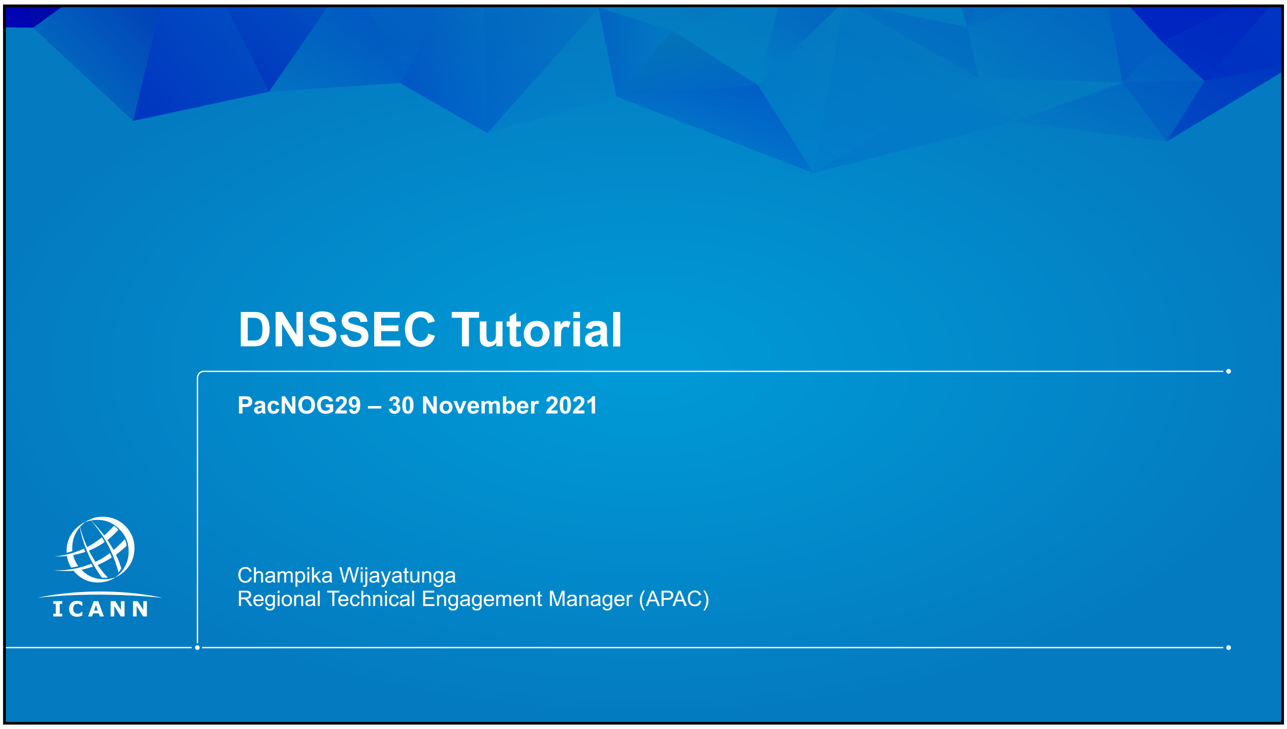
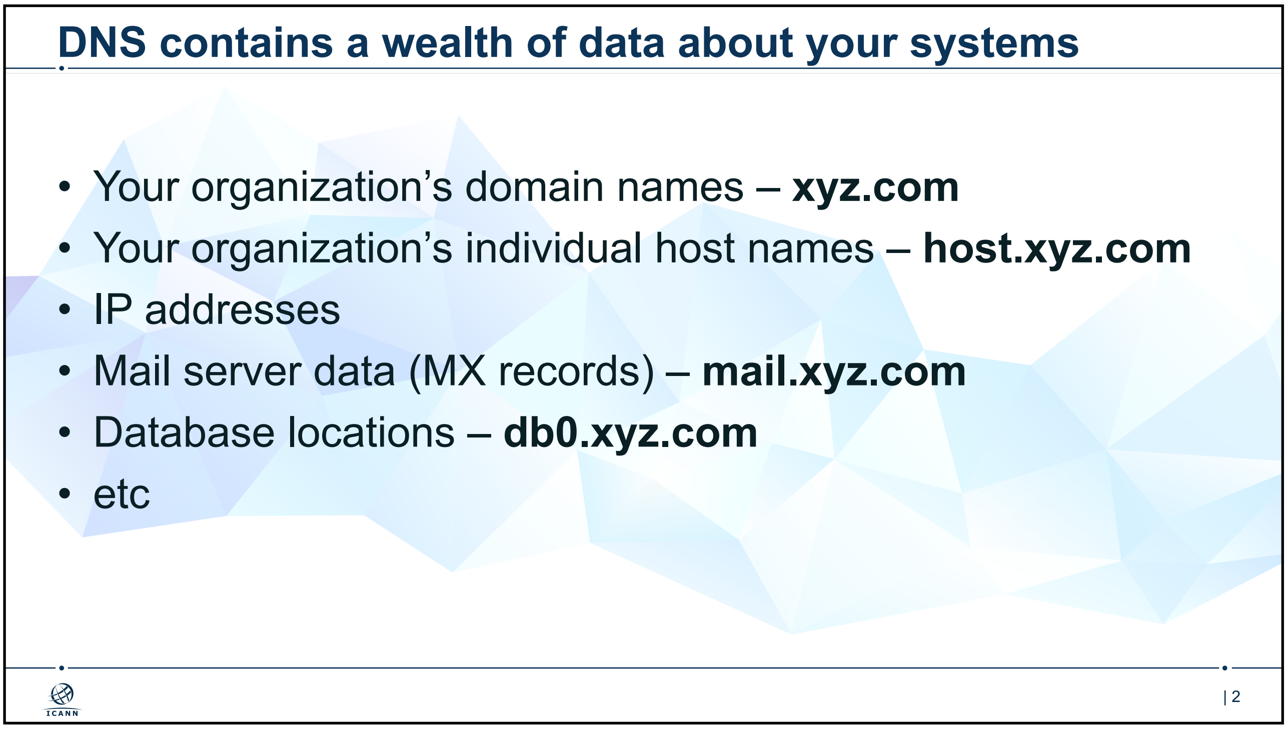


