DNS Security Overview

Track 2 Workshop
June 2010
Pago Pago, American Samoa

What's the Problem?

Until July of 2008 the majority of authoritative DNS servers worldwide were completely insecure… Only we didn't know this.

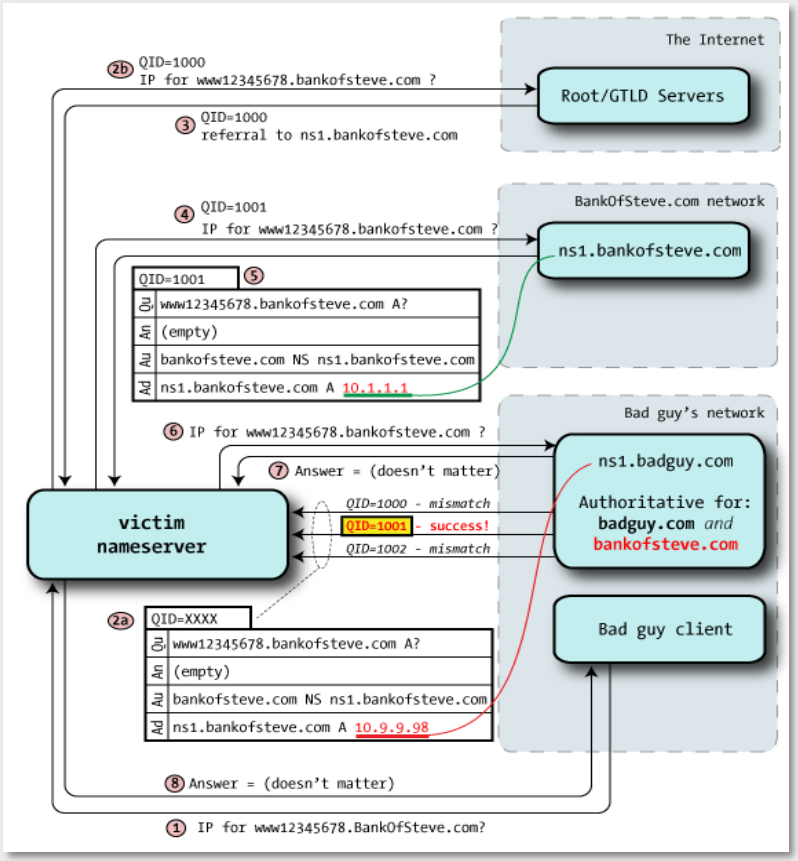
A couple of DNS experts had foreseen potential problems, so their software already had defenses against this security flaw - *djbdns* and *PowerDNS.*

The flaw is formally known as *DNS Cache Poisoning,* but is now referred to as "The Kaminsky Bug."

Dan Kaminsky's Bug

Dan Kaminsky

In March of 2008 was said to have mumbled the words, "*Oh sh_t, I just broke the Internet.*" The only problem was – he had.



- ✓ http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
- ✓ http://en.wikipedia.org/wiki/DNS_cache_poisoning
- ✓ http://www.wired.com/techbiz/people/magazine/16-12/ff_kaminsky?currentPage=1

An Emergency Meeting was Held…

…in building Nine on the Microsoft campus at 10 am on March 31.

