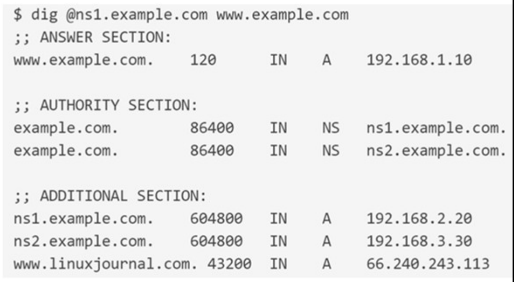
Understanding Kaminsky's DNS Bug [2008]

Dr. Gaurav Varshney, IIT Jammu

- **Bailiwick Checking:** If we ask for information about ftp.example.com then we only accept information in the additional section that is about example.com.
- **DNS Query ID:** A random ID ranging from 0 to 65535 added by the DNS client in query and is expected in reply to match responses.
  - What if attacker guess the ID?
  - What if attacker send random responses with Different query IDs hoping one will match.
  - Attacker can spoof IP…
  - Can cause malicious redirection cache poisoning



```
$ dig @ns1.example.com www.example.com
;; ANSWER SECTION:
www.example.com.    120      IN    A    192.168.1.10

;; AUTHORITY SECTION:
example.com.        86400    IN    NS   ns1.example.com.
example.com.        86400    IN    NS   ns2.example.com.

;; ADDITIONAL SECTION:
ns1.example.com.    604800   IN    A    192.168.2.20
ns2.example.com.    604800   IN    A    192.168.3.30
www.linuxjournal.com. 43200  IN    A    66.240.243.113
```

https://www.linuxjournal.com/content/understanding-kaminskys-dns-bug

Understanding Kaminsky's DNS Bug [2008]

Dr. Gaurav Varshney, IIT Jammu

❖ Authorities must be for queried domain

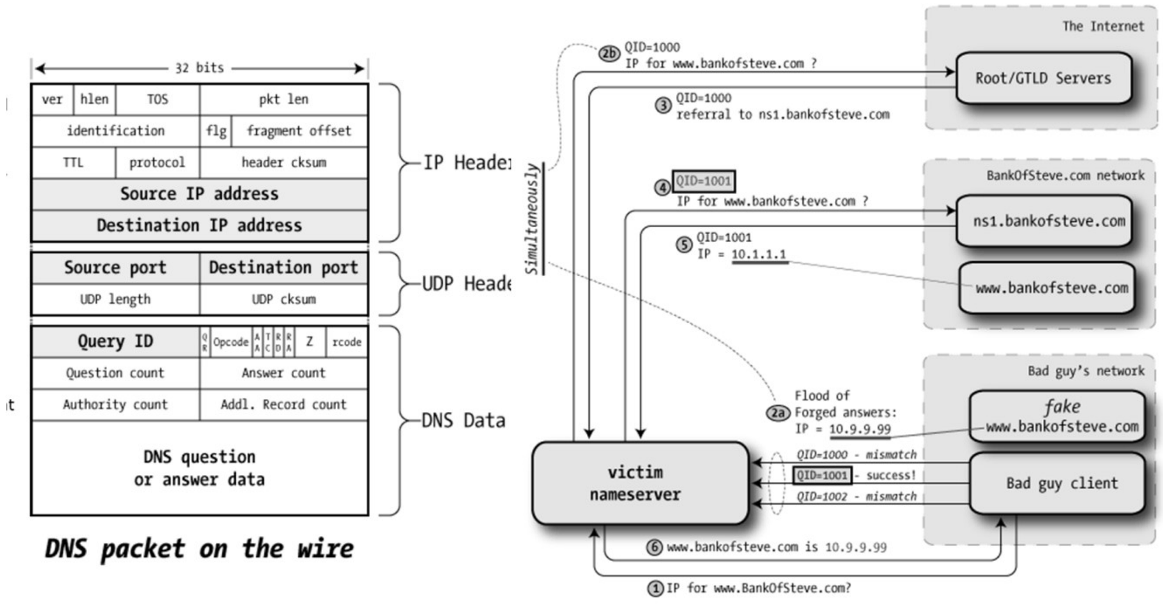
- `ns0.csd.edu` accepted as authority for `ucsd.edu` only when initiating query was for subdomain of `ucsd.edu`

DNS should have been designed with addresses, not names, in NS records and MX records. The "additional section" of DNS responses should have been eliminated. RFC 1035 observes correctly that NS indirection and MX indirection "insure [sic] consistency" of addresses; however, this indirection should have been handled by the server, not the client.

