adressen zu kennen.

Dies wurde mit dem Domain-Name-System (DNS) erreicht. Einfach gesagt werden mit dem DNS Namen zu IP-Adressen zugeordnet. Dies hat mehrere Vorteile. Die IP-Adresse muss dem Internetnutzer nicht bekannt sein damit er auf einen Server zugreifen kann. Falls es nötig ist, die IP-Adresse eines Servers zu ändern müssen lediglich die Einträge auf dem Nameserver geändert werden. Es ist damit nicht mehr erforderlich, dass jeder Nutzer die neuen IP-Adressen beherrscht.

Das DNS besitzt von sich aus kein Sicherheitskonzept. Es wird weder die Vertraulichkeit, die Authentizität, die Integrität oder die Verfügbarkeit gewährleistet. Eine Möglichkeit die Integrität einer DNS Abfrage zu erhalten gelingt erst durch die Anwendung von DNSSEC (eine Erweiterung des DNS). Hier wird mit Hilfe von Signaturen die Integrität gewährleistet. Allerdings sind die Inhalte der DNS-Pakete nach wie vor per Trafficanalyse (dh. Mitlesen das Kommunikation) für Fremde einsehbar.

Ein praktisches Beispiel:

Wir haben zwei Personen A und B. Beide befinden sich mit ihren Laptops in einem gesicherten W-LAN. A will in einem W-LAN eine Online-Überweisung ausführen. Sobald A sich auf der Internetseite seiner Bank eingeloggt hat, ist die Übertragung natürlich verschlüsselt. Das eigentliche Problem für A ist aber, zuerst einmal auf die Internetseite seiner Bank zu gelangen. Dafür wird bisher eine unverschlüsselte DNS-Anfrage an einen DNS-Server gesendet. Wenn jetzt B auf seinem Laptop sieht, dass A auf die Internetseite seiner Bank zugreifen will, kann er eine fingierte Antwort senden. Er leitet A auf eine Internetseite um, die zwar aussieht wie die Seite der Bank, sich aber nicht auf den Servern der Bank befindet. Dadurch kann sich B die kompletten Daten (Login-Daten sowie TAN) von A aneignen, ohne dass A etwas davon mitbekommt. Jetzt kann B sich mit den Daten von A bei der Bank anmelden und eine Überweisung tätigen.

Dies lässt sich zum Beispiel mit der bereits als Standard eingeführte Erweiterung des DNS, DNSSEC verhindern. Bei DNSSEC wird eine fingierte Antwort an Hand der Signatur erkannt und verworfen. Allerdings ist die Übertragung nach wie vor unverschlüsselt. Dadurch bleibt immer noch ein erhöhtes Risiko. Das heißt, B würde nach wie vor sehen, das A. Auf die Internetseite seiner Bank zugreifen möchte.

Um dieses Risiko auszuschließen, wurde mit DNSCurve eine Möglichkeit entwickelt, den gesamten Kommunikationsablauf zwischen einem Client und einem Nameserver zu verschlüsseln.

Ist eine DNS-Anfrage mit DNSCurve geschützt, sieht B lediglich das A ein verschlüsseltes Paket an einen Nameserver schickt. Was dieses Paket beinhalte ist ihm völlig unbekannt.

Integrität, Vertraulichkeit, Authentizität und Verfügbarkeit sind wesentliche Eckpunkte der IT-Sekurity. Die in den Abschnitten oben genannten Sicherheitskonzepte DNSSEC und DNSCurve werden nachfolgend gegenübergestellt und mit Bezug auf die Punkte Integrität, Vertraulichkeit und Authentizität in einem Dreieck ihren Schwerpunkten nach eingeordnet. Die Verfügbarkeit ist bei dieser Einordnung nicht mit einbezogen, da weder DNSCurve noch DNSSEC eine Möglichkeit besitzen, die Verfügbarkeit zu gewährleisten. Nameserver oder Forwarder könnten zum Beispiel mit DDOS Attacken blokkiert werden. Wobei DNSCurve in der Lage ist gefälschte Pakete zu erkennen und diese verwirft.

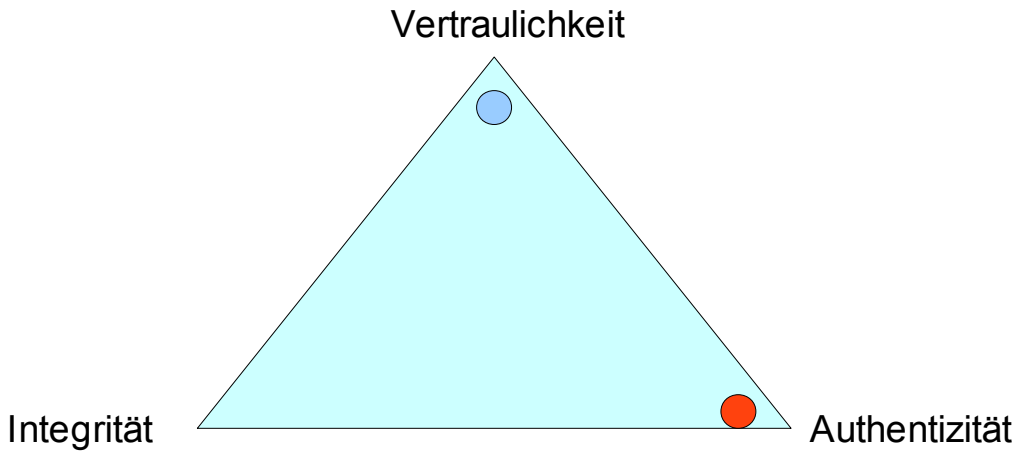
Integrität: DNSSEC schützt die Integrität eine DNS-Pakete durch eine verschlüsselte Signatur. Dadurch wird gewährleistet, dass eine Information von einem bestimmten Server kommt. Allerdings schützt DNSSEC nicht vor sogenannten „Man in the middel“ Attacken. Bei diesem Attacken würde, wie in dem praktischen Beispiel oben, ein Angreifer die Kommunikation mithören. Die Integrität von DNSCurve Paketen wird hingegen durch die Verschlüsselung gewährleistet. Da es einem Angreifer nicht möglich ist den Inhalt eines DNS-Paketes zu lesen, kann er die in dem Paket enthaltenen Informationen auch nicht verändern. DNSCurve schützt also die Integrität des DNS-Paketes während DNSSEC viel mehr die Integrität der in dem Paket enthaltenen Informationen bezüglich ihrer Herkunft schützt.

Authentizität: DNSSEC schützt die Authentizität von Nachrichten durch eine Signatur des Servers. DNSCurve besitzt keine Möglichkeit die Authentizität des Absenders des DNS-Paketes zu prüfen. Es sei denn, der öffentliche Schlüssel des Nameservers war bereits vorher bekannt oder der Schlüssel befindet sich bereits im Cache.

Vertraulichkeit: DNSCurve gewährleistet die Vertraulichkeit durch die Verschlüsselung. Bei DNSSEC hingegen werden DNS-Pakete unverschlüsselt übertragen.

Sowohl DNSSEC als auch DNSCurve sind in Bezug auf Vertraulichkeit und Integrität unterschiedlich geschützt, aber vom Sicherheitslevel vergleichbar. In Bezug auf die Verfügbarkeit bieten beide Verfahren keinen Schutz.

Dies wird in der folgenden Abbildung veranschaulicht die Schwerpunkte von DNSSEC und DNSCurve. DNSSEC wird durch den roten Kreis und DNSCurve durch den blauen Kreis dargestellt.



Zeichnung 1: Dreieck der IT-Sekurity

Eine Implementierung von DNSCurve unter DJBDNS gibt es bisher nur für das Internetprotokoll Version 4. Auf Grund der vorher geschilderten Bedeutung von DNSCurve und der Erweiterung von Ipv4 auf Ipv6, befasst sich diese Arbeit mit den Grundlagen, den Programmen sowie den erforderlichen Erweiterungen zur Implementierung von DNSCurve unter DJBDNS mit Verwendung des Ipv6.

Dazu werden im Kapitel 2 die allgemeinen Grundlagen vorgestellt, die für diese Arbeit von Bedeutung sind. Zunächst wird auf das Internet-Protokoll Version 6 und auf DNSCurve eingegangen. Anschließend werden die Programme beschrieben, die nötig sind, eine mit DNSCurve geschützte Kommunikation zu ermöglichen.

Im Kapitel 3 werden die mathematischen und kryptografischen Grundlagen zu den elliptische Kurven beschrieben.

Im Kapitel 4 wird die Bibliothek NaCl erläutert deren Funktionen DNSCurve nutzt.

Das 5. Kapitel beschäftigt sich mit der Erweiterung von DNSCurve für Ipv6 und noch einigen weiteren Anpassungen, die an DJBDNS vorgenommen wurden.

Das Kapitel 6 beschreibt die Installation und Konfiguration einer mit DNSCurve geschützten Verbindung. Es wird auf die Installation der DJBDNS+IPv6+DNSCURVE Version eingegangen und erläutert wie diese eingerichtet werden muss.

Im Kapitel 7 wird der Aufbau einer mit der DJBDNS+IPv6+DNSCURVE Version geschützten Kommunikation beschrieben. Der Ablauf der Kommunikation wird anhand der gesendeten DNS-Pakete erläutert. Ebenso wird das Verhältnis von Paketgröße und der zur Bearbeitung benötigten Zeit zwischen herkömmlichen DNS-Paketen und DNS-Paketen die mit DNSCurve übertragen wurden untersucht.

Das 8. Kapitel gibt Aufschluss über die derzeitige praktische Verwendung von DNSCurve.

Das Kapitel 9 beinhaltet das Fazit dieser Arbeit.

Im Anhang befindet sich eine CD auf der alle ursprünglichen Patches und Programme enthalten sind sowie die DJBDNS+IPV6+DNSCurve Version und diese Arbeit in Form eines PDF Dokuments.

2. Allgemein

In den Unterkapiteln Internet Protokoll Version 6 (Ipv6) und DNSCurve werden das Protokoll bzw. das Übertragungsverfahren erläutert. Im Abschnitt Daemontools wird erklärt, wie Daemons der zur Übertragung nötigen Programme gesteuert werden können. Im Unterkapitel DJBDNS wird zunächst die Programmsammlung erklärt. Anschließend wird näher auf die Programme TinyDNS und Dnscache eingegangen. Danach wird das Programm CurveDNS erläutert.

2.1 Ipv6

Bedingt durch die steigende Zahl der Internetnutzer, kam es zu einem Mangel an Ipv4-Adressen. Deshalb wurde es notwendig, eine Alternative für das Internet-Protokoll Version 4 (Ipv4) zu finden. Die Internet Engineering Task Force (IETF) begann 1995 mit den Arbeiten an dem Nachfolger des Ipv4, dem Internet-Protokoll Version 6 (Ipv6). 1998 wurde das Ipv6 von der IETF als standardisiertes Verfahren vorgestellt.

IPv6 beinhaltet neben dem erweiterten Adressraum auch noch weitere Verbesserungen, wie die Verbesserung der Header-Struktur, eine Unterstützung der Autokonfiguration und eine Verbesserung der Sicherheit. Im nachfolgendem Abschnitt wird nur der erweiterte Adressraum näher erläutert.

Der Adressraum bei Ipv6-Adressen beträgt 128 Bits. Bei Ipv4-Adressen beträgt der Adressraum lediglich 32 Bits. Die Zahl der verfügbaren IP-Adressen steigt also von 2^{32} auf 2^{128}. Ipv6-Adressen bestehen aus 8 Blöcken. Jeder Block besteht aus vier Zahlen. Die Zahlen werden nicht wie Ipv4-Adressen in dezimaler, sondern in hexadezimaler Form dargestellt. Die Blöcke werden mit einem „:“ abgegrenzt.

Zur Veranschaulichung:

Ipv4-Adresse : 10.0.2.16
Ipv6-Adresse : ADCF:BA56:6000:FEDC:0000:0000:0000:0000

Ipv6-Adressen können verkürzt dargestellt werden. Dabei können die Blöcke, falls sie ausschließlich Null-Werte enthalten, mit nur einem Null-Wert dargestellt oder sollten mehrere Blöcke mit Null-Werten aufeinander folgen durch „::“ ersetzt werden.

Beispiel:

Ipv6-Adresse : ADCF:BA56:0600:FEDC:0000:0000:0000:0000
Verkürzt Version: ADCF:BA56:600:FEDC::

Falls vor einem Wert wie im Beispiel der „6“ ein Nullwert vorkommt, kann dieser ebenfalls weggelassen werden.

Es gibt es noch einige besondere Formen von Ipv6-Adressentypen wie zum Beispiel Globale Unicast Adressen, Link-local Unicast Adressen, Site-local Unicast Adressen und Multicast Adressen. Ich möchte an dieser Stelle nur auf den Loopback-Adresstyp und die Adressen die zur Kompatibilität für Ipv4-Adressen definiert wurden, eingehen.

Die Loopback-Adresse ist für den Zugriff auf das derzeit genutzte System erforderlich. Sie ist also eine Alternative zur IP-Adresse des Systems. Diese wird verkürzt dargestellt als:

::1

Um eine Kompatibilität von Ipv4-Adressen und Ipv6-Adressen zu erreichen wurden drei Ipv6-Adresstypen definiert:

Ipv4-kompatiblen Ipv6-Adressen (Ipv4-compatible Ipv6-Addresses):

Diese Adresse unterscheidet sich nur durch die Null-Werte von der Ipv4-Adresse. Dadurch kann eine Ipv6-Adresse durch das weglassen der Null-Werte in einem Ipv4 Netz übertragen werden.

Ipv4-Adresse	a.b.c.d
Darstellungsform der Ipv4-compatible Ipv6-Addresses	0:0:0:0:0:0:a.b.c.d oder 0:0:0:0:0:0:ab:cd
Darstellung in Bytes	0000 0000 0000 0000 0000 0000 0a0b 0c0d

Tabelle 1: Darstellung der Ipv4-kompatiblen Ipv6-Adresse

Ipv4-Adressen die auf Ipv6-Adressen abgebildet werden (Ipv4-mapped Ipv6-Addresses):

Dieser Adresstyp dient zur Übertragung einer Ipv4-Adresse über ein Ipv6-Netz.

Ipv4-Adresse	a.b.c.d
Darstellungsform der Ipv4-mapped Ipv6-Addresses	0:0:0:0:0:FFFF:a.b.c.d oder 0:0:0:0:0:FFFF:ab:cd
Darstellung in Bytes	0000 0000 0000 0000 0000 FFFF 0a0b 0c0d

Tabelle 2: Darstellung der Ipv4-mapped Ipv6-Adresse

IPv4 Adressen die auf IPv6 Format übersetzt werden. (IPv4-translated Ipv6-Addresses):

Der Adresstyp dient zur Protokollübersetzung von Ipv4-Adressen in Ipv6-Adressen und umgekehrt.

Ipv4-Adresse	a.b.c.d
Darstellungsform der IPv4-translated Ipv6-Addresses	0:0:0:0:FFFF:0:a.b.c.d oder 0:0:0:0:FFFF:0:ab:cd
Darstellung in Bytes	0000 0000 0000 0000 FFFF 0000 0a0b 0c0d

Tabelle 3: Darstellung der IPv4-translated IPv6 Adresse

[28][26]

2.2 DNSCurve

Bei DNSCurve handelt es sich um ein von Daniel J. Bernstein entwickeltes Verfahren, DNS-Pakete verschlüsselt zu übertragen. Es nutzt ein von der NaCl Bibliothek (siehe Kap. 4) zur Verfügung gestelltes Verschlüsselungsverfahren Curve25519, das ebenfalls von Bernstein entwickelt wurde. Curve25519 nutzt elliptische Kurven für die Verschlüsselung (siehe dazu Kap.3).

Der Schlüsselaustausch funktioniert bei DNSCurve nach dem Diffie-Hellmann Protokoll. Das bedeutet, dass jeder Kommunikationsteilnehmer einen öffentlichen und einen privaten Schlüssel besitzt. Der öffentliche Schlüssel dient zum Verschlüsseln der Daten und der private Schlüssel zum Entschlüsseln. Bei DNSCurve wird der öffentliche Schlüssel im A-Record des Nameservers übertragen. Es wird die Verwendung eines von DNSCurve in Form eines Forwarders empfohlen. Falls ein Name-Server keinen A-Record hat in dem ein öffentlicher Schlüssel vorhanden ist, bleiben die Daten unverschlüsselt, werden aber übertragen. Für die Verwendung eines DNSCurve-Forwarders wird die IP-Adresse des Nameservers mit der IP-Adresse des DNSCurve-Forwarders ersetzt. Für TinyDNS würde der NS-Record wenn der DNSCurve-Forwarder die IP-Adresse 10.0.2.16 hätte folgendermaßen aussehen:

.example.org: 10.0.2.16:uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72.example.org

Für BIND würden die Einträge zum Beispiel lauten:

example.org. IN NS uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72.example.org.
uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72.example.org. IN A 10.0.2.16

Bei uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72 handelt es sich um eine 54 Byte Zeichenfolge. Mit „uz5“ (magic string) wird definiert, dass die Zeichenfolge ein DNSCurve-Schlüssel ist. Die restlichen 51 Byte sind der in Base-32 formatierte, öffentliche Schlüssel der decodiert den 32 Byte großen, öffentlichen Schlüssel von einem DNSCurve-fähigen Programm ergibt. Der Aufbau des öffentlichen Schlüssels wurde so gewählt, dass es keine Kompatibilitätsprobleme zu anderen Servernamen gibt.

Mit dem Base-32 Verfahren können Binärdaten einer Zeichenfolge, die nur aus 32 verschiedenen ASCII-Zeichen besteht, kodiert werden. Dafür werden 32 ASCII-Zeichen ausgewählt, mit deren Hilfe die Daten kodiert werden. Für DNSCurve wurde die Zeichenfolge „0123456789bcdfghjklmnpqrstuvwxyz“ gewählt.

Ein DNS-Paket im DNSCurve Format besteht aus dem schon oben beschriebenen eigenen, öffentlichen Schlüssel, dem Nonce und der kryptographischen Box in der sich das eigentliche DNS-Paket befindet.

Mit der **Nonce** (number used once) wird das Paket identifiziert. Die Nonce besteht aus einem 64 Bit Zähler und einer 32 Bit Zufallszahl. Durch eine Nonce sollen zum Beispiel Replay Attacken verhindert werden. Bei Replay Attacken zeichnet ein Angreifer den Kommunikationsablauf auf. Er versucht sich dann, durch das Senden von Informationen die während der Kommunikation ausgetauscht wurden als berechtigter Nutzer auszugeben.

Dadurch, das sich die Nonce bei jedem gesendetem Paket erhöht, kann erkannt werden ob ein Paket mit einer ungültigen Nonce gesendet wurde.

Ein DNS-Paket wird von DNSCurve in eine kryptographische Box gepackt. In der NaCl-Bibliothek ist dafür die crypto_box_curve25519xsalsa20poly1305 Funktion vorgesehen. Die kryptografische Box wird mit Hilfe des eigenen privaten Schlüssels, des öffentlichen Schlüssels des Kommunikationspartners und dem Nonce verschlüsselt.

DNSCurve unterstützt das streamlined Format und das TXT Format.

Das streamlined-Format ist wesentlich kompakter und effizienter als das TXT-Format.

Eine DNS-Anfrage die sich im streamlined-Format befindet beinhaltet einen String Q6fnvWj8, den 32 Byte öffentlichen Schlüssel, den 12 Byte großen Client Nonce und die kryptographische Box in der sich die DNS-Anfrage befindet.
Eine DNS-Antwort die sich im streamlined-Format befindet beinhaltet einen String R6fnvWJ8, 12 Byte Nonce, 12 Byte Server Nonce und die Kryptographische Box in der sich die DNS-Antwort befindet.

Einige Firewalls blockieren ihnen unbekannte Formate, die über den Standard DNS Port 53 empfangen werden. Aus diesem Grund unterstützt DNSCurve auch eine Übertragung über das nachfolgend beschriebene TXT-Format.

Ein DNS-Paket besteht aus Header-Sektion, Question-Sektion, Answer-Sektion, Authority-Sektion und Additional-Sektion. Die folgenden Screenshots zeigen den grundsätzlichen Aufbau einen DNS-Pakets.

```
+---------------------+
|       Header        |
+---------------------+
|      Question       | the question for the name server
+---------------------+
|       Answer        | RRs answering the question
+---------------------+
|     Authority       | RRs pointing toward an authority
+---------------------+
|     Additional      | RRs holding additional information
+---------------------+
```

Abbildung 1: Aufbau eines DNS-Pakets aus RFC 1035

Aufbau eines DNS-Headers.