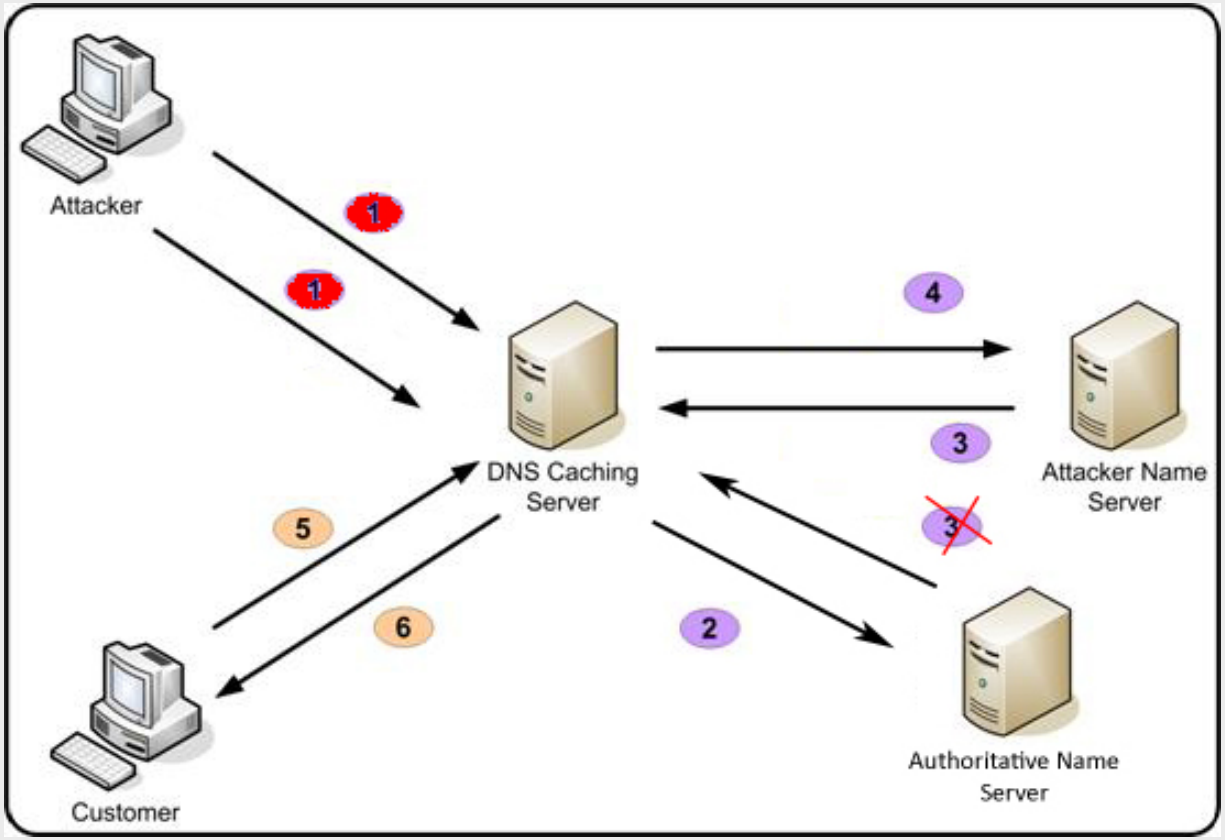
- Paul Vixie, an original author of BIND and president of the Internet Systems Consortium (isc.org) arranged the meeting and the attendees.
- Kaminsky presented and gave those present until August 6 to find a solution to the Cache Poisoning problem he had found.
- Secrecy around the issue was maintained until July 21 when a "security expert" leaked the flaw.
- The temporary solution is known as "*source port randomization*". The permanent solution is known as DNSSEC.

DNS Cache Poisoning

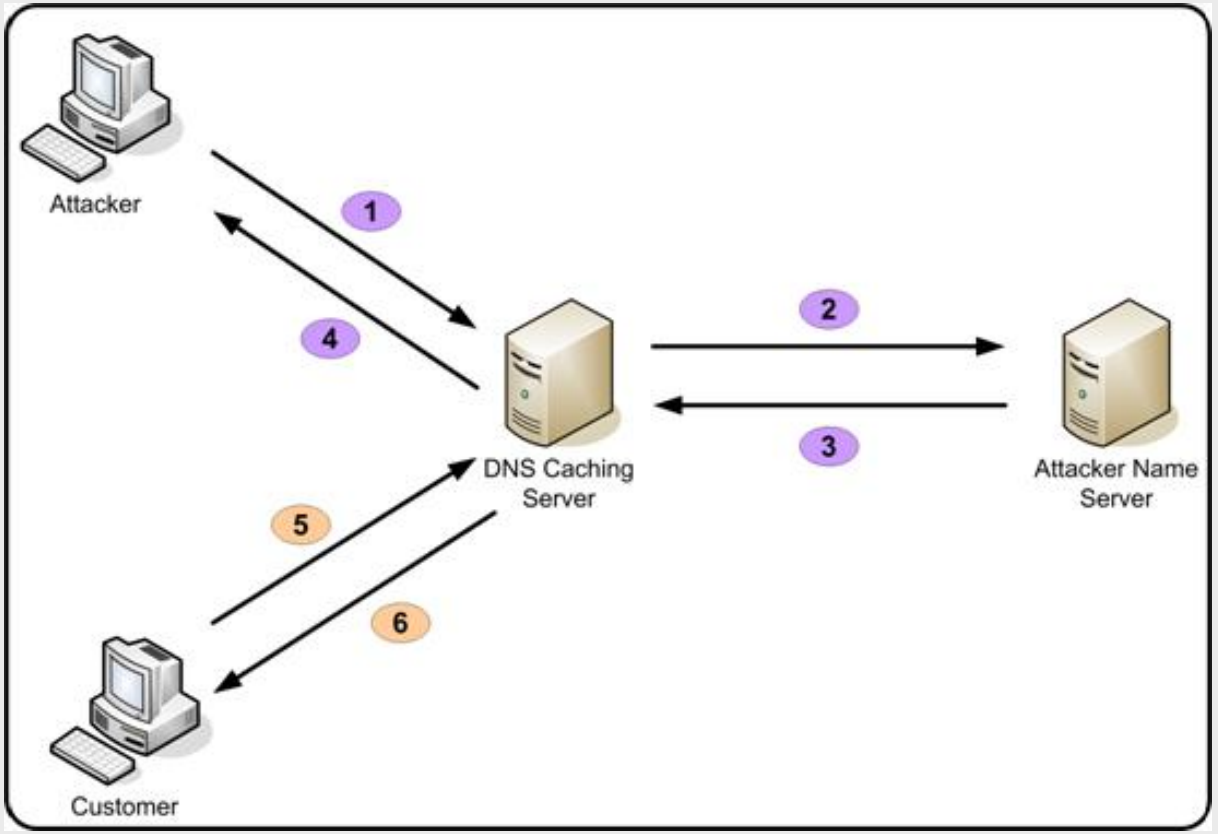
Conceptually quite simple:

1. Create a fake authoritative name server for an entire zone.
2. From attack machine ask a local DNS cache server for the IP address of a site in that zone not already in the cache.
3. Caching serving goes to authoritative server to ask for the site's IP address.
4. Meanwhile attack box floods the DNS cache with fake responses for the site. Fake responses include information indicating "I don't know the answer, but you can ask over there" – i.e. authoritative information for the zone is contained.
5. If the attacking box can "guess" the query ID that the caching server is listening for on port 53, then it will accept the response, believe the new authority record and *stop listening to the actual authoritative server for that zone!*
6. There are, by default, 65535 potential query IDs, but with proper flooding techniques you can generally "guess" a query ID within 10 seconds on an unpatched Caching server.

Cache Poisoning: The Attack



Cache Poisoning: Post Attack



Source Port Randomization

- Rather than 65535 possible query IDs increase this number dramatically – between several hundred million and 4 billion.
- You can still poison the cache, but a concerted, longer-term attack is required.
- You *must* monitor your DNS boxes for this.
- If you have not patched your DNS software, then your DNS server can be owned in 10 seconds or less…
- Sour port randomization does not solve the cache poisoning attack, it only puts a bandage on the issue…

DNS Security

We need DNS Security to solve DNS Cache Poisoning attacks as well as generally:

- Authenticating DNS data
- Authenticate the denial of existence of domains.
- To ensure data integrity