— *Daniel J. Bernstein*

Dr. Gaurav Varshney, IIT Jammu

Understanding Kaminsky's DNS Bug [2008]



Dr. Gaurav Varshney, IIT Jammu

Understanding Kaminsky's DNS Bug [2008]

- The Exploit

```
<img src="http://aaaa.example.com/image.jpg"/>
<img src="http://aaab.example.com/image.jpg"/>
<img src="http://aaac.example.com/image.jpg"/>
```

When a browser tries to render this page it will ask the resolver to look up the address of aaaa.example.com, aaab.example.com, and so on until it has looked up the addresses of all 1000.

If the attacker is constantly sending answers to the resolver with QID 12345, for example, it will eventually have the right QID and the response will be accepted.

But what about bailiwick checking?

he's simply sending his own answers to the resolver, so he can craft the responses to appear to be in bailiwick

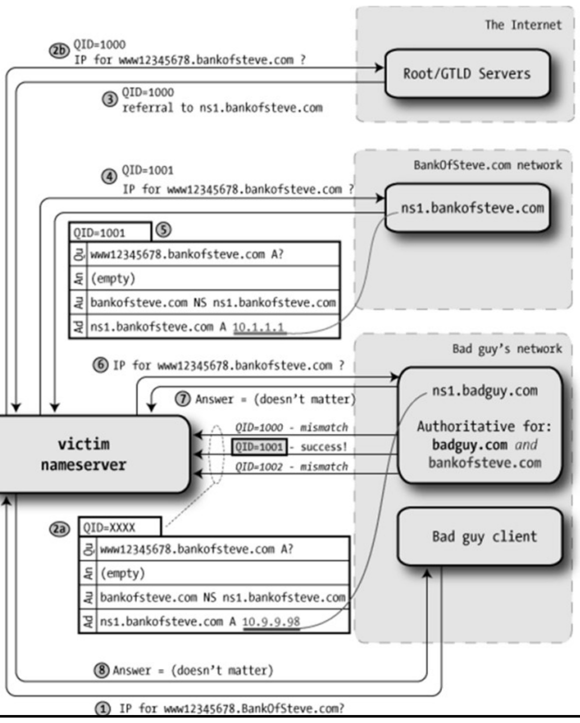
```
;; ANSWER SECTION:
aaaa.example.com.    120    IN    A     10.10.10.10

;; AUTHORITY SECTION:
example.com.        86400   IN    NS    www.example.com.

;; ADDITIONAL SECTION:
www.example.com.    604800  IN    A     10.10.10.20
```

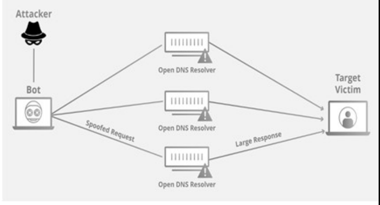
Dr. Gaurav Varshney, IIT Jammu

Understanding Kaminsky's DNS Bug [2008]