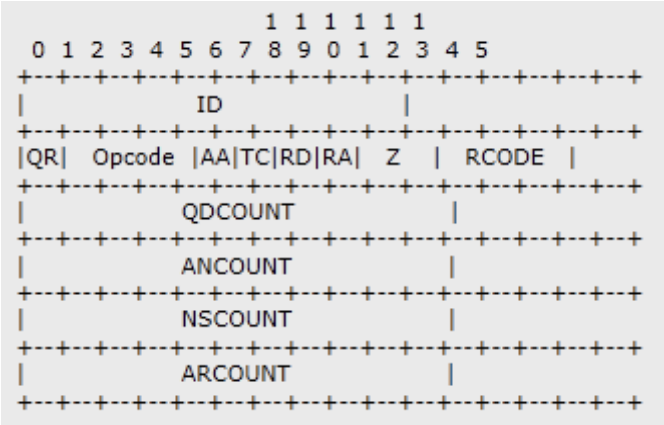


Abbildung 2: Aufbau der Header-Sektion aus RFC 1035

Aufbau der Question-Sektion:

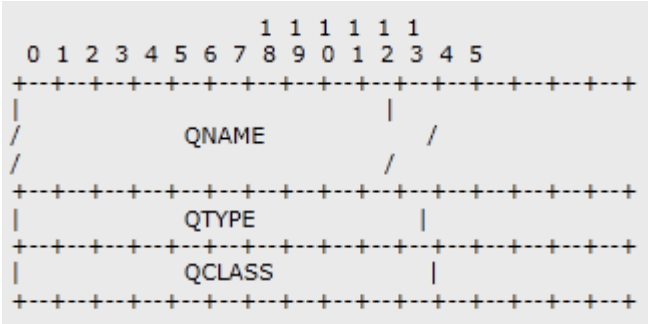


Abbildung 3: Aufbau der Question-Sektion aus RFC 1035

Eine DNS-Anfrage (Question) im TXT-Format nutzt die oben beschriebene Struktur des DNS-Pakets und wird wie folgt definiert:

Header-Sektion
ID = Zwei-Byte ID durch die durch den Client ausgewählt wurde
Opcode|AA|TC = Byte \000 (Abfrage Opcode 0, nicht autoritative, nicht verkürzt, keine Rekursion)
RD|RA |Z = Byte \000 (Rekursion nicht verfügbar, keine Z-Bits, RCODE 0).
QDCOUNT = Bytes \000\001 (Eine Anfrage)
ANCOUNT|NSCOUNT|ARCOUNT = Bytes \000\000\000\000\000\000 (keine Antwort-Records, keine authority-Records, und keine additional-Records)

Question-Sektion
QNAME = Name der DNS-Anfrage
QTYPE = Bytes \000\020 (Typ der Anfrage = TXT)
QCLASS = Bytes \000\001 (Internet-Query-Klasse)

Der Name der DNS-Anfrage (im Domain-Name-Format nach RFC 1035) besteht aus einem oder mehreren Labels. Jedes Label muss genau 50 Bytes groß sein. Das letzte Label darf höchstens 50 Bytes groß sein. Die Verknüpfung dieser Labels ist die Basis-32-Codierung eines 12-Byte-Client-Nonce für dieses Paket gefolgt von einer kryptografischen Box die das Original-DNS-Abfrage-Paket enthält. Danach folgt ein Label das 54-Byte groß ist und den öffentlichen Schlüssel des Clients beinhaltet. Die Zeichenfolge mit dem der Schlüssel identifiziert wird ist statt „uz5“ jetzt „X1A“. Zuletzt folgen noch optionale additional Labels, die den Namen der Zone angeben.

Dieser Screenshot zeigt den Aufbau der Answer-Sektion

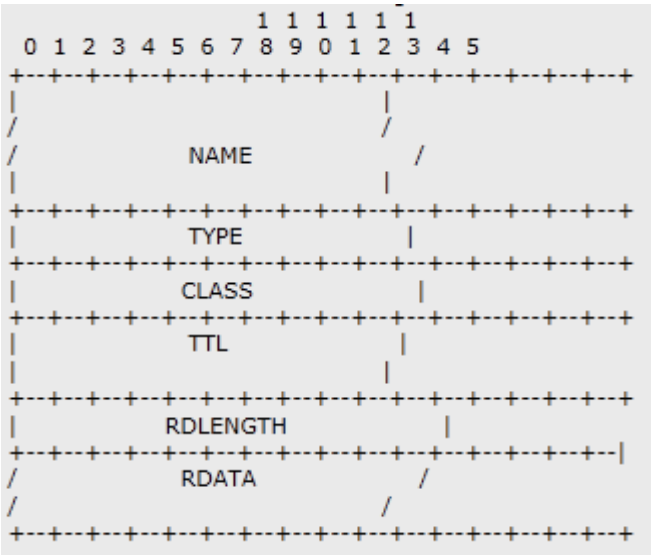


Abbildung 4: Aufbau der Answer-Sektion aus RFC 1035

Eine DNS-Antwort wird im TXT-Format folgendermaßen aufgebaut:

Header-Sektion
ID =Zwei-Byte ID die an die ID des Clients angepasst wurde.
Opcode|AA|TC= Byte \204 (Antwort Opcode 0, autoritär, nicht verkürzt, keine Rekursion).
RD|RA |Z|RCODE= Byte \000 (Rekursion nicht verfügbar ist, keine Z-Bits, RCODE 0).
QDCOUNT = Bytes \000\001 (eine Frage).
ANCOUNT = Bytes \000\001 (eine Antwort).
ARCOUNT = Bytes \000\000\000\000 (keine zusätzlichen Informationen).

Question-Sektion
QNAME = Der Name der vom Client gesendeten Abfrage Client.
QTYPE =Bytes \000\020 (Abfragetyp TXT).
QCLASS=Bytes \000\001 (Internet-Question-Klasse).

Answer-Sektion
NAME = Bytes \300\014 (der gleiche Name der vom Client gesendet wurde).
TYPE = Bytes \000\020 (Typ TXT).
CLASS =Bytes \000\001 (Internet-Question-Klasse).
TTL = Bytes \000\000\000\000 (TTL 0).
RDLENGTH = Zwei Bytes, in Big-Endian-Form, die Anzahl der Bytes von RDATA.
RDATA = In RDATA befindet sich eine oder mehrere Zeichenfolgen von höchstens 255 Byte. Vor jeder Zeichenfolge steht ein Byte das die Länge der Zeichenfolge angibt..

Die Verkettung der Strings in RDATA wird ohne Berücksichtigung der Bytes, mit denen die Länge der Strings angegeben werden, durchgeführt. Für die Verkettung wird eine 12-Byte Server-Nonce Erweiterung des Client-Nonce verwendet. Dahinter folgt die von der kryptografischen Box in der sich die ursprünglichen DNS-Antwort befindet.

DNSCurve sieht vor, dass auf eine DNS-Anfrage immer im gleichen Format geantwortet wird. Das bedeutet, wenn eine DNS-Anfrage im streamline-Format übertragen wurde, wird die DNS-Antwort auch im streamline-Format übertragen.

[10][25]

2.3 Daemontools

Bei den Daemontools handelt es sich um eine von Daniel J. Bernstein entwickelte Programmsammlung, mit der Daemons in auf Unix basierenden Betriebssystemen besser verwaltet werden können. Die Daemontools bestehen aus folgenden Programmen:

Daemon-Helper die der Ressourcen-Kontrolle dienen wie zum Beispiel envdir, envuidgid usw. auf diese Programme wird nicht näher eingegangen.

Supervise-Tools für das Starten der Daemons und um diese zu steuern. Auf weitere Programme wie supervise, fghack und pgrphack wird nicht eingegangen.

Logging-Tools für das erstellen von Log-Dateien und die für die Ausgabe von Fehlern.

Zeit-Tools Programme für das Datumsformat TAI64N.

In diesem Abschnitt wird nur auf die für den Betrieb von den Dnscache, TinyDNS und CurveDNS relevanten Programme eingegangen.

Supervise-Tools

Programm svscan
Alle über die Daemontools verwalteten Programme werden in das Verzeichnis /service/“Programm“/ verlinkt und über dieses Verzeichnis mit svc-Kommandos gesteuert.

Jedes Programm, das in das Service Verzeichnis verlinkt wurde und ein run-Skript besitzt, wird automatisch von svscan gestartet. Wenn ein Programm nicht automatisch gestartet werden soll, muss im Verzeichnis in dem sich der run-Skript befindet eine leere Datei mit dem Namen „down“ angelegt werden.

Programm svc

Für die Steuerung von Dnscache, CurveDNS und TinyDNS sind die Befehle svc -u. svc -d und svc -h von großer Bedeutung. Mit ihnen werden die Programme gestartet und beendet. Bei CurveDNS wird mit svc -h der Cache gelöscht.

In der folgenden Tabelle werden die Steuerungsbefehle von svc aufgelistet:

svc-Befehl	Bedeutung	Auswirkung
Svc -u	Up	Startet den Daemon; beendet er sich, wird er neu gestartet.
Svc -d	Down	Beendet einen laufenden Daemonprozess mittels Term Signals.
Svc -o	Once	Startet den Daemon. Beendet er sich, wird er nicht neu gestartet.
Svc -p	Pause	Sendet das Signal STOP an den Daemon.
Svc -c	Continue	Sendet das Signal CONTINUE an den Daemon.
Svc -h	Hangup	Sendet das HUP-Signal an den Daemon; häufig werden hierdurch die Initialisierungsdateien neu gelesen.
Svc -a	Alarm	Sendet das ALRM-Signal an den Prozess, was i.d.R. Mit der Beedigung laufender Subprozesse quittiert wird.
Svc -i	Interrupt	Das Signal INT wird an den Prozess übermittelt.
Svc -t	Terminate	Das Signal TERM wird an den Prozess geschickt.
Svc -k	Kill	Das KILL-Signal wird an den Prozess geschickt.
Svc -x	Exit	Supervise beendet sich, sobald der Prozess stoppt bzw. Heruntergefahren wird.

Tabelle 4: Svc-Kommandos und ihre Bedeutung. (aus [5] Tablle 5-1)

Programm svstat

Der Status eines Daemons kann mit dem Programm „svstat“ ermittelt werden. Das Programm ist wichtig, da es dem aktuellen Status eines Daemon (ob er zur Zeit läuft oder ob er angehalten wurde) anzeigt.

```
svstat /service/“Programm“
```

Logging-Tools

Ein weiterer wichtiger Bestandteil der Daemontools sind die Loggingtools.

Programm readproctitle
Zum Einen gibt es das Tool readproctitle. Dieses schreibt schwerwiegende Fehler die beim Start eines Daemons auftreten in einen Puffer. Dieser kann mit Hilfe des folgenden Befehls ausgelesen werden:

```
ps -auxww | grep readproctitle
```

Programm multilog
Die Log-Dateien werden, bei über Daemontools gestartete Daemons, mit dem Programm multilog geschrieben. Dabei wird der Daemon, der über das Run-Sktipt gestartet wird, mit einem Run-Skcript für das Log über eine Unix-Pipe verbunden. Die einzelnen Log-Einträge werden in die Datei „current“ geschrieben. Für jeden Eintrag in die Log-Datei wird eine neue Zeile mit einem Zeitstempel verwendet.
Sollte eine Log-Datei eine festgelegte Größe übertreffen wird sie mit einem Datumstempel im TAI64N Format versehen und abgespeichert. Als Suffix bekommt die Datei falls sie korrekt geschrieben wurde, ein „.s“. Falls ein Fehler aufgetreten ist endet sie mit „.u“.

Zeit-Tools

Programm tai64n
Für Zeitangaben nutzt multilog das TAI64N Datumsformat. TAI steht für Temps Atomique International. Jeder Datumsstempel beginnt mit einem „@“ danach erfolgen in hexadezimaler Darstellung die Zeitangaben.

Programm tai64nlocal
Mit dem Programm tai64nlocal kann das TAI64N Format in eine für Menschen besser lesbare Form umgewandelt werden.

Eine detailliertere Ausarbeitung findet sich unter http://www.fehcom.de/qmail/docu/05_new.pdf

[5][6]

2.4 DJBDNS

Bei DJBDNS handelt es sich um eine Sammlung von Programmen die ebenfalls von Daniel J. Bernstein entwickelt wurden und die im Zusammenhang mit DNS stehen. Es umfasst folgende Programme in Gruppen geordnet:

Programme mit denen unterschiedliche DNS-Anfragen erstellt werden können.

dnsip liefert die IP-Adresse für einen FQDN (Fully Qualified Domain Name) zurück

dnsipq liefert den FQDN zu einem Namen zurück

dnsname liefert den Namen zur einer IP-Adresse über ein reverse lookup zurück

dnsmx liefert den MX-Record zu einem FQDN

dnstxt liefert den TXT-Record zu einem FQDN

dnsfilter liest mehre IP-Adressen über stdin ein und gibt die dazugehörigen Namen auf stdout aus.

Programme die für die Konfiguration oder für das Debuggen verwendet werden

dnsqr gibt die IP-Adresse zu einem bestimmten Typ und FQDN zurück.

dnsq wie dnsqr nur muss in diesem Fall auch der Server angegeben werden.

dnstrace überprüft alle DNS-Server eines Root-Servers auf die korrekte Auflösung eines FQDN mit einem bestimmten Typ. Es wird ebenfalls registriert ob die Server antworten und ob dies mit einer ausreichenden Geschwindigkeit geschieht. Außerdem wird überprüft ob der Aufbau der DNS-Antwort korrekt ist.

dnstracesort wandelt das Ergebnis von dnstrace in ein für Menschen lesbares Format um.

tinydns-get liefert den Eintrag zu einem FQDN direkt aus der data.cdb.

Dns-Server beziehungsweise Programme die für das Betreiben von einem DNS-Server notwendig sind.

TinyDNS ist ein DNS-Server

rbldns ist ein Server der IP-Adressen auflistet. Er wird zum Beispiel eingesetzt um Blacklist zu bilden. Mit der Hilfe von Blacklist wird Spam bekämpft.

axfrdns ist ein Server für den Zonentransfer.

pickdns ist **e**in DNS-Server, der versucht anhand der IP-Adressen der Einträge den geografisch nächsten zu ermitteln. Ist inzwischen in TinyDNS integriert.

Sonstige

Dnscache ist ein DNS-Cache.

Walldns ist eine reverse DNS Wall

axfr-get ein Client für den Zonentransfer.
[7]
In den nächsten beiden Abschnitten wird Detaillierter auf TinyDNS und den Dnscache eingegangen.

2.4.1 Tinydns

TinyDNS ist ein DNS-Server der ausschließlich iterative Anfragen beantwortet. Alle anderen Anfragen, wie zum Beispiel Anfragen die den Zonentransfer betreffen, werden verworfen. Bei der Konfiguration muss eine IP-Adresse angegeben werden. Diese IP-Adresse kann später im bei der Konfiguration erzeugten Verzeichnis im Unterverzeichnis „env“ in der Datei „IP“ geändert werden. TinyDNS reagiert ausschließlich auf UDP-Pakete auf dieser IP-Adresse unter Port 53. Nachdem TinyDNS eine DNS-Anfrage erhalten hat, durchsucht das Programm eine Binärdatei mit dem Namen „data.cdb“. Falls ein Eintrag gefunden wurde, wird die entsprechende IP-Adresse - bzw. bei einer reversen Suche der Name - zurückgegeben. Diese werden allerdings nicht an Port 53, sondern an einen zufällig generierten Port gesendet. Dies soll insbesondere sogenannte Birthday Attacken auf einen DNS-Cache erschweren.

Ablauf einer Birthday Attacke
Der Name Birthday Attacken wird vom Geburtstagsparadoxon hergeleitet. Kurz zusammengefasst handelt es sich dabei um die Verteilung von Geburtstagen. Je mehr Personen sich in einem Raum aufhalten desto größer ist die Wahrscheinlichkeit, dass zwei Personen am gleichen Tag Geburtstag haben. Bei den Attacken geht es darum, dass bei einer größere Zahl von gesendeten DNS-Anfragen, eine DNS-Anfrage per Zufall eine gültige TransaktionsID besitzt.

Es werden von einem Angreifer möglichst viele DNS-Anfragen an einen Nameserver gesendet. Anschließend sendet der Angreifer gefälschte DNS-Antworten an den Nameserver. Der Nameserver registriert einen korrekten Ablauf der Kommunikation weil die DNS-Anfrage und die DNS-Antwort zueinander passen und speichert das Ergebnis in seinen Cache. Alle weiteren DNS-Anfragen bekommen so eine falsche IP-Adresse für einen vom Angreifer kontrollierten Server.[9]

TinyDNS ist unempfindlich gegenüber Birthday Attacken, da TinyDNS keinen eigenen Cache besitzt der beeinflusst werden könnte.

Die Log-Datei von TinyDNS, zu finden unter dem bei der Konfiguration erstellten Verzeichnis im Unterverzeichnis „log/main/“ in der Datei „current“, ist wie nachfolgend beschrieben aufgebaut:

Zunächst steht die genaue Zeitangabe im TAI64N Format (ein von Daniel J. Bernstein entwickeltes Zeitformat) der IP-Adresse und dem Port auf den TinyDNS die Antwort gesendet hat, der ID, einem „+“ falls der Eintrag gefunden wurde und einem „-“ falls der Eintrag nicht gefunden wurde, dem Typ des gesuchten Records und die Domain die gesucht wurde. IP-Adresse, Port, ID und der Typ werden in hexadezimaler Darstellung in die Log-Datei geschrieben.

Bei der Konfiguration von Tinydns werden folgende Programme im angegebenen Verzeichnis erstellt.

Programm	Funktion	Aufruf
add-ns	Erzeugt einen SOA-Record, einen NS-Record und einen A-Record	add-ns Domain IPv4
add-childns	Erzeugt einen NS-Record und einen A-Record	add-childns Domain IPv4
add-host	Erzeugt einen A-Record mit dazugehörigen PTR-Record.	add-host Domain IPv4
add-alias	Erzeugt nur einen A-Record	add-alias Domain IPv4
add-mx	Erzeugt einen MX-Record.	add-mx Domain IPv4

Tabelle 5: Programme für das Hinzufügen von Informationen in die Datei data

Diese Programme können zum Befüllen eines TinyDNS-Servers mit Informationen genutzt werden. Bei der Ausführung eines dieser Programme greift es auf die Datei „data“ zu und fügt den Eintrag am Dateiende hinzu.

Es ist ebenfalls möglich, die Datei direkt zu bearbeiten. Da die vorgegebenen Programme teilweise deutliche Einschränkungen erzwingen, ist die direkte Bearbeitung sogar notwendig. Bearbeitet werden kann die Datei "data" mit jedem beliebigen Texteditor.

Zu den Einschränkungen zählt, dass über die Programme nur die FQDN und die IP-Adresse eingegeben werden. Werte, wie die Lebensdauer eines Eintrags, werden dann mit Standardwerten gefüllt. Ebenso können für NS-Records über die Programme keine Namen für den Server hinzugefügt werden. Dieses werden Automatisch nummeriert beginnend mit „a“. Auch für die restlichen Record-Typen gibt es teilweise Einschränkungen.

Sowohl nach dem Hinzufügen per Programm wie auch dem Hinzufügungen per Editor muss die Datei anschließend mit make in das cbd Format umgewandelt werden.

In der Tabelle auf der nächsten Seite sind alle Zeichen(Z) die bei TinyDNS in die Datei „data“ am Anfang des Eintrags stehen mit Funktion und Syntax aufgelistet.

In der folgenden Tabelle sind alle Zeichen in TinyDNS mit Funktion und Syntax aufgeführt.

