DNSSEC
Domain Name System Security Extensions

ARIN+NANOG On The Road, Waterloo
13 September 2016

; Chase successful
DNSSEC Introduction

How much trust do we put in the Internet?

13.5% of all purchases were done over the internet in 2010, according to BCG, and this is projected to rise to 23% by 2016.  


How much of that trust relies on DNS?

If DNS were to become unreliable or untrustworthy, what would the result be?
DNSSEC Introduction

In the simplest terms:

**DNSSEC** provides digital signatures that allow validating clients to prove that DNS data was not modified in transit.
DNSSEC Introduction

Sources of DNS data generate signatures for data that they are authoritative for.

Recursive servers check the signatures for correctness and signal to their clients the results of those checks.

If data is provably good, the AD (Authenticated Data) bit may be set in response headers.

If queried data is unable to be validated, yet is signaled to be signed, SERVFAIL responses are generated.
Background Knowledge

Before delving into DNSSEC

DNS resolution mechanics

The Delegation Chain

Some Cryptography Fundamentals

Digital Signatures
DNS Resolution

Resolution is the process of obtaining *answers* from the DNS database in response to queries.

Answers are provided by *authoritative* servers.

are cached by both *recursive* servers and *clients*.
DNS Resolution

Resolution is the process of obtaining answers from the DNS database in response to *queries*

Queries

- originate within applications
- are handled on clients by *stub* resolvers
- are sent to and processed by *recursive* servers
DNS Resolution

What is the address of www.example.com?

Local caching DNS server

www.example.com?
At this point, the local server knows nothing except the addresses of the root servers from "root hints".

Do I have the address of www.example.com in cache?

Local caching DNS server

www.example.com?
DNS Resolution

What is the address of www.example.com?

Local caching DNS server

.(root)
DNS Resolution

That record isn't in my list of "known zones", but it is closest to com.

Local caching DNS server

www.example.com?
DNS Resolution

Here's a list of the .com. name servers

Local caching DNS server

.(root)

www.example.com?
DNS Resolution

What is the address of www.example.com?

Local caching DNS server
DNS Resolution

Here's a list of the example.com name servers.

Local caching DNS server

www.example.com?
DNS Resolution

What is the address of www.example.com?

Local caching DNS server

www.example.com?

.(root)

google.com.

google.com.
Here is the address of www.example.com.

Local caching DNS server

www.example.com?
DNS Resolution

Here is the address of www.example.com.

Local caching DNS server
DNS Data Flow Vulnerabilities

Cache Poisoning

What if someone were able to insert data into a server’s cache
That information would be returned to clients instead of "real" data
DNS Data Flow Vulnerabilities

Servers can send irrelevant information in the Additional Section

By definition, the additional section should contain answers to questions that have yet to be asked
DNS Data Flow Vulnerabilities

Header
Question
Answer
Authority
Additional

www.isc.org. A ?

www.isc.org. IN A 204.152.184.88

www.bank.com. IN A 204.152.184.88
DNS Data Flow Vulnerabilities

Cache Poisoning

DNS uses UDP by default

Sender can fabricate anything in the packet
   including source address
DNS Data Flow Vulnerabilities

If I know a question that is about to be asked

I can flood responses containing my data, but a legitimate source
Background Knowledge

Before delving into DNSSEC

- DNS resolution mechanics
- The Delegation Chain

Some Cryptography Fundamentals

- Digital Signatures
Cryptographic Fundamentals

Cryptography has four purposes:

- **Confidentiality**: Keeping data secret
- **Integrity**: Is it "as sent"?
- **Authenticity**: Did it come from the right place?
- **Non-Repudiation**: Don’t tell me you didn’t say that.
Cryptographic Fundamentals

DNSSEC uses cryptography for two purposes:

- **Confidentiality**: Keeping data secret
- **Integrity**: Is it "as sent"?
- **Authenticity**: Did it come from the right place?
- **Non-Repudiation**: Don’t tell me you didn’t say that.
Cryptography for DNS admins

To provide Authenticity and Integrity, we use:

- Asymmetric Cryptography
- Digital Signatures
Asymmetric Cryptography

Keypairs – Public and Private Key Portions

Data encrypted with one piece of a key can be decrypted or checked for integrity with the other

It is unlikely that a person holding the public key will be able to reverse engineer the private key
Asymmetric Cryptography

Data that can be decrypted is guaranteed to have been unaltered since encryption

**Integrity**

Since the data was decrypted with a public portion of a known key pair, the private portion must have been the one to encrypt the data

**Authenticity**
Digital Signatures

Since we don't care about encrypting the entire content of the message...