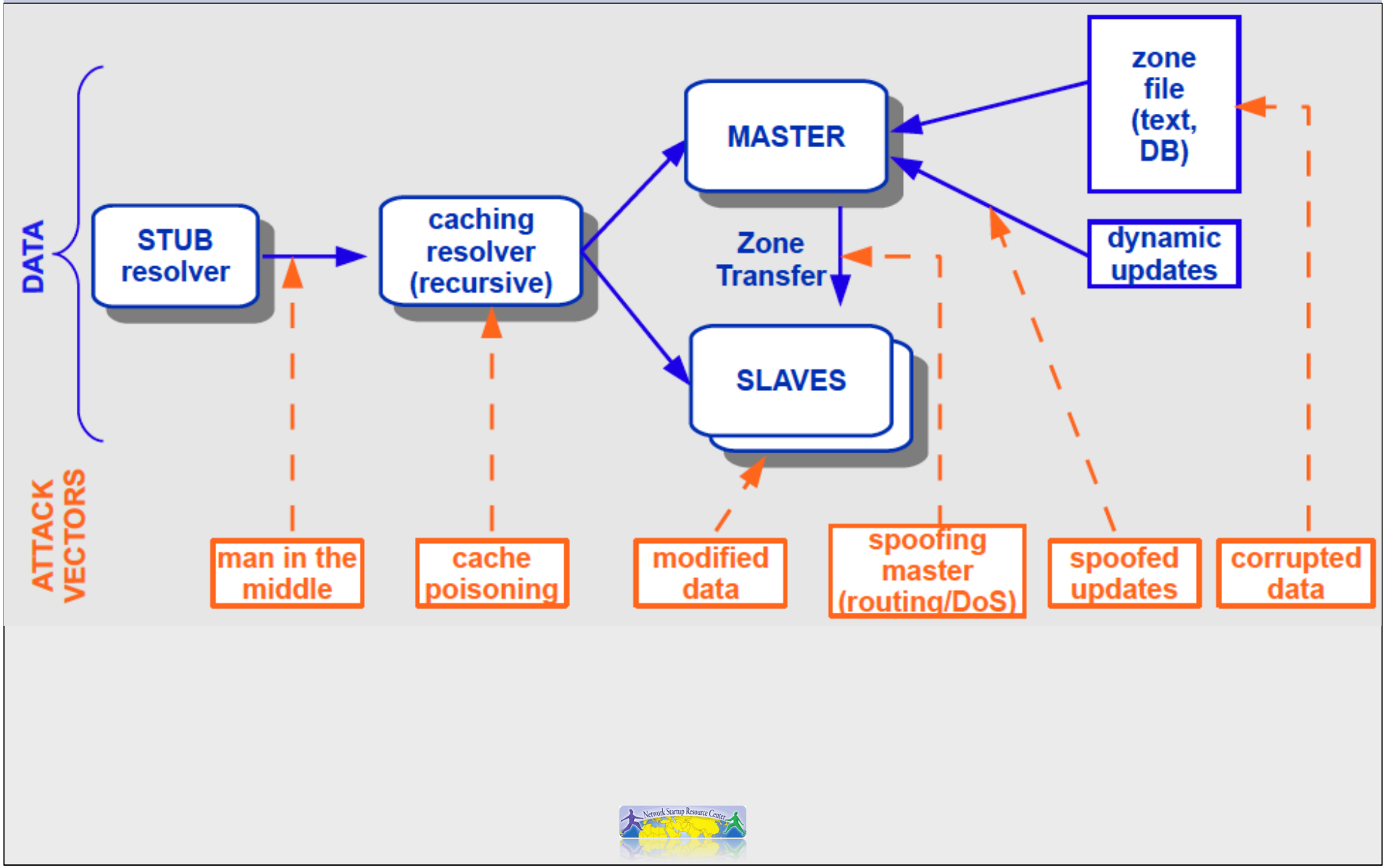
or – in “plain English”

To be able to verifiably know that the reply to our DNS query is valid and has not been spoofed.

DNS Security Does Not…

- Encrypt the data
- Stop DDoS attacks on DNS Servers
- Encrypt DNS zone transfer data

Where is the DNS Vulnerable?



Implementing DNS Security

As the Beatles might say, “This has been a long and winding road.”

In 12 more days the “.” will be signed. A big deal:

http://www.root-dnssec.org/

We have a celebrity in our midst…

Trusted Community Representative (One of “the 14”)
Crypto Officer for the US West Coast Facility
http://www.root-dnssec.org/tcr/selection-2010/

How do we Secure DNS?

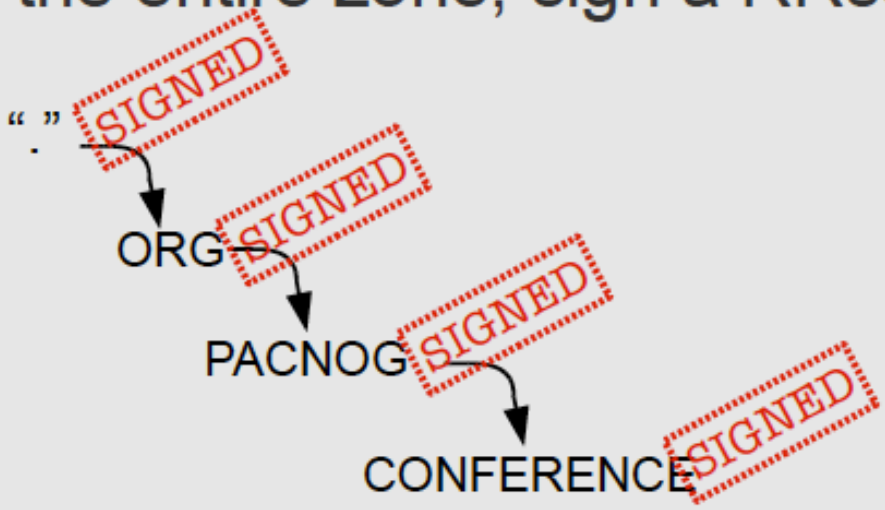
- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEYs published, used to verify the RRSIGs (Resource Record Signatures)
- Children sign their zones with their private key
  - Authenticity of that key established by signature (checksum) by the parent of the delegation signer (DS) record (i.e. the record used to identify the DNSSEC signing key of a delegated zone).
- Repeat for parent...
- Not that difficult on paper
  - Operationally, it is a bit more complicated

How do we Secure DNS?