Z	Funktion	Syntax
#	Kommentar	`#comment`
-	Kann bei automatischer Bearbeitung genutzt werden um Zeilen auszukommentieren	`-fqdn:ip:ttl:timestamp:lo`
.	Erstellt für einen Name-Server einen NS-Record, einen A-Record und einen SOA-Record	`.fqdn:ip:x:ttl:timestamp:lo`
&	Erstellt einen NS-Record und einen A-Record	`&fqdn:ip:x:ttl:timestamp:lo`
%	Wird zur Festlegung der Location (lo) benötigt, zum Beispiel intern und extern für alle IP-Adressen mit ipprefix	`%lo:ipprefix`
+	Erstellt einen Alias für einen fqdn mit einer anderen IP-Adresse	`+fqdn:ip:ttl:timestamp:lo`
=	Erstellt einen A-Record für einen FQDN und einen PTR-Record	`=fqdn:ip:ttl:timestamp:lo`
^	Erstellt einen PTR-Record für einen FQDN der auf den Domainnamen p verweist	`^fqdn:p:ttl:timestamp:lo`
C	Erstellt einen CNAME-Record für FQDN der auf den Domainnamen p verweist	`Cfqdn:p:ttl:timestamp:lo`
:	Es wir ein Record vom Typ n der rdata ausgibt für einen FQDN generiert	`:fqdn:n:rdata:ttl:timestamp:lo`
Z	Erstellt einen SOA-Record für einen FQDN. mname ist der Name vom ersten Nameserver, rname die E-Mail Adresse de Administrators. Mit ser, ref, ret und min können die Serial und Timing Einstallungen der Zone eingerichtet werden.	`Zfqdn:mname:rname:ser:ref:ret:exp:min:ttl:timestamp:lo`
@	Erstellt einen MX-Record und einen A-Record für einen FQDN. Das x steht für den Priorität und dist für den Host,	`@fqdn:ip:x:dist:ttl:timestamp:lo`
'	Erstellt einen TXT-Record mit dem String s für einen FQDN	`'fqdn:s:ttl:timestamp:lo`
*	Platzhalter für Beliebige Domains mit dem selben FQDN (Wildcards)	`*.fqdn`

Tabelle 6: Zeichen (Z), deren Funktion und die Syntax die in der Datei data verwendet werden muss. Legende :FQDN = fully qualified domain name, ip = IP-Adresse, ttl = Time to live (Lebensdauer im Cache in Sek.), timestamp = Gültigkeit des Eintrags (in TAI64N) ohne der Angabe von einer ttl wird der Zeitpunkt angegeben ab dem der Eintrag Gültig ist.

2.4.2 Dnscache

„Dnscache is a DNS cache“ (Zitat Bernstein). Das bedeutet, dass der Dnscache DNS-Anfragen und deren Ergebnis die über ihn laufen, zwischenspeichert. Dnscache akzeptiert rekursive DNS-Anfragen zum Beispiel von Webbrowsern.

Wenn zum Beispiel die IP-Adresse von google.de gesucht wird und eine Antwort von einem DNS-Server angekommen ist, wird diese Antwort zwischengespeichert. Wenn jetzt erneut nach der IP-Adresse für google.de gesucht wird, wird das Ergebnis aus dem Cache des Dnscache geladen. Es ist nicht nötig, wieder einen DNS-Server zu kontaktieren. Dies ist zum Einen für den Client schneller, zum Anderen verringert es auch die Auslastung der DNS-Server.

Der Dnscache akzeptiert sowohl UDP als auch TCP-Pakete auf Port 53. Die IP-Adresse auf die der Dnscache hören soll, muss im Unterverzeichnis „env“ in der Datei „IP“ angegeben werden. Um die Sicherheit zu erhöhen, muss diese IP-Adresse dann noch im Unterverzeichnis „roots/ips“ als Datei vorhanden sein. Soll der Dnscache zum Beispiel DNS-Anfragen von der IP-Adresse 10.0.2.15 akzeptieren, muss eine leere Datei mit der IP-Adresse als Name angelegt werden. Es ist auch möglich, Adressräume freizuschalten (zum Beispiel ein komplettes Netz 10.0.2). Mit dieser Einstellung könnten alle Rechner aus diesem Netz DNS-Anfragen an den Dnscache schicken (auch die IP-Adresse 10.0.2.16).

Für ausgehende Pakete, die an einen höheren Port gesendet werden kann ebenfalls im Verzeichnis „env“ in der Datei „IPSEND“ eine IP-Adresse festlegt werden. Standardmäßig ist hier 0.0.0.0 eingetragen. Dies entspricht der Primären IP-Adresse des Rechners auf dem der Dnscache läuft.

Der Dnscache kennt zwei Arten von DNS-Servern. Zum einen die Root-Server die im Unterverzeichnis „roots/servers“ in der Datei „@“ zu finden sind und Server für die eine Domain mit IP-Adresse festgelegt wurde. Dies geschieht indem im Verzeichnis „servers“ eine Datei mit einem Namen zum Beispiel example.org angelegt wird. In dieser Datei wird dann die IP-Adresse zum Name-Server angegeben der DNS-Anfragen zu diesem Namen auflösen kann. Wenn jetzt eine Anfrage nach test.example.org gestartet wird, leitet der Dnscache die DNS-Anfrage an die in der Datei festgelegte IP-Adresse weiter.
Dnscache entfernt automatisch authority-Records und additionl-Records aus den DNS-Antworten.
[8]

2.5 CurveDNS

Bei CurveDNS handelt es sich um ein Programm, das von der niederländischen Firma ON2IT entwickelt wurde. CurveDNS ist ein DNS-Forwarder da es selbst nicht über Informationen verfügt, eine DNS-Anfrage zu beantworten. Das Programm unterscheidet zwischen zwei verschiedenen Arten von DNS-Anfragen. Falls es sich bei einer Anfrage um eine mit DNSCurve verschlüsselte Anfrage handelt, versucht CurveDNS das DNS-Paket mit seinem privaten Schlüssel zu entschlüsseln. Wenn dieser Vorgang erfolgreich abschlossen werden konnte, wird der in dieser DNS-Anfrage enthaltene öffentliche Schlüssel des Absenders in den Cache von CurveDNS gespeichert.

Später werden die DNS-Antworten des Name-Servers mit dem öffentlichen Schlüssel des Absenders verschlüsselt. Nachdem der Schlüssel in den Cache geschrieben wurde, wird die DNS-Anfrage jetzt wieder unverschlüsselt an einen Name-Server übertragen. Die Antwort des Name-Servers wird dann wieder von CurveDNS mit DNSCurve verschlüsselt und an den Absender zurückgeschickt. Sollte das bei CurveDNS ankommende DNS-Paket um eine normale nicht verschlüsseltet DNS-Anfrage handeln, wird diese weitergeleitet. In diesem Fall verläuft die komplette Kommunikation zwischen dem Resolver der die DNS-Anfrage gestellt hat, CurveDNS und einem Nameserver unverschlüsselt.

CurveDNS kann auch Pakete von anderen DNS-Sicherheitsmechanismen (wie zum Beispiel DNSSEC) weiterleiten. Aber die Sicherheitsmechanismen, die alle bei dem Nameserver ankommenden DNS-Pakete auf ihre Absende-Adressen untersuchen, werden nicht funktionieren, da die Absende IP-Adresse die IP-Adresse von dem CurveDNS Server ist.

Die Installation von CurveDNS ist nicht sehr zeitaufwändig und kann ohne größere Änderungen an einem Nameserver ausgeführt werden. Es muss lediglich der öffentliche Schlüssel, der von CurveDNS generiert wurde, in den NS-Record des Nameservers aufgenommen werden.

CurveDNS sollte auf einen separaten Rechner installiert werden. Falls dies nicht möglich ist muss, da CurveDNS und der notwendige Nameserver beide den Standardport für DNS (Port 53) nutzen, eine weitere IP-Adresse für CurveDNS eingerichtet werden. Für CurveDNS kann der Empfangs und der Sendeport zwar mit sehr geringem Aufwand verändert werden, aber es birgt große Risiken. Falls der Empfangsport von CurveDNS verändert wird, ist es Resolvern die Port 53 nutzen nicht mehr möglich, eine Anfrage zu senden.

Für CurveDNS können sowohl IPv4-Adressen als auch Ipv6-Adressen als Eingangs- bzw. als Ausgangs IP-Adressen eingetragen werden. Wenn die IP-Adresse für den Empfang von Anfragen auf 0.0.0.0 gesetzt wird, akzeptiert CurveDNS alle IP-Adressen. Es ist aber auch möglich eine feste Auswahl zu treffen und diese im run-Skript einzutragen. Allerdings unterstützt CurveDNS keine Ipv4-mapped Ipv6-Adressen.

3. Elliptische Kurven

In den folgenden Unterkapiteln wird zunächst auf die mathematischen Grundlagen zu elliptischen Kurven eingegangen. Anschließend wird ihre Bedeutung für die Kryptographie näher erläutert. Zuletzt wird auf die derzeitig verwendeten Verfahren die elliptische Kurven zur Verschlüsselung nutzen eingegangen.

3.1 Mathematische Grundlagen zu elliptischen Kurven

In der Mathematik wird bereits seit mehr als hundert Jahren im Bereich der elliptischen Kurven geforscht. Die Abbildung 5 zeigt eine elliptische Kurve mit der Kurvengleichung $y^2 = x^3 - 6x + 10$. Diese Gleichung liegt allen nachfolgenden Abbildungen zu elliptischen Kurven zugrunde. Elliptischen Kurven sind singularitätsfreie algebraische Kurven der 3. Ordnung in der projektiven Ebene. Singularitätsfrei bedeutet vereinfacht ausgedrückt, dass die Kurve keine Spitze oder einen Punkt, an dem sich die Kurve kreuzt, besitzt. Beispiele für Kurven mit Singularitäten sind die Neilsche Parabel und das kartesische Blatt. Die elliptische Kurve muss sich in der projektiven Ebene befinden. Das bedeutet, dass es für die Punkte der Kurve eine Gerade gibt die zwei Punkte miteinander verbindet und das zwei Geraden die durch zwei Punkte laufen einen Schnittpunkt haben (Abbildung 6).

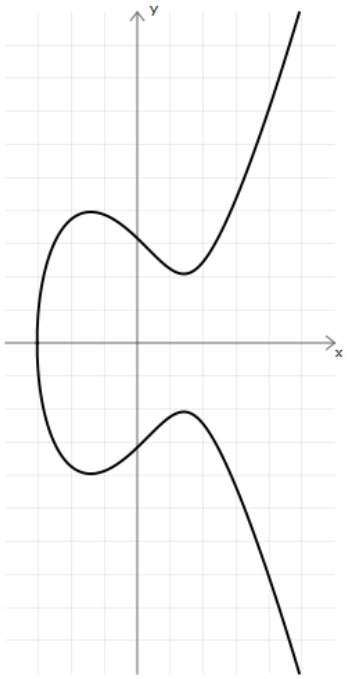


Abbildung 5: Elliptische Kurve aus dem Programm Elliptic Curve Plotter

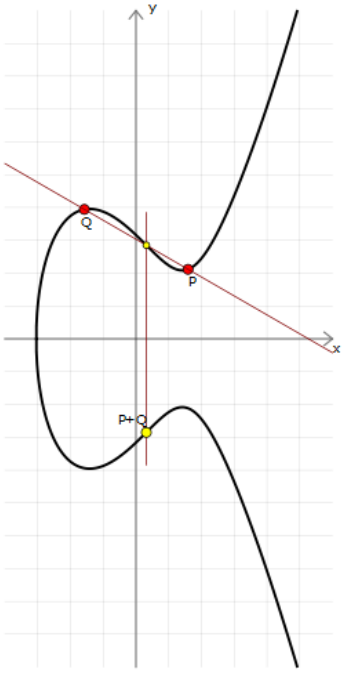


Abbildung 6: Elliptische Kurve mit einer Geraden P+Q aus dem Programm Elliptic Curve Plotter

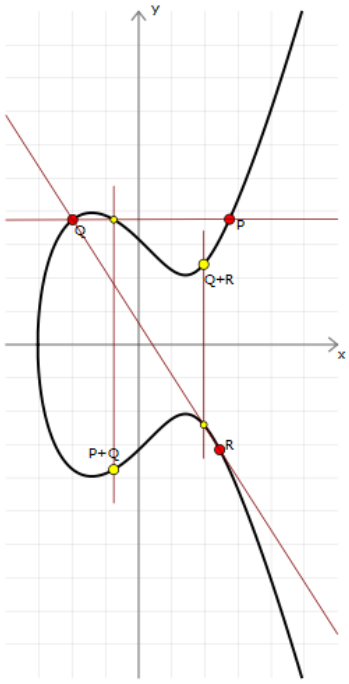


Abbildung 7: Elliptische Kurve mit zwei Geraden P+Q und Q+R aus dem Programm Elliptic Curve Plotter

Die auf einer elliptischen Kurve enthaltenen Punkte werden zu einer Gruppe zusammengefasst. Auf diese Gruppe können bestimmte Gruppengesetze angewendet werden. So zum Beispiel die Addition von Punkten die in Abbildung 6 und 7 gezeigt wurde. Es ist möglich einen Punkt zu verdoppeln (siehe Abbildung 8) und es kann das inverse eines Punktes gebildet werden (siehe Abbildung 9).

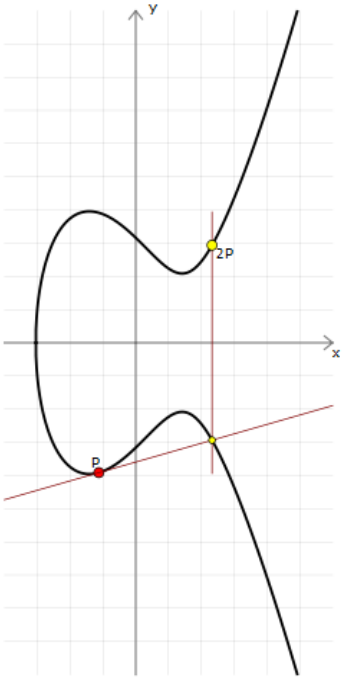


*Abbildung 8: Elliptische Kurve mit dem doppelten des Punktes P (2*P)aus dem Programm Elliptic Curve Plotter*

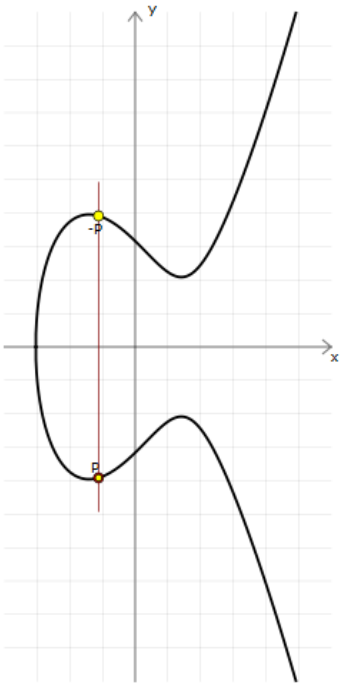


Abbildung 9: Elliptische Kurve mit dem Inversen Punkt -P aus dem Programm Elliptic Curve Plotter

Für die Kryptographie können nur elliptische Kurven verwendet werden deren projektive Ebene bzw. deren Körper aus endlich vielen Elementen besteht. Dadurch besteht auch die elliptische Kurve nur aus endlich vielen Punkten. In der Kryptographie werden als Körper für die Ebene bzw. die Kurve häufig die Primzahlen verwendet.

3.2 Elliptische Kurven in der Kryptographie

In der Kryptographie werden elliptische Kurven seit 1985 eingesetzt. Der größte Vorteil bei dem Einsatz eines Verfahrens das elliptischen Kurven verwendet (ECC-Verfahren) gegenüber anderen asymmetrischen Verschlüsselungsverfahren (wie RSA, benannt nach drei Mathematikern Rivest, Shamir und Adleman) besteht darin, dass sie mit deutlich geringeren Schlüssellängen auskommen. Die Sicherheit entspricht, auch bei geringerer Schlüssellänge, der von RSA. ECC ist wie RSA ein asymmetrisches Verfahren. Die Schlüssellänge von RSA musste bereits mehrfach verlängert werden, da die Rechenleistung stetig steigt. Nach dem Gesetz von Moore verdoppelt sich die Rechenleistung der Computerchips alle 18-21 Monate. [22] Die folgende Abbildung zeigt die Entwicklung der Schlüssellänge von RSA und ECC (Elliptic Curve Cryptography). ECC ist eines der Verfahren die zur Verschlüsselung elliptische Kurven verwendet.

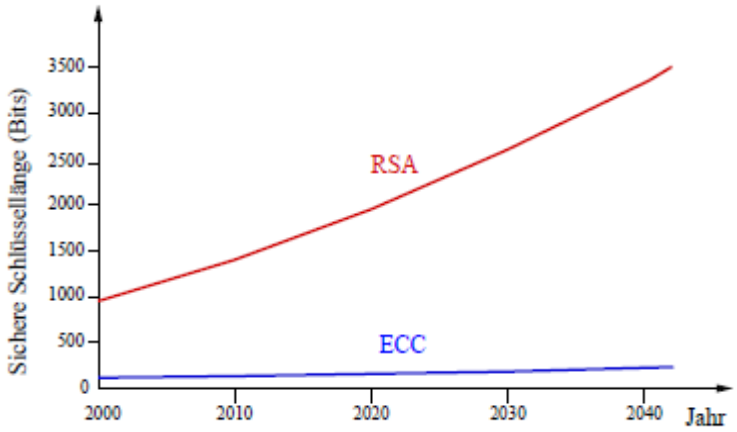


Abbildung 10: Vergleich der steigenden Schlüssellängen von RSA und ECC aus dem Cryptool Skript

Es ist deutlich zu erkennen, dass die Schlüssellänge für eine sichere Verschlüsselung mit RSA gegenüber der Schlüssellänge von ECC deutlich steigt. Ein Nachteil bei Verschlüsselungsverfahren die auf elliptischen Kurven basieren ist, dass es keine Standardisierung für die elliptischen Kurven bzw. deren Parameter gibt. Dadurch gibt es Kompatibilitätsprobleme bei unterschiedlichen Implementierungen. Im RFC 5639 werden zum Beispiel Parameter für elliptische Kurven festgelegt für den Einsatz in X.509 Zertifikaten.
Für elliptische Kurven werden in der Kryptographie meistens folgende Gleichungen verwendet.

$y^2 = x^3 + ax + b$ mit $4a^3 + 27b^2 \neq 0$ oder $y^2+xy=x^3 +ax^2+b$ mit $b \neq 0$[13]

Daniel J. Bernstein nutzt in Curve25519 die Montgomery Funktion

$y^2=x^3+Ax^2+x$ mit $A = 486662$ $(A – 2) / 4$ [27]

Die Bedingungen gewährleisten das die erstellte Kurve für Kryptographie anwendbar ist.

Das elliptische Kurven für die Kryptographie geeignet sind, hängt mit dem Diskreter-Logarithmus-Problem zusammen. Im Zusammenhang mit elliptischen Kurven wird es auch das Elliptische Kurven Diskreter Logarithmus Problem (ECDLP) genannt. Das bedeutet, das ein diskreter Logarithmus zum Beispiel der Form $a^x = m \bmod p$ leicht zu berechnen ist. Das Berechnen der Umkehrfunktion ist deutlich komplexer und beansprucht wesentlich mehr Zeit. Diesen Vorteil nutzt neben der ECC auch das Diffie-Hellman Verfahren. Es gibt einige Algorithmen mit denen versucht wurde, das Logarithmus-Problem zu lösen. Allerdings ist dies, bei den derzeitig verwendeten, Schlüsseln nicht in relevanter Zeit möglich. Zu den schnellsten Algorithmen zählen der Babystep-Giantstep-Algorithmus und die Pollard-Rho-Methode. Allerdings benötigen beide Algorithmen eine exponentielle wachsende Zeit für die Lösung des diskreten-Logarithmus-Problems. [32]

3.3 Schlüsselaustausch

Diffie-Hellman

Bei Diffie-Hellman handelt es sich um ein Protokoll das für die Übertragung eines Schlüssels von einem symmetrischen kryptographischen Verfahrens genutzt wird.
Es wurde 1976 von Martin Hellman, Whitfield Diffie und Ralph Merkle entwickelt. Ziel dieses Protokolls ist es, das Problem des sicheren Schlüsselaustauschs zu lösen. Wie soll ein Schlüssel zum Beispiel über das Internet übertragen werden, ohne das ein Angreifer die Möglichkeit hat ihn ebenfalls zu empfangen. Die Lösung wurde in der asymmetrischen Kryptographie gefunden. Für das Diffie-Hellman Protokoll werden zwei Schlüssel benötigt. Zum Einen ein öffentlicher Schlüssel der für jeden zugänglich sein soll und zum Verschlüsseln einer Nachricht benötigt wird. Zum Anderen ein privater Schlüssel mit dem die verschlüsselten Informationen wieder entschlüsselt werden können. In dem folgenden Abschnitt wird noch näher auf der Elliptic Curve Diffie Hellman-Verfahren eingegangen.