DNSSEC Tutorial

PacNOG29 – 30 November 2021

Champika Wijayatunga
Regional Technical Engagement Manager (APAC)

1



DNS contains a wealth of data about your systems

- Your organization's domain names – **xyz.com**
- Your organization's individual host names – **host.xyz.com**
- IP addresses
- Mail server data (MX records) – **mail.xyz.com**
- Database locations – **db0.xyz.com**
- etc

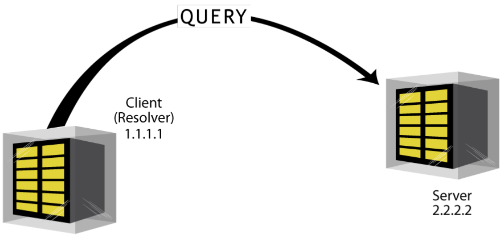
| 2

2

A world without DNSSEC…

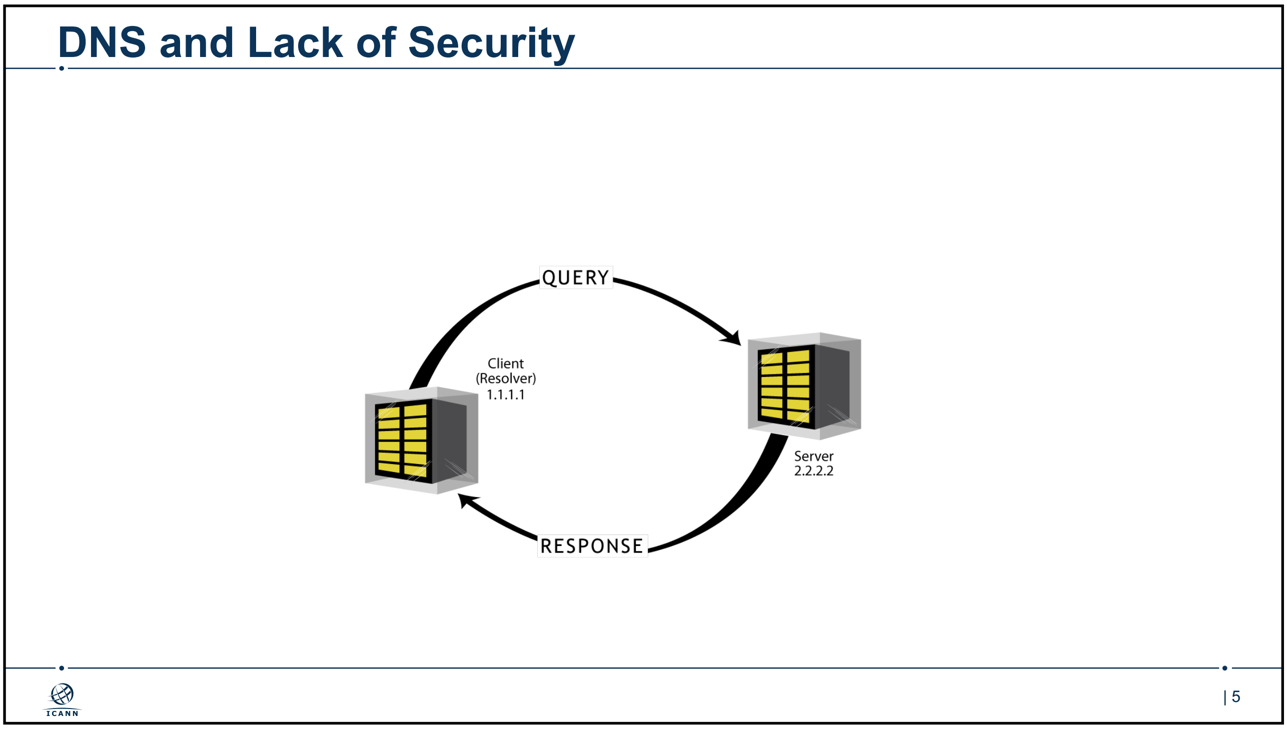
3

DNS and Lack of Security



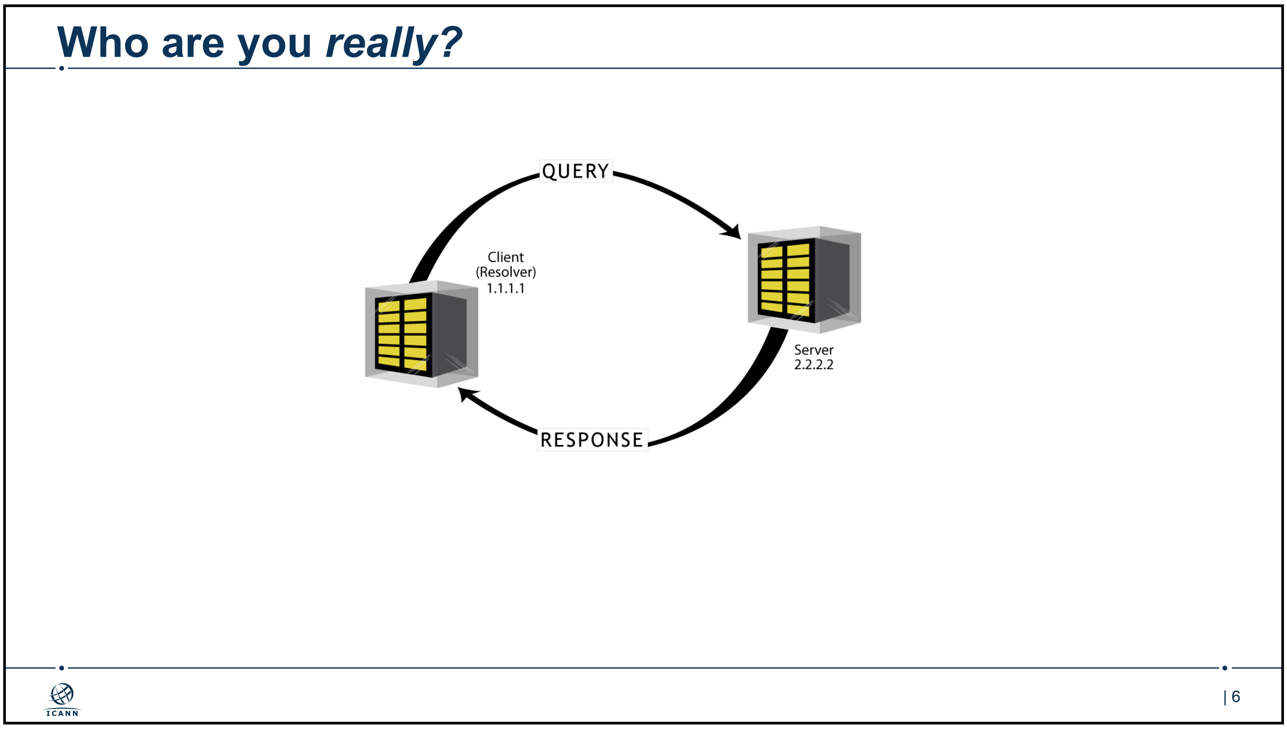