The Internet
Root/GTLD Servers
2b QID=1000 IP for www12345678.bankofsteve.com ?
3 QID=1000 referral to ns1.bankofsteve.com
BankOfSteve.com network
ns1.bankofsteve.com
4 QID=1001 IP for www12345678.bankofsteve.com ?
QID=1001 5
Qu www12345678.bankofsteve.com A?
An (empty)
Au bankofsteve.com NS ns1.bankofsteve.com
Ad ns1.bankofsteve.com A 10.1.1.1
Bad guy's network
6 IP for www12345678.bankofsteve.com ?
7 Answer = (doesn't matter)
ns1.badguy.com
Authoritative for: badguy.com and bankofsteve.com
victim nameserver
QID=1000 - mismatch
QID=1001 - success!
QID=1002 - mismatch
Bad guy client
2a QID=XXXX
Qu www12345678.bankofsteve.com A?
An (empty)
Au bankofsteve.com NS ns1.bankofsteve.com
Ad ns1.bankofsteve.com A 10.9.9.98
8 Answer = (doesn't matter)
1 IP for www12345678.BankOfSteve.com?

Dr. Gaurav Varshney, IIT Jammu

https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/



Attacks on DNS

- DNS Reflection Attack: DNS request with spoofed IP address.
- DNS Amplification Attack: User Botnets to launch DNS reflection.
  - The target of both the attacks is on the availability of the victim's network and DNS infrastructure as a whole.
  - Can be solved by DDoS monitoring and detection.
  - Solutions to DNS Reflection/Amplification - https://www.cisa.gov/uscert/ncas/alerts/TA13-088A
  - Source IP Verification, Disabling Recursion on Authoritative NS and Limiting Recursion to Authorized Clients, Response Rate Limiting to a Single Client
- DNS Cache Poisoning:
  - Rouge DNS server can append malicious Domain Name-IP mapping for a domain not in his control to have it cached
    - Bailiwick checking
  - Attacker can send spoofed DNS responses to DNS servers/resolvers in a hope that they will be accepted [if Query ID in response matches]
- DNS Tunneling: Using DNS as a covert communication channel to bypass firewalls.
- NXDOMAIN Attack: Attacker send queries for non existing domains causing recursive DNS server to receive NXDOMAIN records in reply and get its cache filled up.

https://www.cisa.gov/uscert/ncas/alerts/TA13-088A
https://www.infoblox.com/wp-content/uploads/2016/04/infoblox-solution-note-nxdomain-attack-methods-and-mitigation.pdf

Dr. Gaurav Varshney, IIT Jammu

https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en

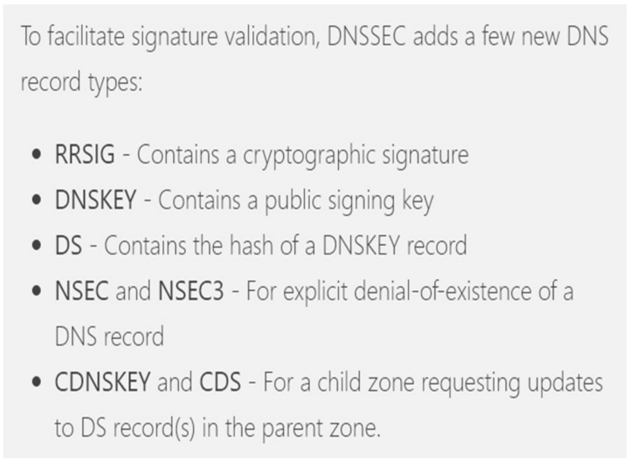
DNSSEC- DNS Security Extensions

- DNSSEC strengthens authentication in DNS using digital signatures based on public key cryptography.
- Every DNS zone has a public/private key pair. The zone owner uses the zone's private key to sign DNS data in the zone and generate digital signatures over that data.
- Any recursive resolver that looks up data in the zone also retrieves the zone's public key and the digital signature records, which it uses to validate the authenticity of the DNS data.
- DNSSEC adds two important features to the DNS protocol:
    - Data origin authentication allows a resolver to cryptographically verify that the data it received actually came from the zone where it believes the data originated.
    - Data integrity protection allows the resolver to know that the data hasn't been modified in transit since it was originally signed by the zone owner with the zone's private key.

Dr. Gaurav Varshney, IIT Jammu

https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

DNSSEC- How it works ?