Elliptic Curve Diffie Hellman-Verfahren (ECDH)

ECDH ist ein Verfahren bei dem der Schlüsselaustausch ebenfalls nach dem Diffie-Hellman-Verfahren erfolgt. Für die Verschlüsselung selbst werden elliptische Kurven verwendet. Damit eine Verschlüsselung, die auf die Anwendung von elliptischen Kurven basiert funktioniert, müssen beide Kommunikationsteilnehmer die selbe elliptische Kurve verwenden. Anschließend muss ein Punkt P auf dieser Kurve ausgewählt werden. A und B müssen jeweils eine zufällige Zahl a und b generieren. Die öffentlichen Schlüssel der Kommunikationsteilnehmer werden berechnet mit $A_P = aP$ und $B_P = bP$. Diese öffentlichen Schlüssel werden dann ausgetauscht. Anschließend bilden beide den gemeinsamen geheimen Sitzungsschlüssel dafür berechnet A $S = aB_P$ und B $S = bA_P$. Der Sitzungsschlüssel S lautet dann also $S_{AB} = abP$. Wenn ein Angreifer jetzt die öffentlichen Schlüssel von A und B mithören würde könnte er mit ihnen nicht auf den Sitzungsschlüssel schließen, da der er nicht erkennen kann, was von aP und bP der Punkt und was der private Schlüssel ist. Die Sicherheit des ECDH beruht also ebenfalls auf dem Diskreten-Logarithmus-Problem. Im Zusammenhang mit dem Diffie-Hellman Verfahren wird es auch Diffie-Hellman-Problem genannt.

Ephemerales Diffie-Hellman

Bei dem Ephemeralem Diffie-Hellman wird ein mit Hilfe vom Diffie-Hellman Verfahren, erzeugter Sitzungsschlüssel noch zusätzlich mit einem öffentlichen Schlüssel von RSA oder DSA signiert. Google verwendet dieses Verfahren in Kombination mit mit dem ECDH (Elliptic Curve Diffie-Hellman) für den Schutz der von Google angebotenen Webdienste ein.
Alternative zu dem Diffie-Hellman Verfahren können auch zum Beispiel das Elgamal Verfahren verwendet werden. Elgamal ist ebenfalls ein asymmetrisches Verschlüsselungsverfahren das 1985 von Taher Elgamal entwickelt wurde. Es basiert auf den Diffie-Hellman-Algorithmus. [16]

Alle beiden Probleme das ECDLP und das Diffie-Hellman-Problem beruhen auf dem diskreter-Logarithmus-Problem. Das bedeutet, sollte eine Möglichkeit gefunden werden dieses Problem in sehr kurzer Zeit zu lösen wären sowohl eine Verschlüsselung mit elliptischen Kurven als auch das Diffie-Hellman Verfahren nicht länger sicher. [2][33]

3.4 Derzeitige Verwendung von elliptischer Kurven Kryptographie

Elliptic Curve Menezes Qu Vanstone (ECMQV)
ECMQV basiert auf dem Diffie Hellman Verfahren für den Schlüsselaustausch. Es wird für die Authentikation verwendet.

Elliptic Curve Integrated Encryption Scheme (ECIES)
ECIES ist ein hybrides Verschlüsslungsverfahren. Eine Nachricht wird mit einem symmetrischen Schlüssel verschlüsselt. Der Schlüssel wird anschließend mit einem asymmetrischen Verschlüsselungsverfahren übertragen.

Elliptic Curve Digital Signature Algorithm (ECDSA)
ECDSA ist eine Variante des DSA (Digital Signature Algorithm) die für die Verschlüsselungs elliptische Kurven verwendet. Dieser Algorithmus dient zum erstellen von Digitalen Signaturen.

Elliptic Curve Nyberg Rueppel (ECNR)
ECNR ist ein Signaturverfahren das zur Verschlüsselung elliptische Kurven einsetzt. [13]

Curve25519
Dieses Verschlüsselungsverfahren nutzt elliptische Kurven für die Verschlüsselung und wird im Kaptiel 4.1 näher erläutert. Für den Schlüsselaustausch ist DNSCurve vorgesehen das auf dem Diffie Hellmann-Verfahren basiert verwendet.

3.5 Schwachstellen in der elliptischen Kurven Kryptographie

Billy Bob Brumley und Nicola Tuveri haben eine Möglichkeit entdeckt, die Verschlüsselung von ECDSA zu knacken. Allerdings nicht mit einer mathematischen kryptoanalytischen Methoden, sondern mit einer sogenannten Timing Attacke. Dabei wurde die Zeit, die für die Verschlüsselung benötigt wurde, gemessen und anschließend die Berechnung für die Schlüssel durchgeführt. Dadurch war es möglich, den privaten Schlüssel zu ermitteln. Timing Attacken nutzen allerdings eher die Schwächen der Implementierung als auf Schwachstellen in der Verschlüsselung zu reagieren. Das Ergebnis wurde an einem Server direkt ausgeführt. Über das Internet wäre dieses Verfahren deutlich schwieriger anzuwenden, da die Übertragungszeit der Informationen von der Rechenzeit des Verschlüsselungssystems abgezogen werden muss. Mit Timing Attacken können nur Implementierungen angegriffen werden die nicht in konstanter Zeit verschlüsseln. [15][24]

In einem weiteren Fall gelang es in Jahr 2009 schweizer Forschern der Eidgenössischen Technischen Hochschule Lausanne eine 112 Bit Verschlüsselung, die auf elliptischen Kurven basierte, erfolgreich zu knacken. Dafür nutzten sie die Rechenleistung von 200 Playstation 3 und benötigten ein halbes Jahr. Nach eigenen Angaben wurde der Code um das diskrete-Logarithmus-Problem zu lösen noch optimiert und es wäre auch in dreieinhalb Monaten möglich. [30]

Laut Daniel J. Bernstein ist Curve25519 nicht mit Timing Attacken angreifbar. Die Schlüssellänge von DNSCurve bzw. Curve25519 liegt bei 32 Byte (also 256 Bit). Sie soll der Sicherheit einer RSA Verschlüsselung mit einer Schlüssellänge von 3000 Bit entsprechen.

3.6 Weitere Entwicklung hyperelliptische Kurven

Die hyperelliptischen Kurven wurden 1989 von Neal Koblitz für den Einsatz in der Kryptographie vorgeschlagen. Im Gegensatz zu elliptischen Kurven sind die Eigenschaften von hyperelliptischen Kurven noch nicht ausreichend erforscht. Deswegen werden sie bislang noch nicht in der Kryptographie eingesetzt. Es gibt allerdings einige Personen und Projekte die sich für eine Umsetzung einsetzten. Für hyperelliptische Kurven lassen sich nicht wie bei elliptische Kurven Gruppen definieren und damit keine Gruppengesetze festlegen. Dies erschwert den Umgang mit hyperelliptischen Kurven. [18][19]

4. NaCl

Bei NaCl (Networking and Cryptography library) handelt es sich um eine C/C++ Bibliothek die in einem Projekt von CACE (Computer Aided Cryptography Engineering) entwickelt wurde. Dieses Projekt wird von Tanja Lange und Daniel J. Bernstein geleitet.

Ein Ziel der Entwickler von NaCl war es Programmieren eine Auswahl von bereits vorhanden Verfahren wie AES oder ECDSA zur Verfügung zu stellen. Aber noch wichtiger sind die neu für das Projekt entwickelte Verfahren wie Curve25519. Allerdings wird darauf hingewiesen das es eventuell für bereits existierende Verfahren schnellere oder sichere Implementierungen gibt, die nicht in NaCl enthalten sind.

NaCl wird bei der Installation an die CPU des Systems angepasst. Dies gewährleistet eine maximale Geschwindigkeit bei der Anwendung von Funktionen aus der NaCl Bibliothek. Aus diesem Grund kann die Konfiguration von NaCl 10-15 Minuten dauern. In dieser Zeit werden die optimalen Compiler-Optionen und die durch das System unterstützten Funktionen ermittelt. Bei der Kompilierung vom NaCl werden zum Beispiel Änderungen für ein 32 Bit oder 64 Bit System automatisch vorgenommen.

Nachfolgend werden die Funktionen der NaCl Bibliothek beschrieben. Es werden zunächst immer nur das C++ Interface erläutert. Wenn das C Interface genutzt werden soll müssen, anstatt Strings für die Eingabe-und Rückgabewerte, Felder des Typs const unsigned char verwendet werden. Zusätzlich muss, falls eine Funktion eine Nachricht als Eingabewert besitzt, die Länge der Nachricht in Form eines usigned long long angegeben werden.

Die NaCl Bibliothek beinhaltet folgende Funktionen:

Mit der crypto_box Funktion wird eine Nachricht verschlüsselt und authentifiziert. Die crypto_box Funktion bekommt als Eingabewerte die Strings sk (den geheimen bzw. privaten Schlüssel des Senders), pk (öffentlicher Schlüssel des Empfängers), n (die Nonce), m (die Nachricht) und als Rückgabewert den String c (die verschlüsselte Nachricht).

Mit der crypto_box_open Funktion können Nachtrichten verifiziert und entschlüsselt werden. Die Funktion bekommt als Eingabewerte die Strings c (die verschlüsselte Nachricht), sk (den geheimen Schlüssel des Senders 32 Bytes), pk (den öffentlichen Schlüssel des Empfängers 32 Bytes) und n (die Nonce 24 Bytes). Als Rückgabewert liefert die Funktion m (die Nachricht).

Mit der crypto_box_keypair Funktion werden sowohl der geheime als auch der öffentliche Schlüssel generiert. Die Funktion liefert als Rückgabewert den öffentlichen Schlüssel und schreibt den geheimen Schlüssel in die String Variable sk.

Es gibt für C ein besonderes precomputation Interface. Damit kann der Ablauf der crypto_box Funktion aufgeteilt werden. Die crypto_box_beforenm Funktion bekommt als Eingabewerte sk und pk und schreibt den Rückgabewert in k (den geheimen Sitzungs-schlüssel 32 Bytes). Die crypto_box_afternm bekommt die Eingabewerte m, mlen, n und k.

Der Rückgabewert wird in die Variable c geschrieben. Durch das Aufteilen der Funktion kann die Geschwindigkeit erhöht werden, falls mehrere Nachtrichten an den selben Empfänger gesendet werden. Die Schlüssel müssen dann nicht für jede Nachricht neu berechnet werden.

Zum Entschlüsseln der Nachrichten kann ebenfalls zunächst die Funktion crypto_box_beforenm genutzt werden. Anschließend wird die Funktion crypto_box_open_afternm aufgerufen. Sie bekommt als Eingabewerte m, c, clen, n und k. Der Rückgabewert wird in die Variable m geschrieben.

Bei der im Kapitel 2.2 erwähnten Funktion crypto_box_curve25519xsalsa20poly1305 handelt es sich um eine Kombination von Curve25519, Salsa20 stream cipher und Poly1305 für die Nachrichten Authentifikation. Die als Standartfunktion für crypto_box verwendet wird.

Die Funktion crypto_scalarmult vervielfacht einen Punkt mit einer Zahl. Die Funktion benötigt die Eingabewerte n (Zahl) und p (Gruppenelement bzw. Punkt). Der Rückgabewert ist q (Gruppenelement).

Mit der Funktion crypto_scalarmult_base wird ein neues Gruppenelement aus einem Standard Gruppenelements und einer Zahl berechnet. Die Funktion benötigt den Eingabewerte n und liefert den Rückgabewert q.

Die Funktion crypto_secretbox verschlüsselt und authentifiziert eine Nachricht. Als Eingabewerte bekommt die Funktion m (Nachricht), n (Nonce) und k (geheimer Schlüssel). Der Rückgabewert ist c (die verschlüsselte Nachricht).

Mit der crypto_secretbox_open Funktion wird eine verschlüsselte Nachricht wieder entschlüsselt und verifiziert. Als Eingabewerte bekommt die Funktion c, n und k. Der Rückgabewert ist m.

Die Funktion crypto_stream Produziert aus einen geheimen Schlüssel und einer Nonce mit eine festgelegte Länge einen stream. Die Funktion bekommt die Eingabewerte clen(die Länge des stream), n (Nonce) und k (geheimer Schlüssel). Rückgabewert ist c (stream).

Mit der Funktion crypto_stream_xor kann eine Nachricht ver- oder entschlüsselt werden. Als Eingabewerte erhält sie m bzw. c, n und k. Als Rückgabewert liefert die Funktion c bzw. m.

Die Funktion crypto_auth authentifiziert eine Nachricht. Die Eingabewerte sind m und k. Der Rückgabewert ist a (Authenticator).

Mit der Funktion crypto_auth_verify wird eine authentifizierte Nachricht verifiziert. Die Eingabewerte sind a, m und k.

Die Funktion crypto_onetimeauth bildet eine Schlüsselziffer. Sie bekommt die Eingabewerte m (Nachricht) und k (geheimer Schlüssel). Als Rückgabewert liefert die Funktion a (Schlüsselziffer). Mit dieser Funktion kann eine Nachricht beglaubigt werden.

Es sollten niemals mehr als ein Nachricht mit dem selben geheimen Schlüssel beglaubigt werden. Dadurch steigt das Risiko das ein Angreifer den Schlüssel erfährt und in der Lage ist selbst Nachrichten zu beglaubigen.

Mit der Funktion crypto_onetimeauth_verify wird überprüft, ob eine Nachricht korrekt beglaubigt wurde. Dafür bekommt die Funktion als Eingabewerte a, m und k.

Die Funktion crypto_hash bildet den Hash-Wert einer Nachricht. Als Eingabewert bekommt die Funktion m (Nachricht). Normalerweise wird SHA-512 (Secure Hash Algorithm die Zahl 512 gibt die Länge des erstellten Hash-Wertes an).

Die Funktion crypto_verify vergleicht zwei Schlüssel auf Übereinstimmung damit ein Schlüssel verifiziert werden kann. zwei Felder des Typs const unsigned char. Es gibt sowohl eine crypto_verify_16 als auch eine crypto_verify_32 Version der Funktion. Die Zeit die benötigt wird ist Konstant damit wird Timing-Angriffen vorgebeugt.

Die Funktion crypto_sign signiert eine Nachricht. Sie bekommt als Eingabewerte m (Nachricht), sk (geheimer Schlüssel des Senders) der Rückgabewert ist sm (die signierte Nachricht).

Mit der Funktion crypto_sign_keypair werden der öffentliche und der geheime Schlüssel für die Funktion crypto_sign erstellt. Der geheime Schlüssel wird in die Varaiable sk geschrieben der Rückgabewert der Funktion ist der öffentliche Schlüssel.

Die Funktion crypto_sign_open verifiziert eine Nachricht. Als Eingabewerte bekommt die Funktion sm und pk. Der Rückgabewert der Funktion ist m.
[23]

4.1 Curve25519

Curve25519 ist ein Verschlüsselungsverfahren das auf der Anwendung von elliptischen Kurven (siehe dazu Kap. 3) basiert. Für die elliptische Kurve wird die Montgomery Funktion verwendet: $y^2=x^3+ax^2+x$ verwendet. Für die projektive Ebene bzw. für den Körper die Primzahl 2^{255} -19. Daher stammt auch der Name Curve25519. Der Schlüsselaustausch verläuft wie bei dem Elliptic Curve Diffie-Hellman-Verfahren. A und B bilden einen privaten Schlüssel a bzw. b als String mit $a \in \{0,1,....,255\}^{32}$ und $b \in \{0,1,....,255\}^{32}$. Mit diesem privaten Schlüssel wird mit der Funktion crypto_scalarmult ein Punkt vervielfacht. Dieser neue Punkt der sich ebenfalls auf der elliptischen Kurve befindet ist der öffentliche Schlüssel. [3][27]

4.2 CurveCP

CurveCP ist ein Protokoll das von Daniel J. Bernstein entwickelt wird. Das Ziel dieser Entwicklung ist es, alle Informationen, die über das Internet übertragen werden zu verschlüsseln. Das CurveCP Protokoll ist mit dem TCP-Protokoll (Transmission Control Protokoll) vergleichbar. Eine erste Implementierung von CurveCP ist bei NaCl enthalten. Für CurveCP werden dieselben Verschlüsselungsfunktionen verwendet wie für DNSCurve. Für den Schlüsselaustausch sollen die öffentlichen Schlüssel vor die Domainnamen der Server geschrieben werden.
[1] [20] [31]

5. DJBDNS+IPV6+DNSCurve

Auf Grund der Umstellung von IPv4 auf das IPv6 war es erforderlich DNSCurve auf die Verarbeitung von Ipv6-Adressen anzupassen.
Dazu wurde die DJBDNS+IPV6+DNSCurve Version durch die Kombination und Anpassung des Ipv6-Partches (von Felix von Leitner) und des DNSCurve-Patches (von Matthew Dempsky) erstellt. Mit DJBDNS+IPV6+DNSCurve ist es jetzt möglich, eine mit DNSCurve geschütze Verbindung zwischen dem Dnscache und einem CurveDNS Forwarder über Ipv6 aufzubauen. Anschließend leitet CurveDNS die DNS-Anfragen an TinyDNS bzw die DNS-Antworten an den Dnscache weiter. Es kann für TinyDNS auch ein anderer DNS-Server wie zum Beispiel BIND gewählt werden. In den nachfolgenden Abschnitten werden der Ipv6-Patch, der DNSCurve-Patch und die weiteren Veränderungen beschrieben.

Dnscurve-Patch
Matthew Dempsky hat einen Patch entwickelt, mit dem der Dnscache mit DNSCurve ver- und entschlüsseln kann. Nach der Installation des Patches läuft die Verschlüsselung automatisch ab. Da der Dnscache zunächst den öffentlichen Schlüssel benötigt erfolgt die Verschlüsselung der Information allerdings erst bei der zweiten DNS-Anfrage. Erst wenn dieser Schlüssel in den Cache geschrieben wurde, werden alle weiteren DNS-Anfragen verschlüsselt.

In der Log-Datei vom Dnscache findet sich nach der Installation vom Patch zusätzlich zu dem normalen Informationen noch ein „-“ falls das Ankommende Paket ein herkömmliches Paket war und ein „+“ falls es mit DNSCurve verschlüsselt werden konnte.

Der gepatchte Dnscache nutzt standardmäßig das streamlined-Format. Die Anfragen werden in diesem Format übertragen da es kleiner und effizienter ist. Bei der Verwendung des streamlined-Formates kann es zu Problemen mit einigen Firewalls kommen. Dann kann das TXT-Format aktiviert werden, indem eine Datei mit dem Namen „USETEXTFORMAT“ und dem Inhalt „1“ in dem Verzeichnis /service/dnscache/env/ erstellt wird.

Zum Beispiel mit:

```
echo 1 > /service/dnscache/env/USETXTFORMAT
```

Ipv6-Patch
Das DJBDNS-Paket unterstützt in der Implementierung von Bernstein nur Ipv4-Adressen. Ipv6-Adressen werden nicht akzeptiert. Felix von Leitner hat einen Patch geschrieben, mit dessen Hilfe DJBDNS auch unter Anwendung von Ipv6-Adressen lauffähig ist. Die derzeit aktuellste und die für diese Arbeit beschriebene ist Version 25. Durch den Patch können neben den bereits möglichen A-Records auch AAAA-Records erstellt werden. Diese enthalten dann Ipv6-Ardessen. In der folgenden Tabelle werden die Erweiterungen für TinyDNS gezeigt.

Z	Funktion	Syntax
.	Erstellt für einen Name-Server einen NS-Record, einen A-Record /AAAA-Record und einen SOA-Record	`.fqdn:ip/ipv6:x:ttl:timestamp:lo`
6	Erstellt einen AAAA-Record für einen FQDN und einen PTR-Record (Erweiterung durch den IPv6-Patch)	`~fqdn:ipv6:ttl:timestamp:lo`
3	Erstellt einen Alias für einen FQDN (Erweiterung durch den Ipv6-Patch)	`*fqdn:ipv6:ttl:timestamp:lo`

Tabelle 7: Erweiterung der möglichen Zeichen der Datei data durch den Ipv6-Patch

Bei der Konfiguration werden neben den bisherigen Programmen auch noch die folgenden Programme erstellt:

Programm	Funktion	Aufruf
add-host6	Erzeugt einen AAAA-Record mit dazugehörigen PTR-Record.	add-host6 Domain Ipv6-Adresse
add-alias6	Erzeugt nur einen AAAA-Record	add-alias6 Domain IPv6-Adresse

Tabelle 8: Programme die durch den IPv6-Patch für das Hinzufügen von Informationen in die Datei data dazugekommen sind

In der DJBDNS Programmsammlung wurden die Programme dnsq sowie dnsqr angepasst und können nun AAAA-Records interpretieren. Weiterhin wurde noch das Programm mit der Bezeichnung dnsip6 hinzugefügt. Es liefert die Ipv6-Adresse zu einem voll qualifizierten Domain Namen. Mit dem Patch wurden auch die reservierten Ipv6-Adressentypen wie zum Beispiel ::1 als Loopback-Adresse eingebaut.

Es werden allerdings keine Programme für das Hinzufügen von MX-Records oder NS-Records erstellt, die mit dem Patch noch nicht für Ipv6-Adressen angepasst sind. Für den Zonentransfer mittels axfr muss ein ebenfalls von Felix von Leitner entwickelter Patch für ucspi-tcp installiert werden.