- Changes DNS trust model from one of ”open” and ”trusting” to one of ”verifiable”
- Extensive use of public key cryptography to provide:
  - Authentication of origin
  - Data integrity
  - Authenticated denial of existence
- No attempt to provide confidentiality
- DNSSEC does not place computational load on the authoritative servers (!= those *signing* the zone)
- No modifications to the core protocol
  - Can coexist with today's infrastructure
    - ... kind of (EDNS0)

How do we Secure DNS?

- Build a chain of trust using the existing delegation-based model of distribution that is the DNS
- Don't sign the entire zone, sign a RRset



- Note: the parent DOES NOT sign the child zone.
  - The parent signs a *pointer* (hash) to the *key* used to sign the data of child zone (important!)

How do we Secure DNS?

To secure the DNS against spoofing we start with 4 new Resource Records (RRs):

- **DNSKEY**: Public Key of RRSIG record. Used in zone signing operations.
- **RRSIG**: Resource Record set SIGnature
- **NSEC/NSEC3**: Next SECure record. Returned as verifiable evidence that the name and/or RR type does not exist.
- **DS** → See next slide…

Complete list of DNS records:

http://en.wikipedia.org/wiki/List_of_DNS_record_types

The DS Record

Delegation Signer: Contains the hash of the public key used to sign the key which itself will be used to sign the zone data. Follow DS RR's until a ”trusted” zone is reached (ideally the root).

An excellent discussion of DNSSec by Geoff Houston:
http://ispcolumn.isoc.org/2006-08/dnssec.html

RRset

- Multiple resource records with *same name and type* are grouped into Resource Record Sets (RRsets):

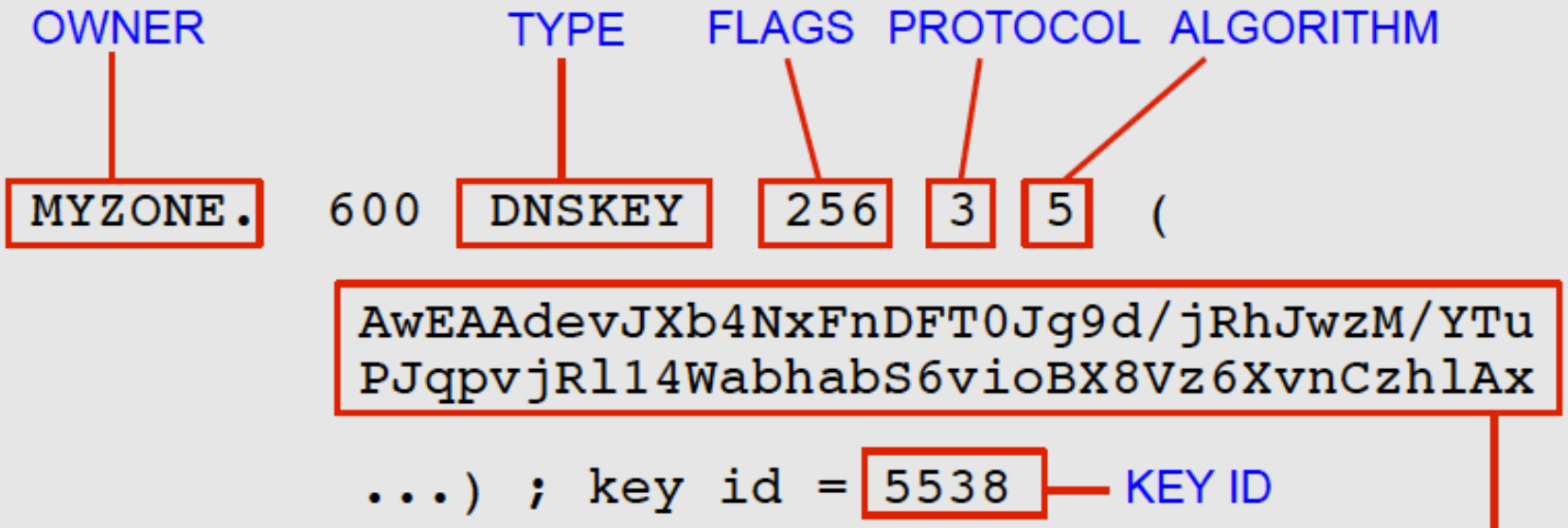
```
mail.zone.      MX    5 server1.zone.
mail.zone.      MX    10 server2.zone.
```
} RRset

```
server1.zone.   A     10.20.30.40
server1.zone.   A     10.20.30.41
server1.zone.   A     10.20.30.42
```
} RRset

```
server1.zone.   AAAA  2001:123:456::1
server1.zone.   AAAA  2001:123:456::2
```
} RRset

```
server2.zone.   A     11.22.33.44
```
} RRset