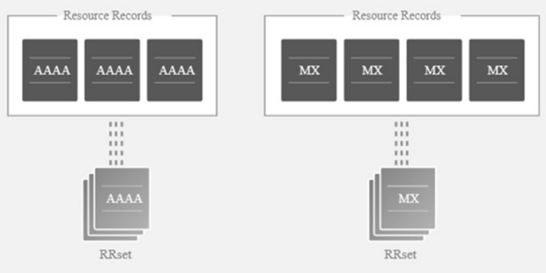


To facilitate signature validation, DNSSEC adds a few new DNS record types:

- **RRSIG** - Contains a cryptographic signature
- **DNSKEY** - Contains a public signing key
- **DS** - Contains the hash of a DNSKEY record
- **NSEC** and **NSEC3** - For explicit denial-of-existence of a DNS record
- **CDNSKEY** and **CDS** - For a child zone requesting updates to DS record(s) in the parent zone.

RRsets

The first step towards securing a zone with DNSSEC is to group all the records with the same type into a resource record set (RRset). For example, if you have three AAAA records in your zone on the same label (i.e. label.example.com), they would all be bundled into a single AAAA RRset.



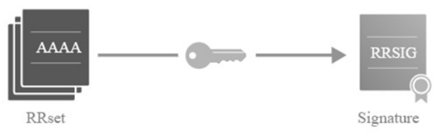
https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

DNSSEC- How it works ?

Dr. Gaurav Varshney, IIT Jammu

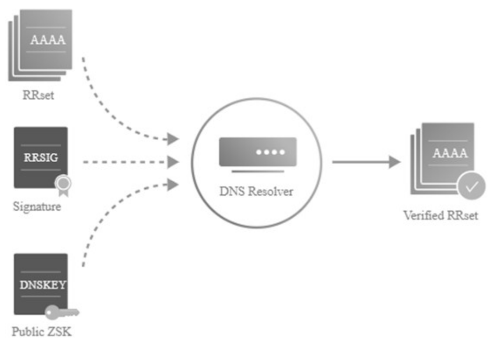
Zone-Signing Keys

Each zone in DNSSEC has a zone-signing key pair (ZSK): the private portion of the key digitally signs each RRset in the zone, while the public portion verifies the signature. To enable DNSSEC, a zone operator creates digital signatures for each RRset using the private ZSK and stores them in their name server as RRSIG records. This is like saying, "These are my DNS records, they come from my server, and they should look like this."



However, these RRSIG records are useless unless DNS resolvers have a way of verifying the signatures. The zone operator also needs to make their public ZSK available by adding it to their name server in a DNSKEY record.

When a DNSSEC resolver requests a particular record type (e.g., AAAA), the name server also returns the corresponding RRSIG. The resolver can then pull the DNSKEY record containing the public ZSK from the name server. Together, the RRset, RRSIG, and public ZSK can validate the response.



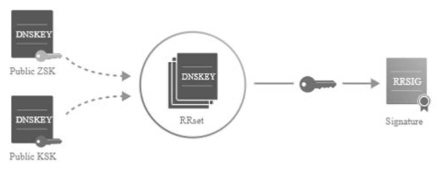
https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

DNSSEC- How it works ?

Dr. Gaurav Varshney, IIT Jammu

Key-Signing Keys

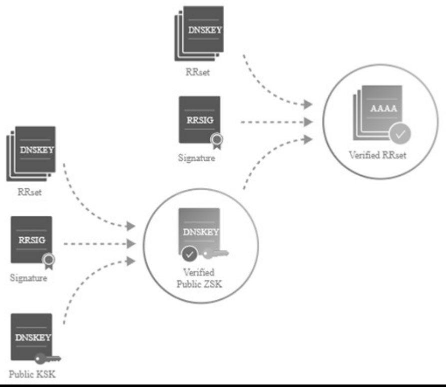
In addition to a zone-signing key, DNSSEC name servers also have a key-signing key (KSK). The KSK validates the DNSKEY record in exactly the same way as our ZSK secured the rest of our RRsets in the previous section: It signs the public ZSK (which is stored in a DNSKEY record), creating an RRSIG for the DNSKEY.