| 4

4



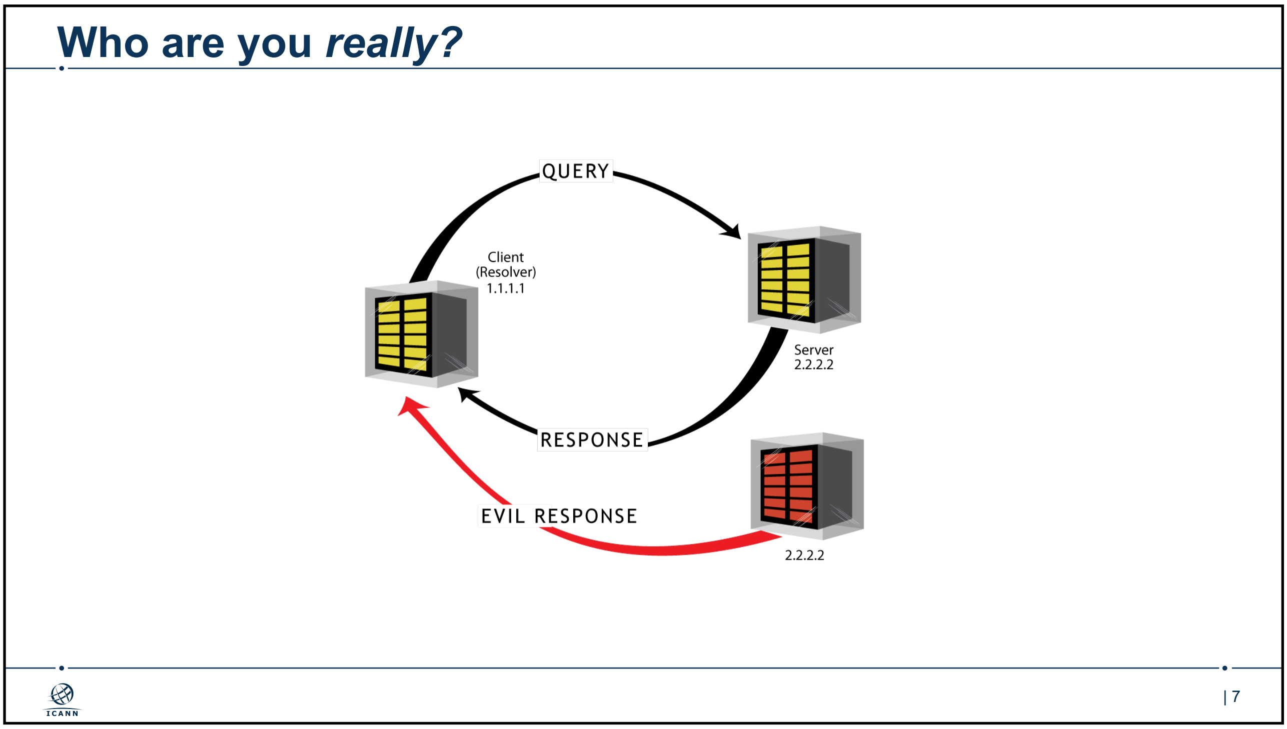
DNS and Lack of Security

5



Who are you *really?*

6



Who are you *really?*

QUERY

Client (Resolver) 1.1.1.1

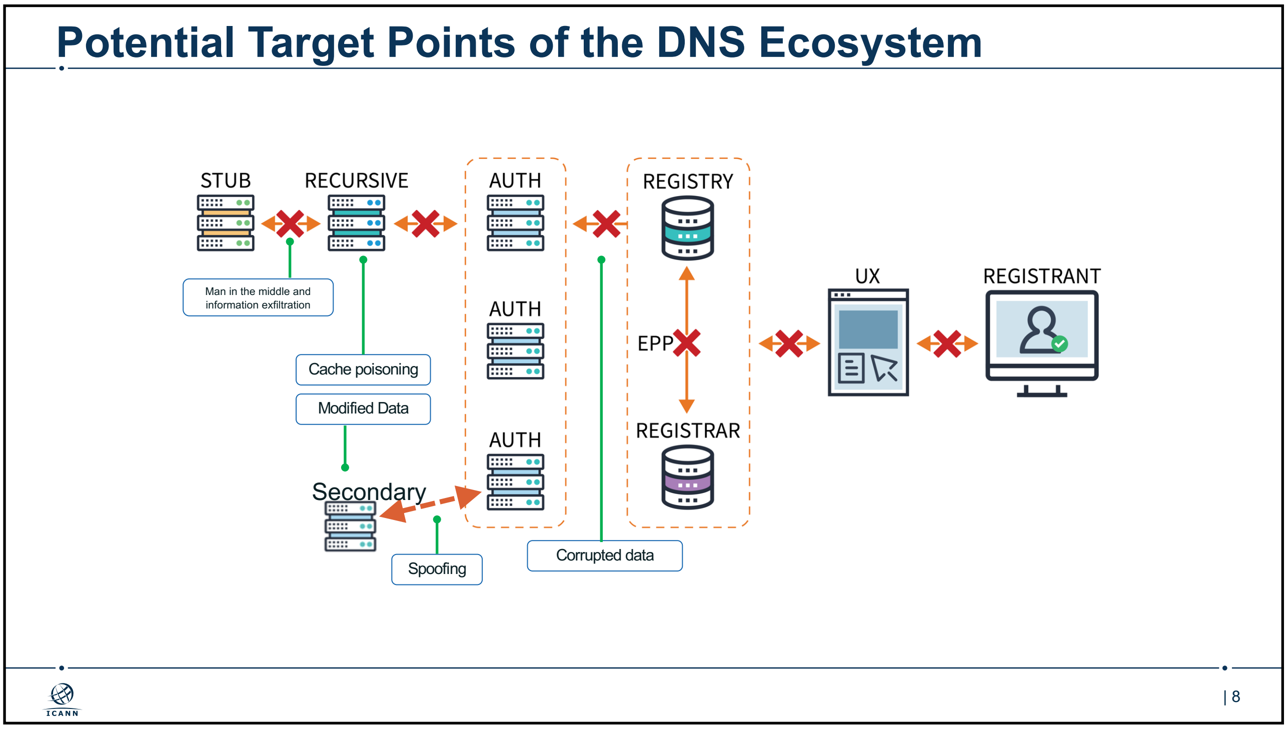
Server 2.2.2.2

RESPONSE

EVIL RESPONSE

2.2.2.2

7