Create a hash of the data to be sent, encrypt the hash with our private key and transmit it with the message

Anyone holding public key can authenticate and confirm integrity of the message

Anyone without the public key can still see the data
Digital Signatures in DNSSEC

If the two hashes match we know that the DNS data has not been modified in transit, and that it was created by the owner of $K_1$. 

- Hashing the DNS data
- Encrypting the hash with $K_1$
- Transmitting the DNS data
- Hashing the received data
- Decrypting the signature with $K_2$
- If the two hashes match
Digital Signatures for those that don't care

If the client does not care about, or is not able to do the math required for validation, the signature can be ignored.
Deploying DNSSEC Zone

Administrative Decisions
Administrative Decisions about DNSSEC

There are decisions that need to be made prior to deployment:

- What algorithm will be used?

- What bit-length for keying material?

- NSEC or NSEC3 for proof of non-existence?

- Two keys per zone? Yes, a Key-Signing Key (KSK) & a Zone-Signing Key (ZSK).
What Algorithm Should Be Used?

Choice of algorithm depends on a number of criteria:

- Interoperability with "legacy" systems
  - Requires use of RSASHA1 algorithm
- Legality issues
  - GOST vs. RSA
- Wide spread ability to validate chosen algorithm
<table>
<thead>
<tr>
<th>ALG#</th>
<th>Name</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSA/MD5</td>
<td>Deprecated</td>
</tr>
<tr>
<td>3</td>
<td>DSA/SHA1</td>
<td>DSA</td>
</tr>
<tr>
<td>5</td>
<td><strong>RSA/SHA-1</strong></td>
<td><strong>RSASHA1</strong></td>
</tr>
<tr>
<td>6</td>
<td>DSA-NSEC3-SHA1</td>
<td>NSEC3DSA</td>
</tr>
<tr>
<td>7</td>
<td>RSASHA1-NSEC3-SHA1</td>
<td>NSEC3RSASHA1</td>
</tr>
<tr>
<td>8</td>
<td><strong>RSA/SHA-256</strong></td>
<td><strong>RSASHA256</strong></td>
</tr>
<tr>
<td>10</td>
<td>RSA/SHA-512</td>
<td>RSASHA512</td>
</tr>
<tr>
<td>12</td>
<td>GOST R 34.10-2001</td>
<td>ECCGOST</td>
</tr>
<tr>
<td>13</td>
<td>ECDSA Curve P-256 w/ SHA-256</td>
<td>ECDSAP256SHA256</td>
</tr>
<tr>
<td>14</td>
<td>ECDSA Curve P-384 with SHA-384</td>
<td>ECDSAP384SHA384</td>
</tr>
</tbody>
</table>
Key Bit Length

The choice of bit-length for keying material is based on the algorithm being used and the purpose of the key.

Algorithm requirements

RSA keys must be between 512 and 2048 bits.

DSA keys must be between 512 and 1024 bits and an exact multiple of 64.

NIST recommends 1024 bit ZSK and 2048 bit KSK.
NSEC vs. NSEC3 denial of existence

The NSEC method of proof-of-nonexistence allows "zone walking", as it proves negative responses by enumerating positive responses.

NSEC3 disallows "zone walking", but it requires additional processing on both authoritative servers providing negative responses and on recursive servers doing validation.

If you disallow zone transfers, you will want to deploy NSEC3.
DS Resource Records - Talking to our Parent…

To create chains of trust "in-protocol," the Key Signing Key of a zone is hashed and that hash is placed into the parent

This record is known as the Delegation Signing (DS) record

The DS record in the parent creates a secure linkage that an external attacker would have to overcome to forge keying material in the child
DNSSEC Trust tree:

www.dnslab.org. (A)
    |---dnslab.org. (DNSKEY keytag: 7308 alg: 8 flags: 256)
    |---dnslab.org. (DNSKEY keytag: 9247 alg: 8 flags: 256)
    |---dnslab.org. (DS keytag: 9247 digest type: 2)
    |---org. (DNSKEY keytag: 24209 alg: 7 flags: 256)
    |---org. (DNSKEY keytag: 9795 alg: 7 flags: 257)
    |---org. (DNSKEY keytag: 21366 alg: 7 flags: 257)
    |---org. (DS keytag: 21366 digest type: 1)
    |   |---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
    |   |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
    |---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
    |---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)

Deploying DNSSEC Zones

Technical Decisions

;; Chase successful
Preparing for DNSSEC Deployment

There are a number of methods of deploying DNSSEC into existing zones:

- Manual zone signing (In 2016, DDT - Don’t Do That!)
- Automatic zone signing of dynamic zones
- Automatic in-line signing "on-box"
- Automatic in-line signing "bump-in-the-wire"
Automatic Zone Signing of Dynamic Zones

BIND 9.7 and newer provide automation of zone signing of dynamic zones

Keying material contains timing "meta-data" that can allow automation of key rollover

Making a zone dynamic is significantly easier in recent versions of BIND

Dynamic zones are not always appropriate or allowed
Automatic In-Line Signing

BIND 9.9 introduced In-Line signing

Signing of zones without knowledge of / changes to existing processes and procedures

On-Box in-line signing DNSSEC signs zones in memory on the same system on which they are mastered

Bump In The Wire signing provides signing on an intermediate system

Use this where existing infrastructure can't be modified
DNSSEC Trust tree:
www.dnslab.org. (A)
|---dnslab.org. (DNSKEY keytag: 7308 alg: 8 flags: 256)
    |---dnslab.org. (DNSKEY keytag: 9247 alg: 8 flags: 256)
    |---dnslab.org. (DS keytag: 9247 digest type: 2)
|---org. (DNSKEY keytag: 24209 alg: 7 flags: 256)
    |---org. (DNSKEY keytag: 9795 alg: 7 flags: 256)
    |---org. (DNSKEY keytag: 21366 alg: 7 flags: 256)
|---org. (DS keytag: 21366 digest type: 1)
    |---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
    |---. (DNSKEY keytag: 19036 alg: 8 flags: 256)
|---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
    |---. (DNSKEY keytag: 19036 alg: 8 flags: 256)

Deploying DNSSEC Zones

Abbreviated Technical Steps

;; Chase successful
DNSSEC Signing - The Short List

Generate keys for zone

Insert public portions of keys Into zone

Sign zone with appropriate keys

Publish signed zone

DS in the parent zone

Validate!
Insert Public Keying Material into Zone

If using in-line signing, inserting keying material into the zone is automatic

```plaintext
zone "dnslab.org" {
    type master;
    file "master/dnslab.org";
    inline-signing yes;
    auto-dnssec maintain;
}
```

In-line signing keeps a separate copy of the zone in memory and adds records to that zone, not modifying the zone on disk
Signing a Zone

dnssec-keygen -K /etc/namedb/keys -a rsasha256 -b 1024 $ZONE

dnssec-keygen -K /etc/namedb/keys -a rsasha256 -b 2048 -f ksk $ZONE

rndc signing -nsec3param 1 0 10 $SALT $ZONE
rndc sign $ZONE
If there is a reason that your provisioning infrastructure can't be touched, consider “bump in the wire” in-line signing…
"Bump In The Wire" In-Line Signing

If there is a reason that your provisioning infrastructure can't be touched, consider “bump in the wire” in-line signing…
"Bump In The Wire" In-Line Signing

zone "dnslab.org" {
    type slave;
    masters { true-master; }
    also-notify { list-of-slaves; }
    file "slave/dnslab.org";
    inline-signing yes;
    auto-dnssec maintain;
};

The master must be modified to only send notifies and allow zone transfers from the signing server

The slave servers must be modified to accept notifies and perform zone transfers from the signing server
"Bump In The Wire" In-Line Signing

In-line signing, automatically inserts keying material into the zone

dnssec-keygen -K ./keys -a rsasha512 -b 1024 dnslab.org

dnssec-keygen -K ./keys -a rsasha512 -b 2048 -f ksk dnslab.org

rndc signing -nsec3param 1 0 10 bad5a170

rndc retransfer dnslab.org

rndc sign dnslab.org
DNSSEC Trust tree:
www.dnslab.org. (A)
|---dnslab.org. (DNSKEY keytag: 7308 alg: 8 flags: 256)
    |---dnslab.org. (DNSKEY keytag: 9247 alg: 8 flags: 257)
    |---dnslab.org. (DS keytag: 9247 digest type: 2)
    |---org. (DNSKEY keytag: 24209 alg: 7 flags: 256)
       |---org. (DNSKEY keytag: 9795 alg: 7 flags: 257)
       |---org. (DNSKEY keytag: 21366 alg: 7 flags: 257)
       |---org. (DS keytag: 21366 digest type: 1)
          |---. (DNSKEY keytag: 33655 alg: 8 flags: 256)
             |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful

Enabling DNSSEC Validation
Validating DNSSEC

Authoritative Servers (master/slave) never do validation nor provide signaling of validation to clients

If a DNS response has the AA (authoritative answer) bit set, it will never have the AD (authenticated data) bit set

It is the job of the recursive (validating) server to do the work required to prove data is unmodified
Validating DNSSEC

To validate DNSSEC, a recursive server must be able to track back to a trust anchor.