Zusätzlich wurde ein von Russ Nelson entwickelter Patch (Anti-VeriSign Patch) übernommen. Mit diesem Patch können indem bei der Konfiguration erstellten Verzeichnis im Unterverzeichnis „env“ in der Datei „IGNOREIP“ IP-Adressen hinzugefügt werden, die ignoriert werden. [21]

Weitere Veränderungen betreffend TinyDNS in der DJBDNS+IPV6+DNSCurve Version:

Das angepasste TinyDNS akzeptiert für den Empfang von DNS-Anfragen unter FreeBSD sowohl eine Ipv4-Adresse als auch eine Ipv6-Addresse. Die IP-Adresse muss im, bei der Konfiguration erstellten, Verzeichnis „env“ in der Datei „IP“ eingetragen werden. Die Ipv6-Addresse kann im verkürzten Format angegeben werden.

Ein NS-Record für einen Nameserver kann jetzt in der veränderten Version eine Ipv6-Adresse oder eine Ipv4-Adresse haben. Falls es sich bei der IP-Adresse des Nameservers um eine Ipv4-Adresse handelt, wird wie bisher auch ein A-Record mit der Ipv4-Adresse des Nameservers zurückgegeben. Wenn die in dem NS-Record eingetragene IP-Adresse eine Ipv6-Adresse ist, wird ein AAAA-Record zurückgegeben.

Zum Beispiel

```
.example.org:10.0.2.16:ns1.example.org
.example.org:20010658000000202e018fffe98b03d:ns2.example.org
```

Alle IPv4-Adressen werden in den Log-Dateien von TinyDNS als IPv4-mapped IPv6-Adressen dargestellt.

Die bei der Konfiguration von einem mit Ipv6 gepatchten TinyDNS erstellten Programme add_host6 und add_alias6 fügen jetzt, je nach Eingabe, entweder eine Ipv4-Adresse oder eine Ipv6-Adresse hinzu. Die beiden Beispiele zeigen die Einträge die der Datei „data“ durch das ausführen des Programms angefügt werden wobei der letzte Wert für die Lebensdauer des Records steht:

Beispiel IPv4:

./add_host6 10.0.2.15 testip4.example.org
=testip4.example.org:10.0.2.15:86400

Beispiel IPv6

./add_host6 ::1 testip6.example.org
6testip6.example.org:00000000000000000000000000000001:86400

Einschränkungen:

Mit DJBDNS+IPV6+DNSCurve ist es nicht möglich eine Verbindung zu einem Name-Server herzustellen der eine Ipv4-mapped IPv6-Adresse hat. Dies hängt vor allem damit zusammen, dass DJBDNS+IPV6+DNSCurve für den Einsatz mit CurveDNS entwickelt wurde und CurveDNS keine Ipv4-mapped Ipv6-Adressen unterstützt.

6. Installation und Konfiguration

Damit eine DNS-Kommunikation mit DNSCurve verschlüsselt werden kann müssen folgende Programme installiert werden.

- Daemontools: Bei den Daemontools handelt es sich um einige von Daniel Bernstein entwickelte Programme um Daemons auf Unix oder Linux Systemen zu beeinflussen und zu steuern.

- Libev: Ist eine Bibliothek die Events verwaltet.

- NaCl: Ist eine C/C++-Bibliothek, die Ver- und Entschlüsselungsfunktionen zur Verfügung stellt.

- CurveDNS: Dient zur Entschlüsselung von mit DNSCurve verschlüsselten DNS-Anfragen und als Forwarder für alle anderen DNS-Anfragen.

- ucspi-tcp: Eine von Daniel J. Bernstein entwickelte auf TCP basierende Client-Server Applikation die für den Zonentransfer benötigt wird. [26]

- DJBDNS: Von Daniel J. Bernstein entwickelte Sammlung von DNS Programmen.

- DJBDNS-DNSCurve-Patch: Ein von Matthrew Demsky entwickelter Patch, der den Verkehr zwischen dem Dnscache und CurveDNS verschlüsselt.

Die Installationsanleitung geht von FreeBSD als Betriebssystem aus. Für andere Betriebssysteme sind eventuell Anpassungen nötig. Daemontools, CurveDNS und DJBDNS sind in der Port-Sammlung von FreeBSD enthalten. Es empfiehlt sich aber auf Grund der Übersichtlichkeit, die Programme selbst zu installieren. Zur weiteren Information über FreeBSD kann die Dokumentation unter http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/ aufgerufen werden.

Konfiguration im FreeBSD

Neue Benutzer-Accounts anlegen:

Mit Hilfe des Kommandos vipw müssen vier neue Benutzer erstellt werden. Ein Eintrag in der Datei muss folgendermaßen aussehen: der Benutzername, das Passwort (für die hier anzulegenden Benutzer wird kein Passwort eingetragen), die Benutzer-ID, Gruppen-ID, Gruppen, Login Klasse, das Home-Verzeichnis des Benutzers und die Shell. Zum Beispiel:

loguser:*:9001:9001::0:0:LOG:/etc/log/:/usr/sbin/nologin
dnscache:*:9002:9002::0:0:Dnscache:/etc/dnscache/:/usr/sbin/nologin
tinydns:*:9003:9003::0:0:Tinydns:/etc/tinydns/:/usr/sbin/nologin
curvedns:*:9004:9004::0:0:Curvedns:/etc/curvedns/:/usr/sbin/nologin

Virtuelle Hosts anlegen:

Um alle Programme auf einem Computer nutzen zu können, müssen zwei zusätzliche virtuelle Hosts angelegt werden. Dies ist nötig, da alle Programme Port 53 nutzen.

In der Datei „/etc/defaults/rc.conf“ müssen die folgenden Zeilen hinzugefügt werden:

if_config_em0_alias0=“inet 10.0.2.16 netmask 255.255.255.255“
if_config_em0_alias1=“inet 10.0.2.17 netmask 255.255.255.255“

Wenn DJBDNS mit Ipv6-Patch oder DJBDNS+IPV6+DNSCurve genutzt werden soll, müssen folgende Änderungen vorgenommen werden:

Es muss ebenfalls in der Datei „/etc/defaults/rc.conf“ die Unterstützung von Ipv6-Addressen für FreeBSD aktiviert werden. Der Wert von ipv6_enable muss auf „YES“ gesetzt sein.

Dann können in derselben Datei Ipv6-Adressen festgelegt werden:

ipv6_ifconfig_em0=“2001:658:0:2:2e0:18ff:fe98:b03d prefixlen 64“
ipv6_ifconfig_em0_alias0=“2001:658:0:2:2e0:18ff:fe98:b03e prefixlen 64“

Dieses Installationsanleitung setzt für den Benutzer Root Zugriffsrechte voraus.

6.1 Daemontools

Zunächst wird ein Verzeichnis „package“ erzeugt:

```
mkdir -p /package
chmod 1755 /package
cd /package
```

Download: wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz

Zum Entpacken:

```
gunzip daemontools-0.76.tar.gz
tar -xpf daemontools-0.76.tar
rm -f daemontools-0.76.tar
cd admin/daemontools-0.76
```

Kompilieren und Installieren:

```
package/install
```

Für den Start von svscan ist bei FeeBSD ein Neustart erforderlich. Bei Linux-Systemen startet svscan in der Regel sofort.
[5]

6.2 Libev

Download: wget http://dist.schmorp.de/libev/libev-4.04.tar.gz

Zum Entpacken:

```
tar -zxvf libev-*.tar.gz
```

Konfigurieren:

```
cd libev-*
./configure --prefix=/usr
```

Kompilieren und Installieren:

```
make install
```

Die Bibliotheken werden dann in das Verzeichnis /usr/lib kopiert.

6.3 NaCl

Eine ältere Version ist bei CurveDNS bereits integriert. Es ist allerdings zu empfehlen, immer die aktuellste Version zu nutzen.

Download: wget http://hyperelliptic.org/nacl/nacl-20110221.tar.bz2

Zum Entpacken:

```
bunzip2 < nacl-20110221.tar.bz2 | tar -xf -
```

6.4 CurveDNS

Download: wget http://curvedns.on2it.net/get/curvedns-0.87.tar.gz

Zum Entpacken:

```
tar -zxvf curvedns-0.87.tar.gz
```

Jetzt muss die veraltete Version von NaCl im Verzeichnis von CurveDNS gelöscht und die aktuelle Version in das Verzeichnis kopiert werden. Anschließend NaCl kompilieren mit:

```
./configure.nacl
```

Das Kompilieren kann bis zu 15 Minuten dauern. NaCl ermittelt die Rechnerleistung und richtet sich anschließend möglichst optimal ein. Weitere Informationen dazu finden sich im Kapitel NaCl.

Jetzt kann CurveDNS kompiliert werden mit:

```
./configure.curvedns
```

```
make
```

Anschließend müssen noch die Binärdateien von CurveDNS und dem Schlüsselgenerator von CurveDNS in das Verzeichnis „/usr/local/bin“ kopiert werden.

```
cp curvedns curvedns-keygen /usr/local/bin
```

Konfiguration von CurveDNS

Zunächst werden die Verzeichnisse für die Log-Dateien und für die Programm Daten erstellt:

```
mkdir -p /etc/curvedns/log /etc/curvedns/env
```

Anschließend wird das Run-Skript und das Skript zum Starten der Logs in die neu erstellten Verzeichnisse kopiert:

```
cp contrib/curvedns-run /etc/curvedns/run
cp contrib/curvedns-log-run /etc/curvedns/log/run
```

Das Run-Skript benötigt jetzt noch Zugriffsrechte, damit es ausgeführt werden kann:

```
chmod 755 /etc/curvedns/run /etc/curvedns/log/run
```

Im Run-Skript von CurveDNS werden die IP-Adressen, bei denen es auf ankommende DNS-Anfragen reagieren soll, der Port auf den es reagieren soll, die IP-Adresse des DNS-Servers und der Port auf den der DNS-Server reagiert, eingetragen. Es empfiehlt sich, den Standard DNS Port 53 zu nutzen. Es können sowohl Ipv4-Adressen als auch Ipv6-Addressen angegeben werden. Falls CurveDNS alle IP-Adressen akzeptieren soll, wird 0.0.0.0 eingetragen. Es ist auch möglich mehrere IP-Adressen einzugeben. Diese müssen, von einem Komma getrennt, direkt (also ohne Leerzeichen) eingegeben werden.

Die Verzeichnisse müssen den Benutzerkennungen zugewiesen werden:

```
chown -R curvedns /etc/curvedns
chown -R loguser /etc/curvedns/log
```

Das Verzeichnis „env“ bekommt nur Eingeschränkte Zugriffsrechte in diesem Verzeichnis wird der öffentliche Schlüssel abgelegt:

```
chmod 0700 /etc/curvedns/env
```

Als nächstes müssen für CurveDNS sowohl ein öffentlicher als auch ein privater Schlüs-sel erzeugt werden. Dies geschieht mit Hilfe des Schlüsselgenerators von CurveDNS. Als Parameter müssen dem Programm das Verzeichnis in dem sich die Daten von CurveDNS befinden und die Domain des DNS-Servers mitgegeben werden.

```
curvedns-keygen /etc/curvedns example.org
```

Anschließend wird nachfolgender Text ausgegeben:

```
Authoritative name server name:
uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72.example.org
DNS public key:
uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72
Hex public key:
42203e5071cd751393eafd16847fae58e68e937ebc3b1a4056714e67eaecfd08
Hex secret key:
481f40fedf711938be54833f274d196bfb0bceee19bb37bded50729212adc3a1
```

Der Text beinhaltet zuerst die neue Domain des DNS-Servers, dann den öffentlichen Schlüssel in hexadezimaler Darstellung und den privaten Schlüssel ebenfalls in hexadezimaler Darstellung. Wichtig ist zunächst der neue Name des Name-Servers. Dieser wird für die Konfiguration von TinyDNS benötigt. Der private Schlüssel wird im Verzeichnis "/etc/curvedns/env/" in der Datei „CURVEDNS_PRIVATE_KEY“ abgelegt damit es später von CurveDNS eingelesen werden kann. Es empfiehlt den neuen Namen des Name-Servers entweder in einer Datei zwischenzuspeichern oder abzuschreiben.

Der Debug-Level vom CurveDNS-Log kann auf unterschiedliche Stufen gesetzt werden (1-5). Wenn die maximalen Informationen in die Log-Dateien geschrieben werden sollen, muss der Debug-Level auf "5" gesetzt werden. Falls kein Debug-Level angegeben wird, werden lediglich Fehler in die Log-Dateien geschrieben.

```
echo 5 > /etc/curvedns/env/CURVEDNS_DEBUG
```

Der folgenden Befehl wandelt die Zeitangaben, die im TAI64N-Format in die Log-Dateien geschrieben werden, in für einen Menschen besser lesbares normales Zeitformat um.

```
tail -f /etc/curvedns/log/main/current | tai64nlocal
```

Damit CurveDNS mit den Daemontools gesteuert werden kann muss ein Softlink erstellt werden:

```
ln -s /etc/curvedns /service/curvedns
```

Mit dem folgenden Befehl kann der Cache von CurveDNS gelöscht werden:

```
svc -h /service/curvedns
```

[4]

6.5 ucspi-Tcp

Download: wget http://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz

Zum Entpacken:

```
gunzip ucspi-tcp-0.88.tar
tar -xf ucspi-tcp-0.88.tar
cd ucspi-tcp-0.88
```

Mit make und ./install wird der Code compiliert und die Binärdateien werden in den Ordner „/usr/local/bin/“ verschoben. [29]

6.6 Djbdns

Download: wget http://cr.yp.to/djbdns/djbdns-1.05.tar.gz

Zum Entpacken:

```
gunzip djbdns-1.05.tar
tar -xf djbdns-1.05.tar
cd djbdns-1.05
```

[7]

InstallationDJBDNS DNSCurve-Patch

Download: wget http://shinobi.dempsky.org/~matthew/patches/djbdns-dnscurve-20090602.patch

Falls Djbdns in den Ordner "package" entpackt wurde muss der Patch Vorgang ebenfalls vom Ordner "package" aufgerufen werden.

patch -p0 < djbdns-dnscurve-20090602.patch

Anschließend müssen noch die folgenden zwei Dateien verändert werden:

In der **conf-cc** muss der Pfad der include Dateien von NaCl angegeben werden. /path/to/nacl steht für den Pfad zu dem NaCl Verzeichnis, host steht für den Hostnamen des Rechners mit dem NaCl kompiliert wurde und arch für die Ermittelte Rechnerarchitektur.

-I/path/to/nacl/build/host/include/arch/

In der **conf-ld** muss der Pfad zu der Libary angegeben werden. Für /path/to/nacl, host und arch gilt das selbe wie bei conf-cc.

-L/path/to/nacl/build/host/lib/arch/

Mit make und ./install wird DJBDNS kompiliert und die Dateien werden in den Ordner "/usr/local/bin/" verschoben.

Konfiguration von TinyDNS

Die Konfiguration von TinyDNS erfolgt über das Programm tinydns-conf, dass bei der Kompilierung mit erstellt wurde. Sofern der Pfad nicht verändert wurde, befindet sich das Programm im Verzeichnis „bin“ und kann direkt ausgeführt werden. Das Programm benötigt folgende Parameter: die Benutzerkennung mit der TinyDNS laufen soll, die Benutzerkennung mit der in die Log-Dateien geschrieben werden soll, den Pfad zu dem Verzeichnis in dem die Daten gespeichert werden sollen und eine IP-Adresse von welcher TinyDNS Verbindungen akzeptiert.

Zum Beispiel könnte das Programm tinydns-conf mit folgenden Angaben aufgerufen werden:

```
tinydns-conf tinydns loguser /etc/tinydns 10.0.2.17
```

Zunächst muss noch ein Soft-Link mit dem Verzeichnis „/service/tinydns“ erstellt werden. Damit wird der TinyDNS-Prozess gesteuert. In diesem Fall:

```
ln -s /etc/tinydns /service/tinydns
```

Falls TinyDNS nicht beim Start des Betriebssystems automatisch gestartet werden soll muss eine leere Datei mit dem Namen „down“ in dem Verzeichnis „/etc/tinydns“ erstellt werden. Zum Beispiel mit:

```
touch down
```

Als nächstes müssen in dem Verzeichnis vom tinydns (/etc/tinydns) im Unterverzeichnis „root“ in der Datei „data“ die Records eingetragen werden.

Zunächst wird der NS-Record eingetragen. Dazu wird (wie im Beispiel unten) als erstes der FQDN (Fully Qualified Domain Name, also die Domain für die der Nameserver zuständig ist), die IP-Adresse (auf der CurveDNS DNS-Anfragen entgegen nimmt) und der Name des Nameservers (in diesem Fall der öffentliche Schlüssel vom CurveDNS gefolgt von dem FQDN) ausgewählt.

```
.example.org:10.0.2.16:uz5rxjwlmxlv26gf0u4lnrjbk13qnuxvc72153vrt9mdyp8
pgqr0mu.example.org
```

Für die inverse Auflösung ist das Netz, in dem sich die IP-Adressen für die inverse Auflösung befindet erforderlich. Dafür wird die IP-Adresse des Netzes in umgekehrter Reihenfolge angegeben. Mit 2.0.10 können alle IP-Adressen aufgelöst werden, die im Nameserver verfügbar sind und sich im Netz 10.0.2. befinden. Danach folgt die Angabe in-addr.arpa. Dies ist die Zone für das Reverse-Lookup, der IP-Adresse des Name Servers und dem Namen des Nameservers.

```
.2.0.10.in-addr.arpa:10.0.2.16:
uz5rxjwlmxlv26gf0u4lnrjbk13qnuxvc72153vrt9mdyp8pgqr0mu.example.org
```

Jetzt werden noch für dieses Beispiel drei A-RR mit passendem Reverse-RR angelegt. Diese Records könnten auch mit dem Programm add-host hinzugefügt werden.

```
=test1.example.org:10.0.2.18:3600
=test2.example.org:10.0.2.19:3600
=test3.example.org:10.0.2.20:3600
```

Anschließend muss noch die Text Datei „data“ in eine cbp Datei umgewandelt werden. Dies wird im Verzeichnis /etc/tinydns/root/ das make-Skript ausgeführt wird.

```
make
```

Konfiguration von Dnscache

Für die Konfiguration von dem Dnscache muss ebenfalls ein dazugehöriges Programm aufgerufen werden. Dieses Programm dnscache-conf benötigt die Benutzerkennung unter der das Programm ausgeführt werden soll, die Benutzerkennung für die Logs sowie das Verzeichnis in den die Daten geschrieben werden sollen. Zum Beispiel könnte das Programm folgendermaßen aufgerufen werden:

```
dnscache-conf dnscache dnslog /etc/dnscache
```

Wie bei TinyDNS muss auch hier ein Soft-Link erstellt werden.

```
Ln -s /etc/dnscache /service/dnscache
```

Falls der Dnscache nicht automatisch starten soll muss (wie bei TinyDNS) eine Datei „down“ erstellt werden.

Zusätzlich zur bisherigen Konfiguration müssen jetzt noch die IP-Adressen eingetragen werden.

Dies geschieht in dem /etc/dnscache/ Verzeichnis. Unter dem Verzeichnis „env“ in der Datei „IP“ muss die Adresse für den Dnscache angegeben werden. Der Dnscache reagiert nur auf Anfragen auf dieser IP-Adresse. Bei der Erstellung des Verzeichnisses wird die 127.0.0.1 also die lokale Adresse des Computers eingetragen.

Anschließend muss die IP-Adresse im Verzeichnis vom Dnscache im Unterverzeichnis „/root/ip“ als Datei erstellt werden. Für die 127.0.0.1 ist die Adresse bzw. die Datei bereits vorhanden. Dies dient als zusätzliche Sicherheitsfunktion. Falls eine IP-Adresse nicht in diesem Ordner steht ist es nicht möglich, eine Verbindung zu dem Dnscache aufzubauen.

Jetzt muss noch der Server eingetragen werden, zu dem der Dnscache eine Verbindung aufbauen soll. In dem Dnscache Verzeichnis im Unterverzeichnis „/root/servers“ werden die Server festgelegt. In der Datei „@“ befinden sich die Root-Server. Damit die Verschlüsselung des DNSCurve-Patches genutzt werden kann, muss die IP-Adresse des TinyDNS-Servers hier eintragen werden. Es gibt noch eine weitere Möglichkeit Server für den Dnscache hinzuzufügen. Diese wird in dem Kapitel Dnscache erläutert. Warum diese nicht für einen Dnscache mit DNSCurve-Patch funktioniert, wird im Kapitel DJBDNS+IPV6+DNSCurve erklärt.