DNSKEY Resource Record



- FLAGS determines the usage of the key (more on this...)
- PROTOCOL is always 3 in the current version of DNSSEC
- ALGORITHM can be:
  - 0 – reserved
  - 1 – RSA/MD5 (deprecated)
  - 2 – Diffie/Hellman
  - 3 – DSA/SHA-1 (optional)
  - 4 – reserved
  - 5 – RSA/SHA-1 (mandatory)

DNSKEY Resource Record

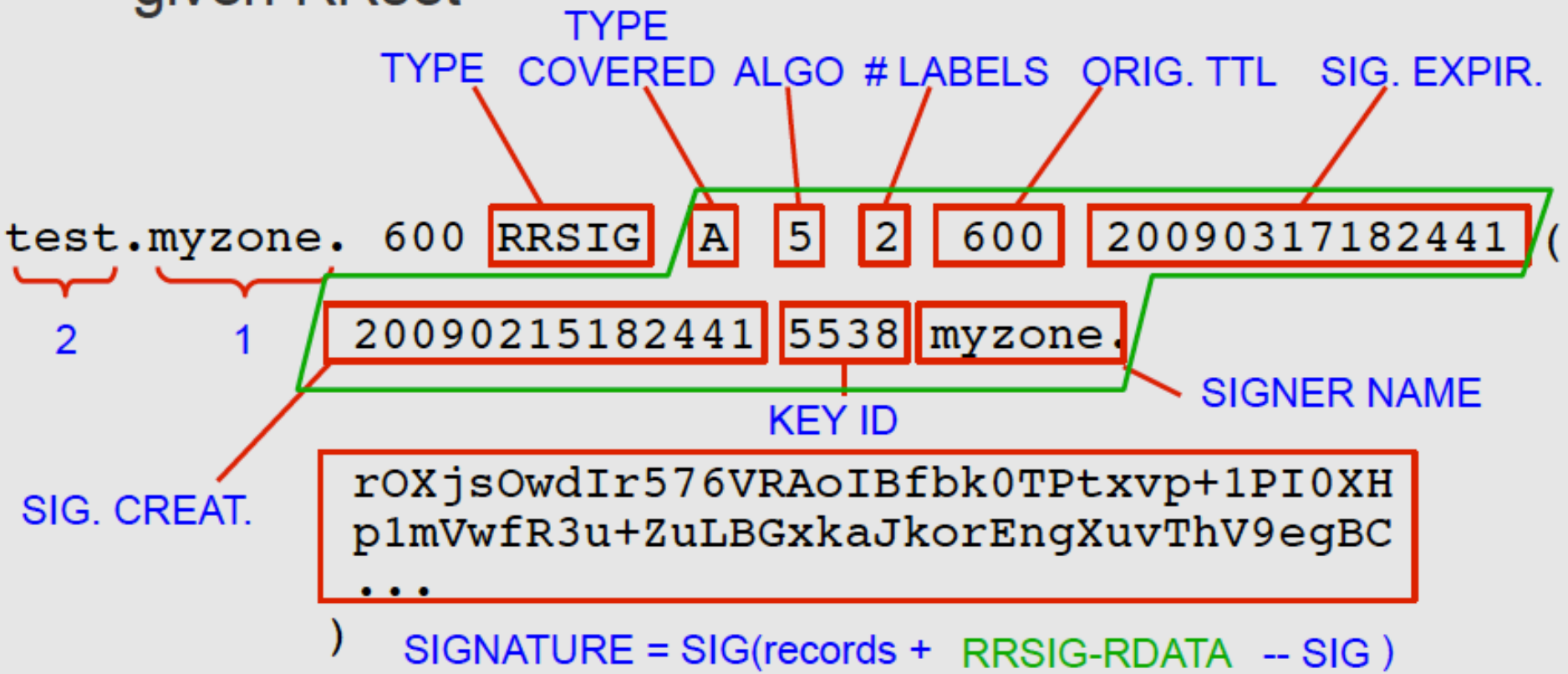
- There are in practice at least two DNSKEY pairs for every zone:
  - Originally, one key-pair (public, private) defined for the zone:
    - private key used to sign the zone data (RRsets)
    - public key published (DNSKEY) in zone
    - DS record (DNSKEY hash) published in parent zone, and signed in turn with rest of data
- Problem with using a single key:
  - to update this key, DS record in parent zone needs to be updated (since DS is fingerprint of public key)
    - Introduction of Key Signing Key (flags = 257)