Just like the public ZSK, the name server publishes the public KSK in another DNSKEY record, which gives us the DNSKEY RRset shown above. Both the public KSK and public ZSK are signed by the private KSK. Resolvers can then use the public KSK to validate the public ZSK.

Validation for resolvers now looks like this:

- Request the desired RRset, which also returns the corresponding RRSIG record.
- Request the DNSKEY records containing the public ZSK and public KSK, which also returns the RRSIG for the DNSKEY RRset.
- Verify the RRSIG of the requested RRset with the public ZSK.
- Verify the RRSIG of the DNSKEY RRset with the public KSK.

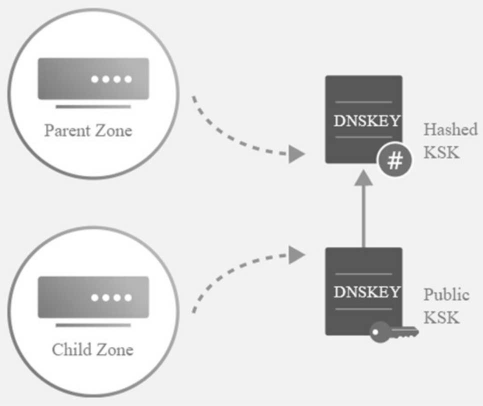


DNSSEC- How it works ?

https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

Delegation Signer Records

DNSSEC introduces a delegation signer (DS) record to allow the transfer of trust from a parent zone to a child zone. A zone operator hashes the DNSKEY record containing the public KSK and gives it to the parent zone to publish as a DS record.



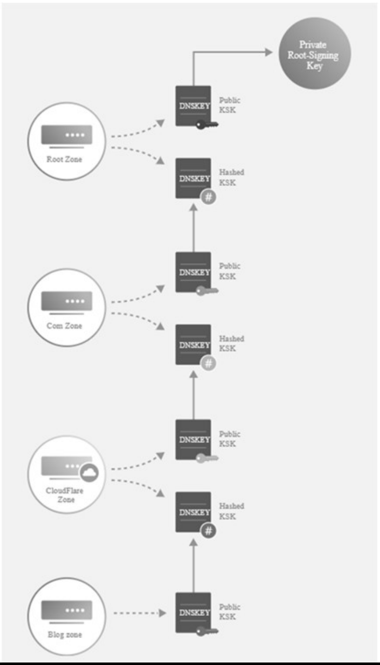
Every time a resolver is referred to a child zone, the parent zone also provides a DS record. This DS record is how resolvers know that the child zone is DNSSEC-enabled. To check the validity of the child zone's public KSK, the resolver hashes it and compares it to the DS record from the parent. If they match, the resolver can assume that the public KSK hasn't been tampered with, which means it can trust all of the records in the child zone. This is how a chain of trust is established in DNSSEC.

DNSSEC-

https://www.cloudflare.com/dns/dnssec/how-dnssec-works/

The Chain of Trust

Ok, so we have a way to establish trust within a zone and connect it to its parent zone, but how do we trust the DS record? Well, the DS record is signed just like any other RRset, which means it has a corresponding RRSIG in the parent. The whole validation process repeats until we get to the parent's public KSK. To verify that, we need to go to that parent's DS record, and on and on we go up the chain of trust.



Explicit Denial of Existence

If you ask DNS for the IP address of a domain that doesn't exist, it returns an empty answer—there's no way to explicitly say, "sorry, the zone you requested doesn't exist." This is a problem if you want to authenticate the response, since there's no message to sign. DNSSEC fixes this by adding the NSEC and NSEC3 record types. They both allow for an authenticated denial of existence.

NSEC works by returning the "next secure" record. For example, consider a name server that defines AAAA records for api, blog, and www. If you request a record for store, it would return an NSEC record containing www, meaning there's no AAAA records between store and www when the records are sorted alphabetically. This effectively tells you that store doesn't exist. And, since the NSEC record is signed, you can validate its corresponding RRSIG just like any RRset.