6.8 Installation von DJBDNS+IPV6+DNSCurve

Die DJBDNS+IPV6+DNSCurve Version liegt derzeit nur als Sourcecode vor. Wenn diese Version genutzt werden soll, muss zunächst die Installation der Programme aus den Unterkapiteln 2.1-2.4 abgeschlossen sein.
Anschließend müssen wie im Unterkapitel 2.7 die Pfadangaben für NaCl in den Dateien „conf-cc“ und „conf-ld“ angepasst werden.
Mit make und ./install wird DJBDNS+IPV6+DNSCurve kompiliert und die Binärdateien werden in den Ordner /usr/local/bin/ verschoben.

Konfiguration von DJBDNS+IPV6+DNSCurve

Setzt die Konfiguration von CurveDNS (beschrieben in Kapitel 6.4) voraus.

Konfiguration für TinyDNS

In der DJBDNS+IPV6+DNSCurve Version können für TinyDNS sowohl Ipv4-Adressen als auch Ipv6-Ardressen genutzt werden. Ipv6-Addressen können in der verkürzten Form eingegeben werden.

```
tinydns-conf tinydns loguser /etc/tinydns 10.0.2.17
```

Dann muss noch ein Soft-Link mit dem Verzeichnis „/service/tinydns“ erstellt werden, damit der TinyDNS-Prozess gesteuert werden kann. In diesem Fall:

```
ln -s /etc/tinydns /service/tinydns
```

Als nächstes müssen in dem Verzeichnis vom TinyDNS (/etc/tinydns) im Unterverzeichnis „root“ in der Datei „data“ die Records eingetragen werden.
Dazu wird für den NS-Record als erstes der FQDN, also die Domain für die der Nameserver zuständig ist, die IPv6-Adresse auf der CurveDNS DNS-Anfragen akzeptiert und der Name des Nameservers, in diesem Fall der öffentliche Schlüssel vom CurveDNS gefolgt von dem FQDN, hinzugefügt. Wie in dem unten folgenden Beispiel zu sehen ist, muss die Ipv6-Addresse ohne Verkürzungen oder Blockbegrenzungen eingegeben werden. Für die Ipv6-Addresse fe80::a00:27ff:fe1b:9804 würde der Eintrag wie folgt lauten:

```
.example.org:20010658000000202e018fffe98b03d:
uz5rxjwlmxlv26gf0u4lnrjbk13qnuxvc72153vrt9mdyp8pgqr0mu.example.org
```

Für die inverse Auflösung wird jedes einzelne Zeichen aus der Ipv6-Adresse gefolgt von einem Punkt angegeben. Bei Ipv6-Adressen wird die Zone für Reverse-Lookup durch ip6.arpa angegeben. Dadurch das Zeichen weggelassen werden, kann die inverse Auflösung für ein Netz eingerichtet werden.

```
.d.3.0.b.8.9.e.f.f.f.8.1.0.e.2.0.2.0.0.0.0.0.0.0.8.5.6.0.1.0.0.2
.ip6.arpa:fe800000000000000a0027fffe1b9804:
uz5rxjwlmxlv26gf0u4lnrjbk13qnuxvc72153vrt9mdyp8pgqr0mu.example.org
```

Jetzt werden noch für dieses Beispiel drei A-Records und ein AAAA-Record mit passendem Reverse-Record angelegt. Diese Records könnten auch mit den Programmen add-host6 hinzugefügt werden. Dieses Programm erstellt bei Ipv6-Adressen automatisch die vollständige Ipv6-Addresse, wenn es mit einer verkürzten Ipv6-Addresse aufgerufen wird.

```
=test1.example.org:10.0.2.18:3600
=test2.example.org:10.0.2.19:3600
=test3.example.org:10.0.2.20:3600
6test6.example.org:fe800000000000000a0027fffe1b9805:3600
```

Anschließend muss noch die Textdatei „data“ in eine .cbp Datei umgewandelt werden. Dazu wird im Verzeichnis /etc/tinydns/root/ das make-Skript ausgeführt.

```
make
```

Konfiguration für Dnscache

Zunächst muss der Dnscache mit Angabe der Benutzerkennung für den Dnscache, der Benutzerkennung für das Log und dem Verzeichnis in dem sich die Konfigurationsdateien vom Dnscache befinden, aufgerufen werden.

```
dnscache-conf dnscache dnslog /etc/dnscache
```

Es muss auch hier ein Soft-Link erstellt werden.

```
ln -s /etc/dnscache /service/dnscache
```

Die IP-Adresse auf der Dnscache in dieser Version DNS-Anfragen akzeptiert, muss eine Ipv6-Addresse sein (zum Beispiel ::1 als lokale Adresse). Ipv4-Adressen werden, sobald der Ipv6-Patch installiert ist, nicht mehr akzeptiert. Zunächst muss die Ipv6-Addresse im Konfigurationsverzeichnis im Unterverzeichnis „/env“ in der Datei „IP“ eingetragen werden.

Für den Dnscache muss im bei der Konfiguration erstellten Verzeichnis im Unterverzeichnis „/root/servers“ in der Datei „@“ die Ipv6-Adresse hinzugefügt werden. Diese sollte am besten am Anfang der Liste von IP-Adressen hinzugefügt werden.

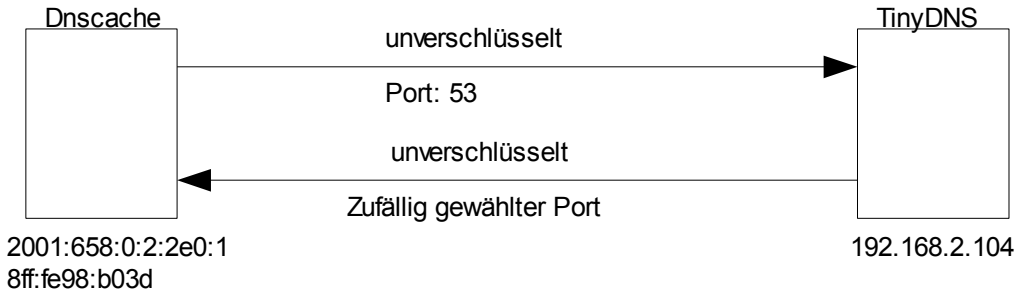
Anschließend muss die IP-Adresse im Verzeichnis vom Dnscache im unter Verzeichnis „/root/ip“ als Datei erstellt werden. Für die ::1 Adresse muss eine Datei mit dem Namen „::1“ ohne Inhalt erstellt werden. Für das Beispiel wird eine Ipv6-Adresse mit:

```
touch ::1
```

7. Aufbau der Kommunikation von DJBDNS+IPV6+DNSCurve

Der folgende Abschnitt erläutert den Ablauf der Kommunikation zwischen Dnscache (DJBDNS+IPV6+DNSCurve Version), TinyDNS (DJBDNS+IPV6+DNSCurve Version) als DNS-Server und CurveDNS als Forwarder.

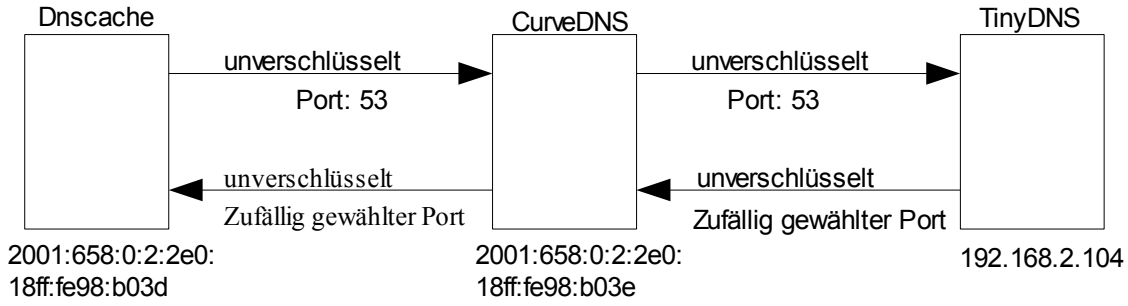
Die Abbildung zeigt den herkömmlichen Ablauf einer Kommunikation zwischen Dnscache mit der IP-Adresse 2001:658:0:2:2e0:18ff:fe98:b03d und TinyDNS mit der IP-Adresse 192.168.2.104. Es könnte für TinyDNS auch eine Ipv6-Adresse genutzt werden. Die DNS-Pakete werden unverschlüsselt übertragen. Für die DNS-Anfrage wird der Port 53 verwendetet. Bei der DNS-Antwort wird ein zufällig gewählter Port verwendet.



Zeichnung 2: Struktur der DNS-Kommunikation

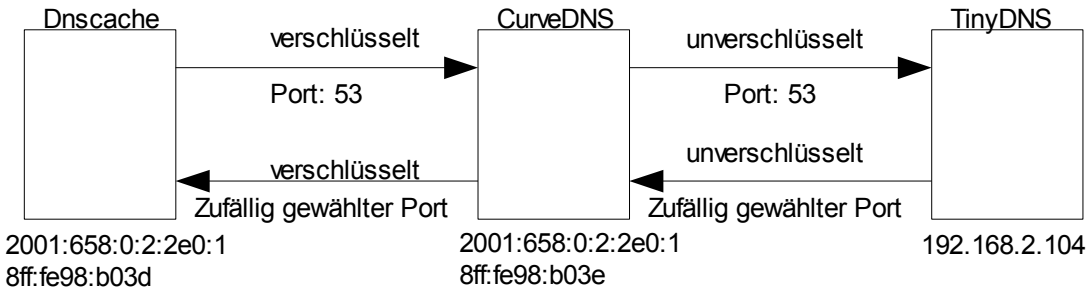
Die nächste Abbildung zeigt die grobe Struktur des Kommunikationsablaufes wenn CurveDNS dem Aufbau hinzugefügt wird und es sich nicht um DNSCurve verschlüsselte Pakete handelt. Der Dnscache kommuniziert mit dem CurveDNS und benutzt dabei die IPv6-Adresse 2001:658:0:2:2e0:18ff:fe98:b03e sowie die Ipv4-Adresse 192.168.2.105. CurveDNS leitet die DNS-Anfrage anschließend an den TinyDNS Server weiter.

Zunächst sendet der Dnscache eine unverschlüsselte Anfrage an den CurveDNS Forwarder. Dieser erkennt, dass die Anfrage kein DNSCurve-Paket ist und leitet sie unverschlüsselt an den TinyDNS Server weiter. Dieser beantwortet die Anfrage mit den Informationen über den Zuständigen Name-Server.



Zeichnung 3: Erweitung der Struktur mit CurveDNS

Wie in der nächsten Abbildung zu sehen ist, wird eine Anfrage jetzt von dem Dnscache verschlüsselt und an den CurveDNS-Forwarder gesendet. Dieser erkennt, dass das DNS-Paket mit DNSCurve verschlüsselt ist und entschlüsselt es. Die entschlüsselte Anfrage wird an den TinyDNS-Server gesendet und von diesem beantwortet. Die Antwort wird von dem CurveDNS-Forwarder wieder verschlüsselt und an den Dnscache gesendet, wo die Antwort wieder entschlüsselt wird.



Zeichnung 4: Struktur der DNS-Kommunikation Verschlüsselung nur zwischen Dnscache und CurveDNS

Wichtig ist hierbei zu beachten das die IP-Adressen von allen DNS-Anfragen die bei dem TinyDNS-Server eingehen die Absende-Adresse vom CurveDNS haben.

Im nächsten Kapitel wird die Kommunikation anhand der gesendeten DNS-Pakete noch genauer erklärt.

7.1 Ablauf der Kommunikation in Paketen

In diesem Abschnitt wird die Kommunikation zwischen Resolver (2001:658:0:2:2e0:18ff:fe98:b03b), Dnscache (2001:658:0:2:2e0:18ff:fe98:b03d), CurveDNS (2001:658:0:2:2e0:18ff:fe98:b03e bzw. 192.168.2.105) und TinyDNS (192.168.2.104) noch einmal genauer anhand der gesendeten DNS-Pakete beschrieben. Es werden insbesondere der Ablauf des ersten und zweiten Durchlaufs im Detail dargestellt. Der Kommunikationsablauf wurde mit Wireshark einem Programm zur Analyse der Netzwerkkommunikation aufgezeichnet. Die zugehörigen Screenshots des Kommunikationsablaufes befinden sich jeweils unter dem erklärenden Text. Mit einem Durchlauf ist die komplette Kommunikation zwischen den drei Programmen gemeint.

1. Durchlauf (Aufruf dig test2.example.org)

Der erste Screenshot zeigt die Kommunikation zwischen dem Resolver des Betriebssystems und dem Dnscache. An den Dnscache wird eine DNS-Anfrage nach test2.example.org gestartet.

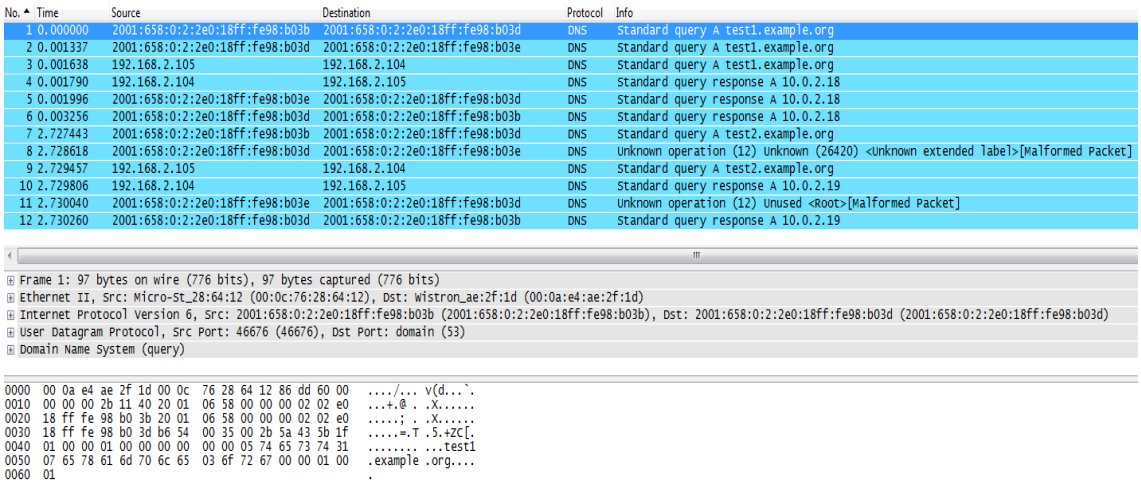


Abbildung 11: DNS-Anfrage vom Resolver an den Dnscache

Der Dnscache stellt fest, dass er diesen Eintrag noch nicht im Cache hat und dass sich auch kein Name-Server für diese Domain im Cache befindet. Der Dnscache sucht im Ordner „servers“ nach Dateien für die Domains example.org oder org. Wenn er keinen Eintrag findet, werden die Root-Server aus der Datei „@“ geladen. Der Dnscache übermittelt die DNS-Anfrage an CurveDNS.

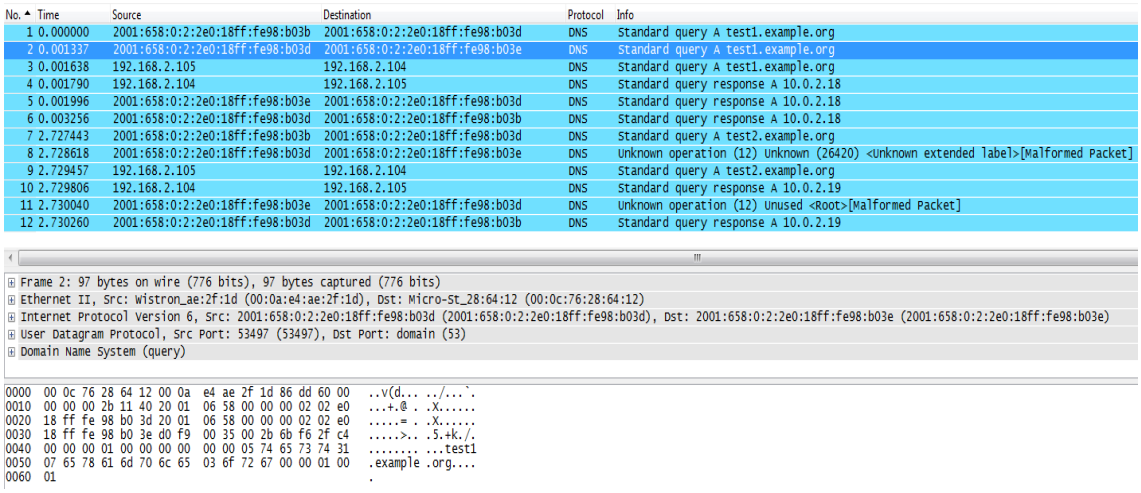


Abbildung 12: DNS-Anfrage vom Dnscache an CurveDNS

CurveDNS stellt fest das es sich bei dem DNS-Paket aufgrund der zu geringen Größe und der fehlenden Identifikation nicht um ein mit DNSCurve verschlüsseltes Paket handelt und sendet es weiter an TinyDNS.

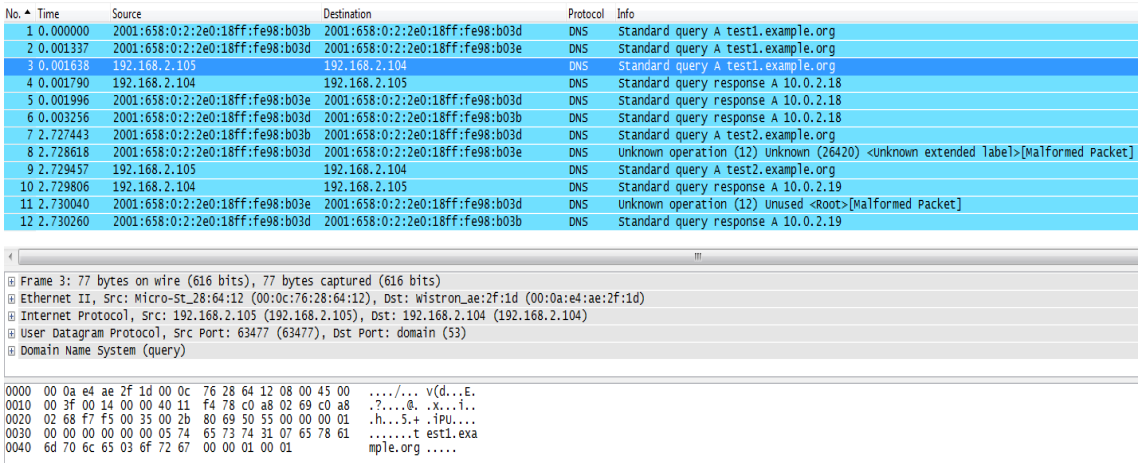
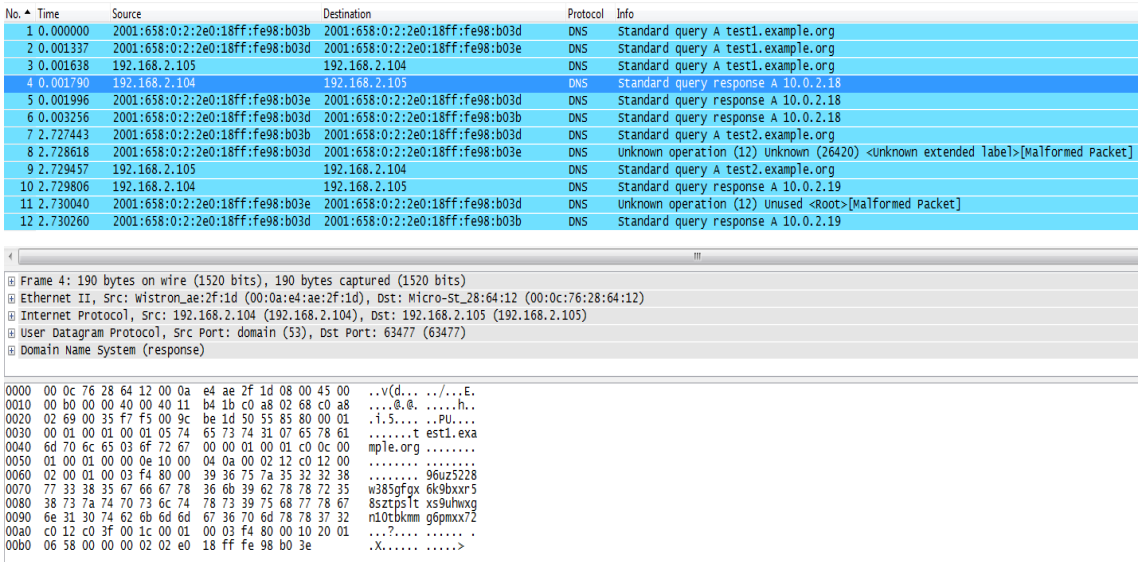


Abbildung 13: DNS-Anfrage vom CurveDNS an TinyDNS

TinyDNS durchsucht die Einträge in der Datei „data.cdb“ und liefert als Antwort die IP-Adresse der DNS-Anfrage in diesem Beispiel 10.0.2.19 und den Namen des Nameservers, wie in diesem Fall uz5228w385gfgx6k9bxxr58sztpsltxs9uhwxgn10tbkmmg6pmxx72.example.org, zurück. Im Namen ist der öffentliche Schlüssel enthalten.