Key Signing Key / Zone Signing Key

- To allow for key updates ("rollovers"), generate two keys:
  - Key Signing Key (KSK)
    - pointed to by parent zone (Secure Entry Point), in the form of DS (Delegation Signer)
    - used to sign the Zone Signing Key (ZSK)
  - Zone Signing Key (ZSK)
    - signed by the Key Signing Key
    - used to sign the zone data RRsets
- This decoupling allows for independent updating of the ZSK without having to update the KSK, and involve the parent – less administrative interaction.

DNSSEC: RRSIG

- Resource Record Signature
  - lists the signatures performed using the ZSK on a given RRset



DNSSEC: RRSIG

- By default:
  - Signature creation time is *1 hour before*
  - Signature expiration is *30 days from now*
  - Needless to say, proper timekeeping (NTP) is strongly recommended
- What happens when the signatures run out ?
  - SERVFAIL...
  - Your domain effectively disappears from the Internet
  - ... more on this later
- Note that the *keys* do *not* expire.
- Therefore, *regular* re-signing is part of the operations process (not only when changes occur)
  - the entire zone doesn't have to be resigned...

DNSSEC: NSEC/NSEC3

- NSEC – proof of non-existence
- Remember, the authoritative servers are serving precalculated records. No on-the-fly generation is done.
  - NSEC provides a pointer to the **N**ext **SEC**ure record in the chain of records.
    - "there are no other records between this one and the next", signed.
  - The entire zone is sorted lexicographically:

```
myzone.
sub.myzone.
test.myzone.
```

DNSSEC: NSEC/NSEC3

```
myzone. 10800 NSEC test.myzone. NS SOA RRSIG NSEC DNSKEY

myzone. 10800 RRSIG NSEC 5 1 10800 20090317182441 (
                    20090215182441 5538 myzone.

                    ZTYDLeUDMlpsp+IWV8gcUVRkIr7KmkVS5TPH
                    KPsxgXCnjnd8qk+ddXlrQerUeho4RTq8CpKV
                      ...
                    )
```

- Last NSEC record points back to the first.
- Problem:
  - Zone enumeration (walk list of NSEC records)
  - Yes, DNS shouldn't be used to store sensitive information, but future uses may require this “feature” (data privacy laws)

DNSSEC: NSEC/NSEC3

- If the server responds NXDOMAIN:
  - One or more NSEC RRs indicate that the name (or a wildcard expansion) does not exist
- If the server's response is NOERROR:
  - ...and the answer section is empty
    - ➔The NSEC proves that the TYPE did not exist

DNSSEC: NSEC/NSEC3

- What about NSEC3 ?
  - We won't get into details here, but the short story is:
    - Don't sign the name of the Next SECure record, but a ***hash*** of it
      - Still possible to prove non-existence, *without* revealing name.
    - This is a simplified explanation. RFC 5155 covering NSEC3 is *53* pages long.
  - Also introduces the concept of "opt-out" (see section 6 of the RFC) which has uses for so-called delegation-centric zones with unsigned delegations – in short: don't bother signing RRsets for delegations which you know don't implement DNSSEC.

DNSSEC: DS

- Delegation Signer
- Hash of the KSK of the child zone
- Stored in the parent zone, together with the NS RRs indicating a delegation of the child zone
- The DS record for the child zone is signed *together* with the rest of the parent zone data
  NS records are *NOT* signed (they are a hint/pointer)

```
myzone. DS 61138 5 1        Digest type 1 = SHA-1, 2 = SHA-256
 F6CD025B3F5D03040895053 54A0115584B56D683
myzone. DS 61138 5 2
 CCBC0B557510E4256E88C01B0B1336AC4ED6FE08C826
 8CC1AA5FBF00 5DCE3210

 digest = hash( canonical FQDN on KEY RR | KEY_RR_rdata)
```