Even if there is no trust anchor in place, a server may return signature data to the client in case the client can do validation itself.

DNSSEC data (RRSIGS) are returned if the DO bit is set in the EDNS0 header.

The AD bit is returned if validation to a trust anchor succeeded.
Validating DNSSEC

BIND uses trust anchors from "trusted-keys" statements:

```plaintext
trusted-keys {
    "."  257 3 8 "AwEAA[...]ihz0=";
};
```

But what happens if the key changes? RFC-5011!

```plaintext
managed-keys {
    "." initial-key 257 3 8 "AwE[..]ihz0=";
};
```
Validating DNSSEC

RFC-5011 covers the problem of validating servers having to be reconfigured when trust-anchor material changes.

If a trust anchor KSK RRSET adds a new key and that key remains published in the zone for 30 days, that key may be considered as a trust anchor for the zone.

If the REVOKE bit is then set in the old KSK, the new KSK should be employed as the new trust-anchor for the zone.
The Root KSK will be rolled! Use managed-keys!

```plaintext
options {
    dnssec-enable yes;
    dnssec-validation yes;
};
managed-keys {
    "." initial-key [.....];
};
```
DS Resource Records

To create chains of trust "in-protocol," the Key Signing Key of a zone is hashed and that hash is placed into the parent

This record is known as the Delegation Signing (DS) record

The DS record in the parent creates a secure linkage that an external attacker would have to overcome to forge keying material in the child
DS Resource Records

The DS record contains:

- The key tag of the key in the child
- The algorithm number of the key
- The hashing algorithm number used to create the DS

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th></th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SHA-1</td>
<td>2</td>
<td>SHA-256</td>
</tr>
<tr>
<td>3</td>
<td>GOST R 34.11-94</td>
<td>4</td>
<td>SHA-384</td>
</tr>
</tbody>
</table>

- The hash of the key
DS Records

dnslab.org.  86400 IN DS 9247 8 2 (F788167DCF705C97D0CB1FD61F7B8EA807E61D8077FA2F50660B871FF9D8DE24)
DS Records

dnslab.org. 86400 IN **DS 9247 8 2** (F788167DCF705C97D0CB1FD61F7B8EA807E61D8077FA2F50660B871FF9D8DE24)
DS Records

dnslab.org. 86400 IN DS 9247 8 2 (F788167DCF705C97D0CB1FD61F7B8EA807E61D8077FA2F50660B871FF9D8DE24)
DS Records

dnslab.org.  86394 IN DS 9247 8 2 (F788167DCF705C97D0CB1FD61F7B8EA807E61D8077FA2F50660B871FF9D8DE24 )
dnslab.org.  86394 IN RRSIG DS 7 2 86400 (20140318154949 20140225144949 24209 org.VWhUKxm+ig+yA/gV5kpEKB/Tb91R7b8dZTjpBtt4ZJFNAI70VFT6w1EL9T1ZGYsOXd8bYB5VQhK6ZOMATIoDS/gG hQKGtC8sJG3I4ktuU/nMnyK/0FBCLnUpcGfk+A0E2ECjGLOLu6N/0cst9UH01+1oh30hMoMQVfpL9UOse+c= )

DS record lives in the parent and is signed with parent ZSK
DS Records

**Parent:**

dnslab.org. 86400 IN DS 9247 8 2 (F788167DCF705C97D0CB1FD61F7B8EA807E61D8077FA2F50660B871FF9D8DE24)

**Child:**

dnslab.org. 3600 IN DNSKEY 257 3 8 (AwEAAaHaqpWsLOXTNKdaYa9kQcK/HTaYYsT05rKzPHsY [...] BFLYBHoDZ6HHf5RmSYWUSXr3YYCpf9DwYnqT6Rc=) ; KSK; alg = RSASHA256; key id = 9247
DNSSEC in the real world

;; Chase successful
DNSSEC in the real world

Sandia National Labs & Verisign provide a web page that performs DNSSEC chain testing

http://www.dnsviz.net
DNSSEC in the real world - what about the clients?

run your own validating resolver… NLNetLab’s dnssec-trigger

do validation in the browser… cz.nic’s DNSSEC Validator for Chrome
More Real-World… Key Rollover Schedule

There is not “one answer” as to how often you should roll your keys.

NIST recommends:

- KSK should be rolled once a year
- ZSK should be rolled every 3 months