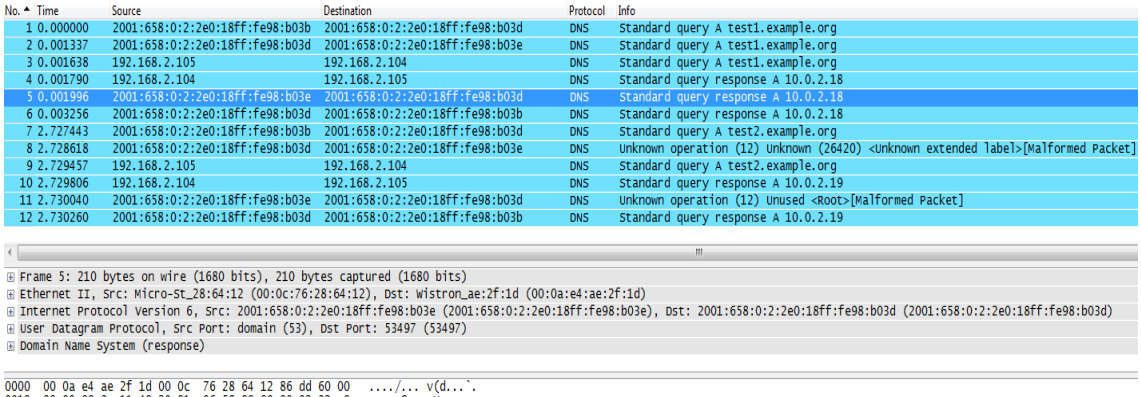
No.	Time	Source	Destination	Protocol	Info
1	0.000000	2001:658:0:2:2e0:18ff:fe98:b03b	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query A test1.example.org
2	0.001337	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03e	DNS	Standard query A test1.example.org
3	0.001638	192.168.2.105	192.168.2.104	DNS	Standard query A test1.example.org
4	0.001790	192.168.2.104	192.168.2.105	DNS	Standard query response A 10.0.2.18
5	0.001996	2001:658:0:2:2e0:18ff:fe98:b03e	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query response A 10.0.2.18
6	0.003256	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03b	DNS	Standard query response A 10.0.2.18
7	2.727443	2001:658:0:2:2e0:18ff:fe98:b03b	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query A test2.example.org
8	2.728618	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03e	DNS	Unknown operation (12) Unknown (26420) <Unknown extended label>[Malformed Packet]
9	2.729457	192.168.2.105	192.168.2.104	DNS	Standard query A test2.example.org
10	2.729806	192.168.2.104	192.168.2.105	DNS	Standard query response A 10.0.2.19
11	2.730040	2001:658:0:2:2e0:18ff:fe98:b03e	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Unknown operation (12) Unused <Root>[Malformed Packet]
12	2.730260	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03b	DNS	Standard query response A 10.0.2.19

```
Frame 4: 190 bytes on wire (1520 bits), 190 bytes captured (1520 bits)
Ethernet II, Src: Wistron_ae:2f:1d (00:0a:e4:ae:2f:1d), Dst: Micro-St_28:64:12 (00:0c:76:28:64:12)
Internet Protocol, Src: 192.168.2.104 (192.168.2.104), Dst: 192.168.2.105 (192.168.2.105)
User Datagram Protocol, Src Port: domain (53), Dst Port: 63477 (63477)
Domain Name System (response)
```

```
0000  00 0c 76 28 64 12 00 0a  e4 ae 2f 1d 08 00 45 00   ..v(d... ../...E.
0010  00 b0 00 00 40 00 40 11  b4 1b c0 a8 02 68 c0 a8   ....@.@. .....h..
0020  02 69 00 35 f7 f5 00 9c  be 1d 50 55 85 80 00 01   .i.5.... ..PU....
0030  00 01 00 01 00 01 05 74  65 73 74 31 07 65 78 61   .......t est1.exa
0040  6d 70 6c 65 03 6f 72 67  00 00 01 00 01 c0 0c 00   mple.org ........
0050  01 00 01 00 00 0e 10 00  04 0a 00 02 12 c0 12 00   ........ ........
0060  02 00 01 00 03 f4 80 00  39 36 75 7a 35 32 32 38   ........ 96uz5228
0070  77 33 38 35 67 66 67 78  36 6b 39 62 78 78 72 35   w385gfgx 6k9bxxr5
0080  38 73 7a 74 70 73 6c 74  78 73 39 75 68 77 78 67   8sztpslt xs9uhwxg
0090  6e 31 30 74 62 6b 6d 6d  67 36 70 6d 78 78 37 32   n10tbkmm g6pmxx72
00a0  c0 12 c0 3f 00 1c 00 01  00 03 f4 80 00 10 20 01   ...?.... ...... .
00b0  06 58 00 00 00 02 02 e0  18 ff fe 98 b0 3e        .X...... .....>
```

Abbildung 14: DNS-Antwort von TinyDNS an CurveDNS

CurveDNS empfängt die Antwort von TinyDNS und überprüft ob es für diese Verbindung einen öffentlichen Schlüssel gespeichert hat. Dies ist hier nicht der Fall, da die DNS-Anfrage nicht mit DNSCurve verschlüsselt war. Also wird das Paket wie im Screenshot zu sehen an den Dnscache weitergeleitet.



No.	Time	Source	Destination	Protocol	Info
1	0.000000	2001:658:0:2:2e0:18ff:fe98:b03b	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query A test1.example.org
2	0.001337	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03e	DNS	Standard query A test1.example.org
3	0.001638	192.168.2.105	192.168.2.104	DNS	Standard query A test1.example.org
4	0.001790	192.168.2.104	192.168.2.105	DNS	Standard query response A 10.0.2.18
5	0.001996	2001:658:0:2:2e0:18ff:fe98:b03e	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query response A 10.0.2.18
6	0.003256	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03b	DNS	Standard query response A 10.0.2.18
7	2.727443	2001:658:0:2:2e0:18ff:fe98:b03b	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Standard query A test2.example.org
8	2.728618	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03e	DNS	Unknown operation (12) Unknown (26420) <Unknown extended label>[Malformed Packet]
9	2.729457	192.168.2.105	192.168.2.104	DNS	Standard query A test2.example.org
10	2.729806	192.168.2.104	192.168.2.105	DNS	Standard query response A 10.0.2.19
11	2.730040	2001:658:0:2:2e0:18ff:fe98:b03e	2001:658:0:2:2e0:18ff:fe98:b03d	DNS	Unknown operation (12) Unused <Root>[Malformed Packet]
12	2.730260	2001:658:0:2:2e0:18ff:fe98:b03d	2001:658:0:2:2e0:18ff:fe98:b03b	DNS	Standard query response A 10.0.2.19

```
Frame 5: 210 bytes on wire (1680 bits), 210 bytes captured (1680 bits)
Ethernet II, Src: Micro-St_28:64:12 (00:0c:76:28:64:12), Dst: Wistron_ae:2f:1d (00:0a:e4:ae:2f:1d)
Internet Protocol Version 6, Src: 2001:658:0:2:2e0:18ff:fe98:b03e (2001:658:0:2:2e0:18ff:fe98:b03e), Dst: 2001:658:0:2:2e0:18ff:fe98:b03d (2001:658:0:2:2e0:18ff:fe98:b03d)
User Datagram Protocol, Src Port: domain (53), Dst Port: 53497 (53497)
Domain Name System (response)
```

```
0000  00 0a e4 ae 2f 1d 00 0c  76 28 64 12 86 dd 60 00   ..../... v(d...`.
0010  00 00 00 9c 11 40 20 01  06 58 00 00 00 00 02 02 e0   .....@ . .X......
0020  18 ff fe 98 b0 3e 20 01  06 58 00 00 00 00 02 02 e0   .....> . .X......
0030  18 ff fe 98 b0 3d 00 35  d0 f9 00 9c a9 aa 2f c4   .....=.5 ....../.
0040  85 80 00 01 00 01 00 01  00 01 05 74 65 73 74 31   ........ ...test1
0050  07 65 78 61 6d 70 6c 65  03 6f 72 67 00 00 01 00   .example .org....
0060  01 c0 0c 00 01 00 01 00  00 0e 10 00 04 0a 00 02   ........ ........
0070  12 c0 12 00 02 00 01 00  03 f4 80 00 39 36 75 7a   ........ ....96uz
0080  35 32 32 38 77 33 38 35  67 66 67 78 36 6b 39 62   5228w385 gfgx6k9b
0090  78 78 72 35 38 73 7a 74  70 73 6c 74 78 73 39 75   xxr58szt pslxs9u
00a0  68 77 78 67 6e 31 30 74  62 6b 6d 6d 67 36 70 6d   hwxgn10t bkmmg6pm
00b0  78 78 37 32 c0 12 c0 3f  00 1c 00 01 00 03 f4 80   xx72...? ........
00c0  00 10 20 01 06 58 00 00  00 02 02 e0 18 ff fe 98   .. ..X.. ........
00d0  b0 3e                                             .>
```

Abbildung 15: DNS-Antwort von CurveDNS zum Dnscache

Der Dnscache empfängt die Antwort auf die Anfrage und schreibt den Namen des Nameservers sowie die dazugehörige IP-Adresse in den Cache. Anschließend wird das Ergebnis der DNS-Anfrage in den Cache geschrieben, dann an das Betriebssystem gesendet und anschließend von diesem ausgegeben. Der Dnscache verkürzt das DNS-Paket und löscht alle Informationen die normalerweise über den Nameserver ausgegeben würden.

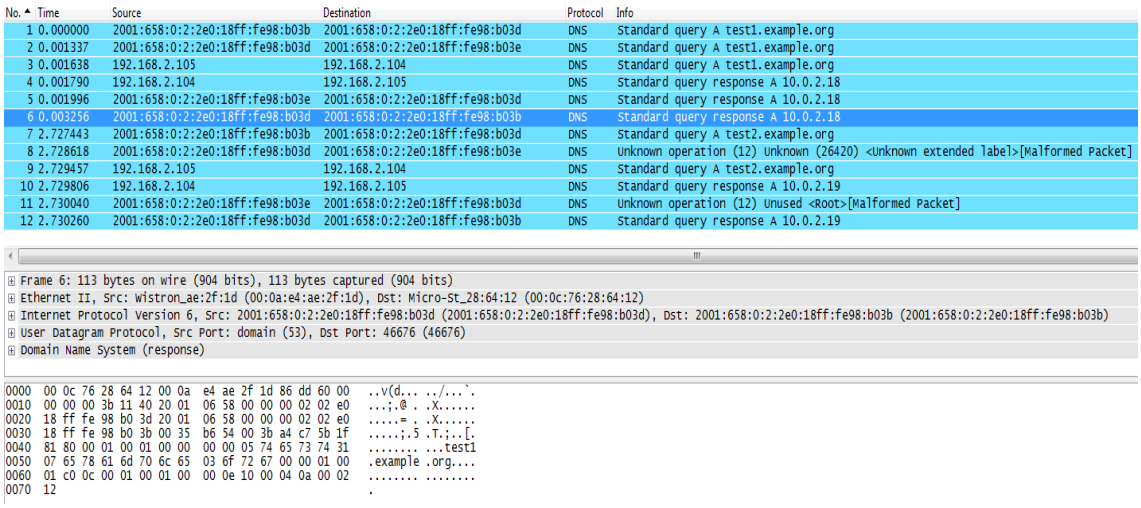


Abbildung 16: DNS-Antwort vom Dnscache an den Resolver

2. Durchlauf (Aufruf dig test1.example.org)

Als erstes wieder die Kommunikation zwischen dem Betriebssystem und dem Dnscache.

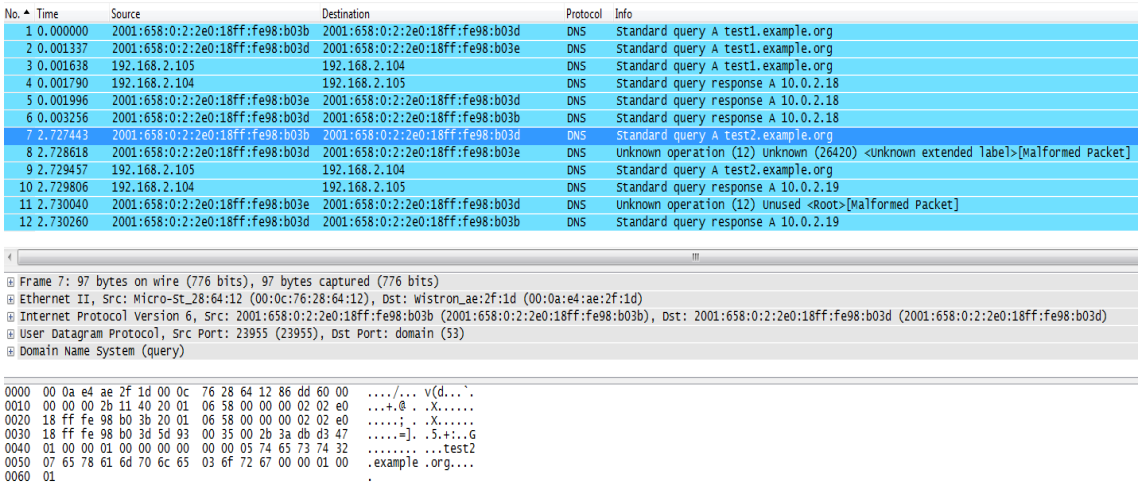


Abbildung 17: DNS-Anfrage vom Resolver an den Dnscache

Der Dnscache erhält die DNS-Anfrage und findet jetzt einen Eintrag für einen Name-Server für example.org im Cache. Mit Hilfe des öffentlichen Schlüssels der sich am Anfang des Name-Servers befindet verschlüsselt der Dnscache die DNS-Anfrage und sendet sie an CurveDNS. Dem DNS-Paket wird noch eine Identifikation (Q6fnvWj8) hinzugefügt um die DNS-Paket als eine mit DNSCurve verschlüsselte DNS-Anfrage auszuzeichnen. Der zuvor noch lesbare Inhalt des DNS-Paketes ist wie im unteren Screenshot zu sehen ist jetzt verschlüsselt. Nur die Identifikation ist in lesbarer Form dargestellt. Es wird auch ein Fehler angezeigt da Wireshark das DNS-Paket nicht interpretieren kann.

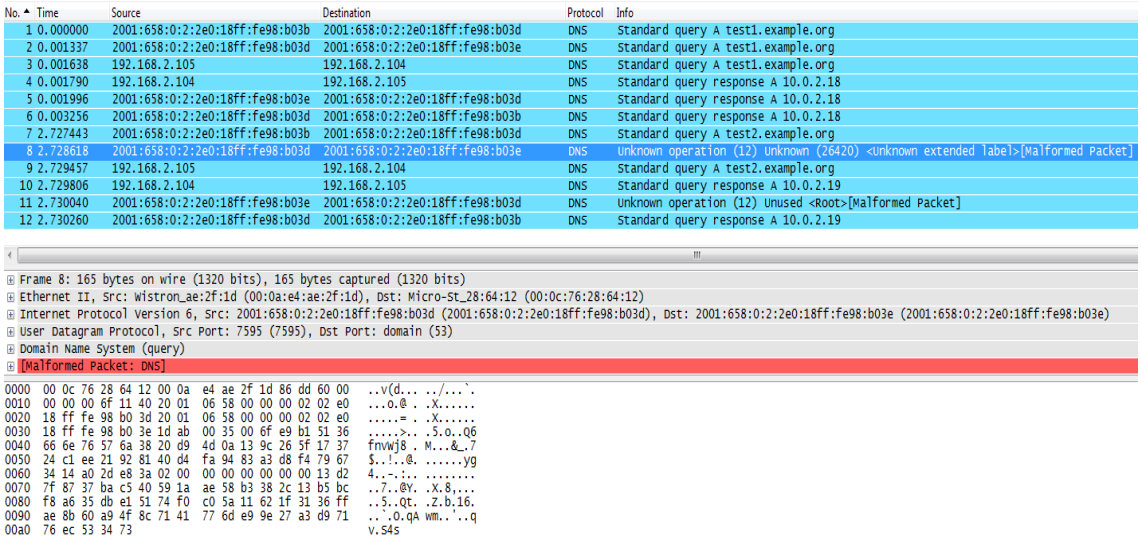


Abbildung 18: DNS-Anfrage vom Dnscache an CurveDNS

CurveDNS erkennt jetzt aufgrund der Größe des DNS-Paketes und der passenden Identifikation das es sich um eine mit DNSCurve verschlüsselte DNS-Anfrage handelt. Das DNS-Paket wird mit Hilfe des privaten Schlüssels von CurveDNS entschlüsselt und der im Paket enthaltene öffentliche Schlüssel des Dnscache wird in den Cache von CurveDNS geladen. Anschließend wird die DNS-Anfrage an TinyDNS weitergeleitet.

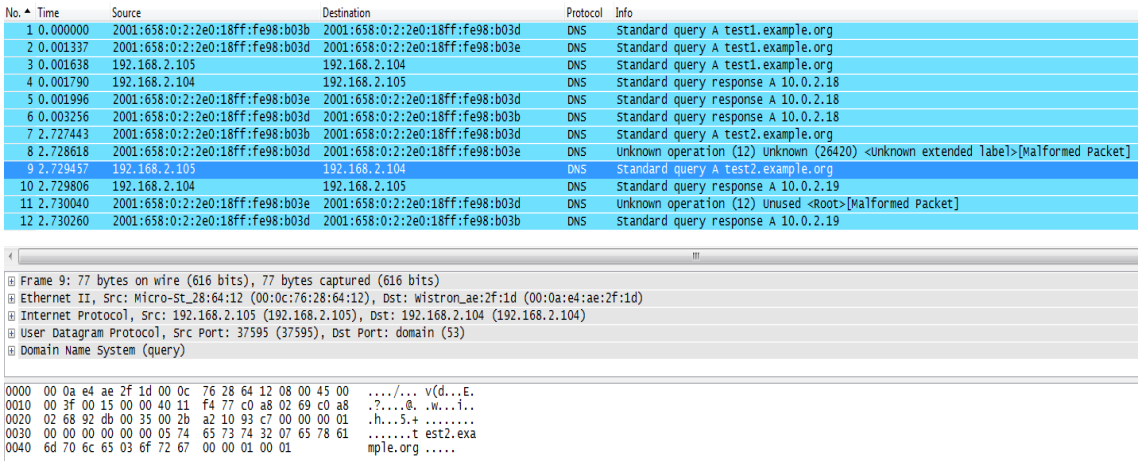


Abbildung 19: DNS-Anfrage von CurveDNS an TinyDNS

TinyDNS durchsucht wieder die Datei „data.cdb“ und liefert die Passende IP-Adresse zu dem Eintrag. Die passende Antwort wird an CurveDNS gesendet.