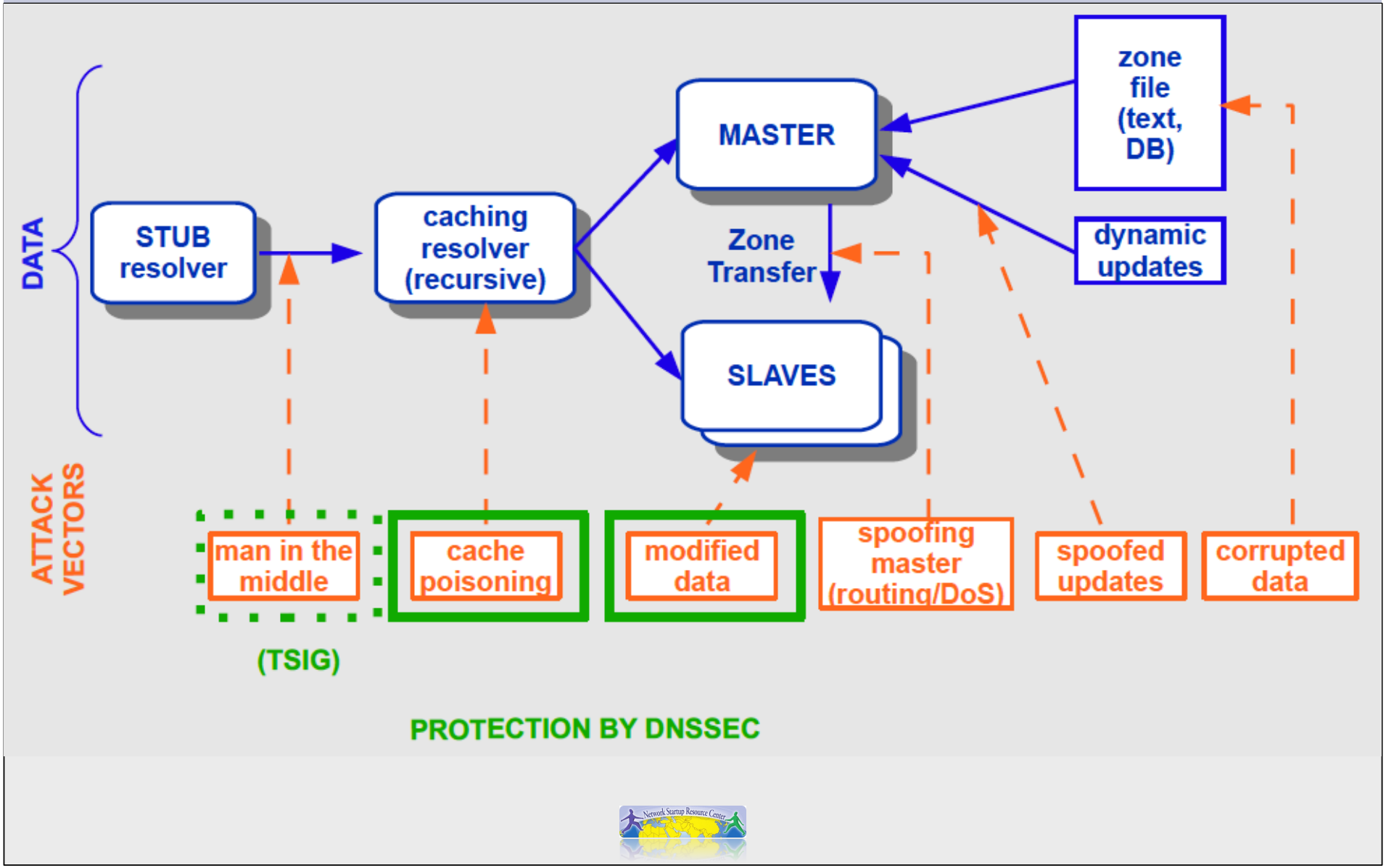
DNSSEC: DS

- Two hashes generated by default:
  - 1 SHA-1 MANDATORY
  - 2 SHA-256 MANDATORY
- New algorithms are being standardised upon
- This will happen continually as algorithms are broken/proven to be unsafe

DNSSEC: New Fields

- Updates DNS protocol at the packet level
- Non-compliant DNS recursive servers *should* ignore these:
  - **CD**: Checking Disabled (ask recursing server to not perform validation, even if DNSSEC signatures are available and verifiable, i.e.: a Secure Entry Point can be found)
  - **AD**: Authenticated Data, set on the answer by the validating server if the answer could be validated, and the client requested validation
- A new EDNS0 option
  - **DO**: DNSSEC OK (EDNS0 OPT header) to indicate client support for DNSSEC options

What Does this Protect in the end?



Conclusion

Next we'll do some exercises and discuss issues.

In addition – we have references to a large bibliography and a recent status update of DNSSEC deployments available linked on this classes web pages:

http://www.pacnog.org/pacnog7/track2/