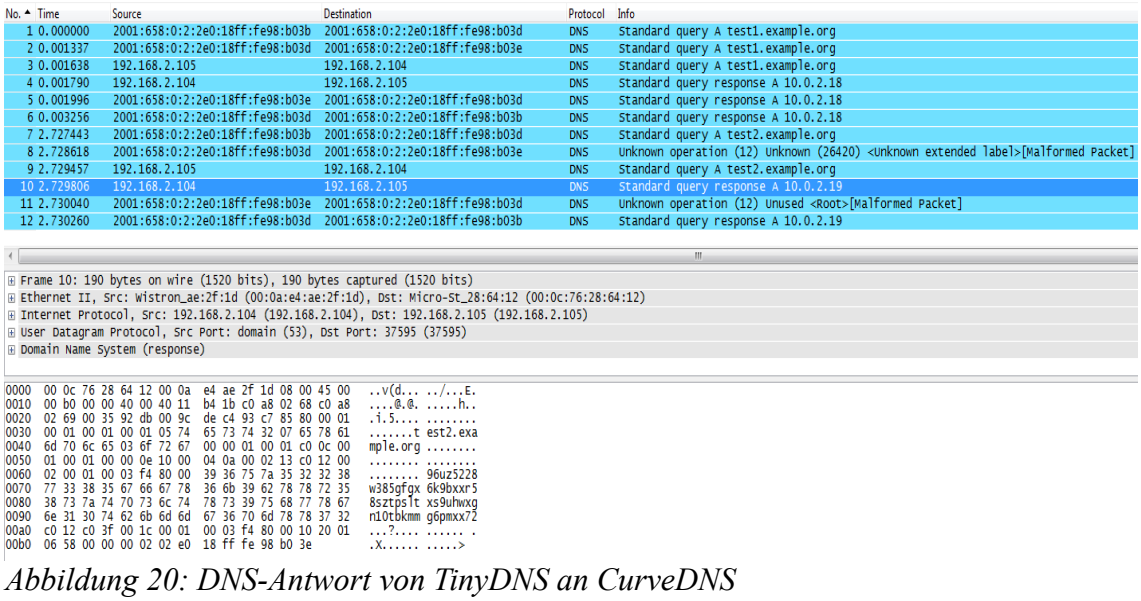


Abbildung 20: DNS-Antwort von TinyDNS an CurveDNS

CurveDNS empfängt die Antwort und verschlüsselt das DNS-Paket mit dem öffentlichen Schlüssel des Dnscache aus seinem Cache. Nach dem das DNS-Paket verschlüsselt ist, wird noch eine Identifikation (R6fnvWJ8) in das DNS-Paket eingefügt um es als eine mit DNSCurve verschlüsselte DNS-Antwort zu kennzeichnen. Außer der Identifikation befindet sich wie im unteren Screenshot zu sehen ist kein Text mehr in lesbarer Form. Wireshark zeigt wieder einen Fehler an, da das Programm das DNS-Paket nicht interpretieren kann.

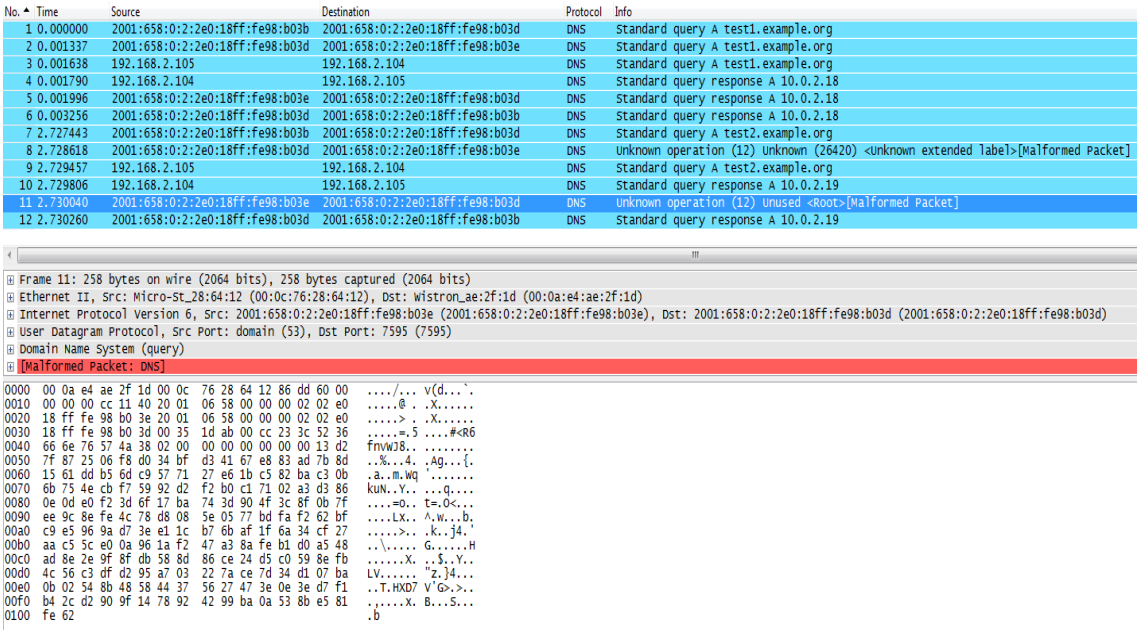


Abbildung 21: DNS-Antwort von CurveDNS an den Dnscache

Der Dnscache entschlüsselt das DNS-Paket mit seinem privaten Schlüssel und leitet das gekürzte DNS-Paket an das Betriebssystem weiter.

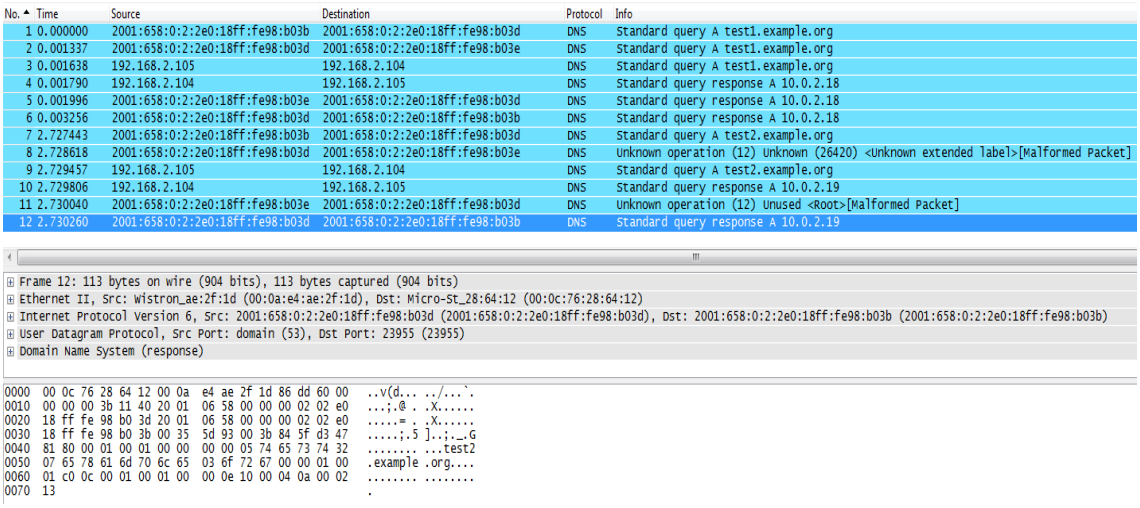


Abbildung 22: DNS-Antwort vom Dnscache an den Resolver

Für alle weiteren Durchläufe gilt, dass sie verschlüsselt werden (wie beim 2. Durchlauf) solange der selbe Name-Server genutzt wird und der Dnscache nicht neu gestartet wird.

7.2 Leistung

In diesem Kapitel wird die Performance von DNSCurve vorgestellt.

Zeitvergleich von unverschlüsselten und verschlüsselten DNS-Paketen.

Die Zeit ist für DNS-Anfragen bzw. Antworten von entscheidender Bedeutung. In diesem Abschnitt wird die Zeit die dafür nötig ist ein DNS-Paket zu verschlüsseln mit der Zeit die eine herkömmliche DNS-Verbindung benötigt verglichen.

Wie anhand der Tabelle zu erkennen ist, benötigt die Verschlüsselung bei dem ermittelten Durchschnitt von 10 Durchläufen länger als die unverschlüsselte Übertragung. Dabei muss allerdings berücksichtigt werden das dieser Test auf einem System mit lediglich 1,7 Gigahertz durchgeführt wurde. Bei schnelleren Systemen dürfe die Zeit für die Verschlüsselung noch einmal sinken.

Durchlauf	unverschlüsselt	verschlüsselt
1	0.001121	0.002574
2	0.003039	0.002924
3	0.001155	0.002878
4	0.001341	0.002761
5	0.001203	0.002776
6	0.003224	0.002686
7	0.001335	0.002653
8	0.001211	0.002694
9	0.001199	0.002649
10	0.001168	0.002641
	0,0015996	0,0027236

Tabelle 9: Die durchschnittlich benötigte Zeit bei 10 Durchläufen

Größenvergleich der unverschlüsselten und verschlüsselten DNS-Pakete

Paketgröße für die Anfragen dig test1.example.org und dig test2.example.org

Ohne DNSCurve Verschlüsselung

DNS-Anfrage (Paket 2 Abbildung 12) = 97 Bytes
DNS-Antwort (Paket 5 Abbildung 15) = 210 Bytes

Mit DNSCurve Verschlüsselung

DNS-Anfrage (Paket 8 Abbildung 18) = 165 Bytes
DNS-Antwort (Paket 11 Abbildung 21) = 258 Bytes

Wie in zusehen ist steigt die Größe der verschlüsselten Pakete an. Diese Pakete wurde im streamline-Format Im Paket 8 befinden sich sowohl der öffentliche Schlüssel vom Dnscache (32 Byte), der Client-Nonce (12 Byte), dem String Q6fnvWj8 (4 Byte) und der kryptographischen Box (Inhalt der DNS-Anfrage in diesem Fall 97 Byte + 20 Byte). Während Paket 11 nur Client-Nonce (12 Byte), Server-Nonce (12 Byte), den String R6fnvWJ8 (4 Byte) und der kryptographischen Box enthält (Inhalt der DNS-Antwort in diesem Fall 210 Byte + 20 Byte).

8. Anwendung von CurveDNS und DNSCurve

Es gibt zur Zeit keine vollständigen Implementierungen, die eine mit DNSCurve geschützte Kommunikation ermöglichen. Als DNS-Forwarder gibt es CurveDNS. Aufgrund der Tatsache das CurveDNS ein eigenständiger DNS-Forwarder ist, ergibt sich ein entscheidender Vorteil. Lediglich für den Rechner, auf dem CurveDNS läuft muss ein auf Unix basierendes Betriebssystem laufen. (zum Beispiel SuseLinux oder FreeBsd). Es ist also theoretisch möglich mit einem Windows-Client Programm, das DNSCurve unterstützt, eine Verbindung zum CurveDNS Server aufzubauen. Es ist ebenfalls möglich, jeden gängigen DNS-Server zu verwenden. Zum Beispiel Bind, TinyDNS oder Microsoft DNS.

Als Clients existieren im Moment nur der Dnscache mit installiertem Dnscurve-Patch oder die DJBDNS+IPV6+DNSCurve Version. Die DJBDNS+IPV6+DNSCurve wurde sowohl auf FreeBSD als auch auf Suse Linux getestet und funktionierte einwandfrei.

Laut einem Eintrag von 2009 auf der Internetseite von DNSCurve soll ein Cache entwickelt werden. [10]

Für Windows sind derzeit keine funktionsfähigen Clients bekannt.

GBDNS, ein von George-Barwood entwickelter DNS-Server für Windows, hat eine Implementierungen von DNSCurve unterstützt. Nach einiger Zeit wurde DNSCurve aber wieder aus dem Funktionsumfang gestrichen. [17]

OpenDNS entwickelt im Moment mit DNSCrypt Programme die auf DNSCurve basieren. Derzeit sind nur ein Client für MAC und ein DNSCrypt Proxy-Server verfügbar. Aber es sollen in nächster Zeit ein Client für Linux sowie ein Client für Windowssysteme folgen. OpenDNS DNS-Server unterstützen mit DNSCurve geschützte Übertragungen. [11][12]

9. Fazit

Das Internet gewinnt immer mehr an Bedeutung. Vor allem auch bei Menschen, die keine tieferen Kenntnisse im IT-Bereich besitzen. Es werden ständig neue Dienste entwickelt, die die Infrastruktur des Internets auch in sensiblen, sicherheitsrelevanten Bereichen (z.B. online-Banking oder soziale Netzwerke) nutzen. Gleichzeitig wächst mit der Verbreitung der Internetanwendung auch der Missbrauch des Internets für kriminelle Zwecke.Wie auch immer wieder in den Medien berichtet wird steigt die Computerkriminalität rasant an. Parallel zur Aufklärungsarbeit ist es erforderlich, die Sicherheit der Internet-Infrastruktur ständig zu verbessern.
Gerade in Fällen wie dem DNS, auf das jeder Internetnutzer angewiesen ist, war/ist die Sicherheit unzureichend.
Natürlich bietet zum Beispiel die DNS-Erweiterung DNSSEC einen gewissen Schutz. Dieser ist allerdings meiner Meinung nach noch nicht ausreichend.
DNSCurve bietet durch die Verschlüsselung der DNS-Pakete auf Basis von elliptischen Kurven eine Vielzahl von Vorteilen, ohne direkte Nachteile zu besitzen. Zwar steigt die Größe der Pakete und es wird eine geringe zusätzliche Zeit für die Verschlüsselung benötigt. Beides befindet sich aber in einem zu vernachlässigendem Bereich. Es sind bisher keine Schwächen in den verwendeten Funktionen bekannt.
Obwohl kürzere Schlüssel eingesetzt werden können, ist die Sicherheit mit dem RSA vergleichbar oder höher. Durch die hyperelliptischen Kurven wird wahrscheinlich auch in Zukunft eine gute Grundlage für asymmetrische Verschlüsselungsverfahren vorhanden sein.
Die NaCl Bibliothek bietet dazu eine Vielzahl von Funktionen. Ob sich Neuentwicklungen wie das CurveCP Protokoll durchsetzten, bleibt abzuwarten.

Bedingt durch die steigende Zahl der Internetnutzer, kam es zu einem Mangel an Ipv4-Adressen.Deshalb wurde es nötig, das Internet-Protokoll Version 6 (Ipv6) als standardisiertes Verfahren einzuführen.
Eine Implementierung von DNSCurve unter DJBDNS gibt es bisher nur für das Internetprotokoll Version 4 (IPv4). Auf Grund der vorher geschilderten Bedeutung von DNSCurve und der Erweiterung von Ipv4 auf Ipv6 war es zur Verbesserung des Schutzes erforderlich, eine Implementierung von DNSCurve zu erzeugen, die kompatibel zu Ipv6-Addressen ist. Gleichzeitig wird damit erreicht, dass DNSCurve aktuell und einsatzfähig bleibt. Dies wurde mit der DJBDNS+IPV6+DNSCurve Version umgesetzt. Mit der DJBDNS+IPV6+DNSCurve Version ist es auch nach einer flächendeckenden Umstellung von Ipv4 auf Ipv6 möglich, eine mit DNSCurve geschützte Kommunikation zu ermöglichen.

10. Literaturverzeichnis

[1] Cryptography in NaCl
http://cr.yp.to/highspeed/naclcrypto-20090310.pdf
Zugriffsdatum: 20.12.2011

[2] Cryptool
http://www.cryptool.org/download/CrypToolScript-de.pdf
Zugriffsdatum: 20.12.2011

[3] Curve25519: new Diffie-Hellman speed records
http://cr.yp.to/ecdh/curve25519-20060209.pdf
Zugriffsdatum: 20.12.2011

[4] CurveDNS A DNSCurve Forwarding Name Server
http://curvedns.on2it.net/docs
Zugriffsdatum: 20.12.2011

[5] Daemontools
http://www.fehcom.de/qmail/docu/05_new.pdf
Zugriffsdatum: 20.12.2011

[6] daemontools
http://cr.yp.to/daemontools.html
Zugriffsdatum: 20.12.2011

[7] djbdns
http://cr.yp.to/djbdns.html
Zugriffsdatum: 20.12.2011

[8] DNS/DHCP Grundlagen und Praxis
Konstantinos Agouros
ISBN 978-3-937514-35-2

[9] DNS Cache Poisoning -The Next Generation
http://www.secureworks.com/research/articles/other_articles/dns-cache-poisoning/
Zugriffsdatum: 20.12.2011

[10] DNSCurve: Usable security for DNS
http://dnscurve.org/
Zugriffsdatum: 20.12.2011

[11] DNSCrypt soll das Domain Name System schützen
http://www.heise.de/ix/meldung/DNSCrypt-soll-das-Domain-Name-System-schuetzen-1392786.html
Zugriffsdatum: 20.12.2011

[12] DNSCrypt – Critical, fundamental, and about time
http://blog.opendns.com/2011/12/06/dnscrypt-%E2%80%93-critical-fundamental-and-about-time/
Zugriffsdatum: 20.12.2011

[13] Elliptische-Kurven-Kryptografie
http://www.itwissen.info/definition/lexikon/elliptic-curve-cryptosystem-ECC-Elliptische-Kurven-Kryptografie.html
Zugriffsdatum: 20.12.2011

[14] Elliptische Kurven in der Kryptographie
Annette Werner
ISBN 3-540-42518-7

[15] Erfolgreiche Timing-Angriffe auf Verschlüsselung mit elliptischen Kurven
http://www.heise.de/newsticker/meldung/Erfolgreiche-Timing-Angriffe-auf-Verschluesselung-mit-elliptischen-Kurven-1247697.html
Zugriffsdatum: 20.12.2011

[16] Google mit besserer Langzeit-Verschlüsselung
http://www.heise.de/newsticker/meldung/Google-mit-besserer-Langzeit-Verschluesselung-1384441.html
Zugriffsdatum: 20.12.2011

[17] GbDns : A recursive caching DNS server for Windows
http://www.george-barwood.pwp.blueyonder.co.uk/DnsServer/changes.htm
Zugriffsdatum: 20.12.2011

[18] Hyperelliptic Curves Allowing Fast Arithmetic
http://www.ruhr-uni-bochum.de/itsc/tanja/KoblitzC.html
Zugriffsdatum: 20.12.2011

[19] Hyperelliptische Kurve
http://de.wikipedia.org/wiki/Hyperelliptische_Kurve
Zugriffsdatum: 20.12.2011

[20] Improved Networking and Cryptography Library
http://www.cace-project.eu/downloads/deliverables-y3/33_CACE_D2.5.pdf
Zugriffsdatum: 20.12.2011

[21] Ipv6-Patch
http://www.fefe.de/dns/
Zugriffsdatum: 20.12.2011

[22] Mooresches Gesetz
http://de.wikipedia.org/wiki/Mooresches_Gesetz
Zugriffsdatum: 20.12.2011

[23] NaCl: Networking and Cryptography library
http://nacl.cace-project.eu/
Zugriffsdatum: 20.12.2011

[24] Remote Timing Attacks are Still Practical
http://eprint.iacr.org/2011/232.pdf
Zugriffsdatum: 20.12.2011

[25] RFC 1035
http://tools.ietf.org/html/rfc1035
Zugriffsdatum: 20.12.2011

[26] Sonderadressen für den Übergang von IPv4 nach IPv6
http://www.rdfnuernberg.de/iav1a/theorie/adresskonzepte3_2.html
Zugriffsdatum: 20.12.2011

[27] Skript der Vorlesung IT-Security
Prof. Dr. Hoffmann
Fachhochschule Frankfurt am Main

[28] Technik der IP-Netze Funktionsweise, Protokolle und Dienste
Anatol Badach, Erwin Hoffmann.
ISBN 978-3-446-21935-9 2.Auflage

[29] ucspi-tcp
http://cr.yp.to/ucspi-tcp.html
Zugriffsdatum: 20.12.2011

[30] Verschlüsselung mit Elliptischen Kurven angekratzt
http://www.heise.de/newsticker/meldung/Verschluesselung-mit-Elliptischen-Kurven-angekratzt-6451.html
Zugriffsdatum: 20.12.2011

[31] Vorschlag zur Rundum-Verschlüsselung des Datenverkehrs
http://www.heise.de/newsticker/meldung/27C3-Vorschlag-zur-Rundum-Verschluesselung-des-Datenverkehrs-1161545.html
Zugriffsdatum: 20.12.2011

[32] Wie sicher ist Verschlüsselung mit elliptischen Kurven?
http://www.scienceblogs.de/mathlog/2009/03/kryptographie-xiii.php
Zugriffsdatum: 20.12.2011

[33] Zur Effizenz von Elliptische Kurven Kryptographie
http://tuprints.ulb.tu-darmstadt.de/391/1/BirgitHenhaplDiss.pdf
Zugriffsdatum: 20.12.2011

Anhang

Inhalt der CD:

Im Ordner djbdnscurveip6

Befindet sich der angepasste Sourcecode der DJBDNS+IPV6+DNSCurve Version

Im Ordner Programme:

Daemontools	daemontools-0.76.tar.gz
DJBDNS	djbdns-1.05.tar.gz
ucspi-tcp	ucspi-tcp-0.88.tar.gz
libev	libev-4.04.tar.tar
NaCl	nacl-20110221.tar.bz2
CurveDNS	curvedns-0.87.tar.gz
DJBDNS Ipv6-Patch	djbdns-1.05-test25.diff.bz2
DJBDNS DNSCurve-Patch	djbdns-dnscurve-20090602.patch

Sowie die Bachelorarbeit als PDF Dokument und die 10 mit Wireshark aufgenommenen Durchläufe .

Eidesstattliche Versicherung

Ich versichere hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist auch noch nicht veröffentlicht.

Frankfurt am Main 30.12.2011