Potential Target Points of the DNS Ecosystem

STUB

RECURSIVE

AUTH

REGISTRY

Man in the middle and information exfiltration

AUTH

EPP

UX

REGISTRANT

Cache poisoning

Modified Data

AUTH

REGISTRAR

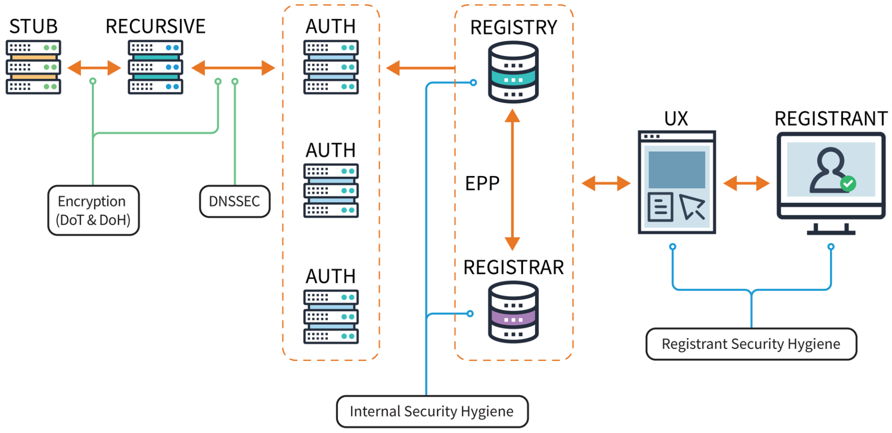
Secondary

Spoofing

Corrupted data

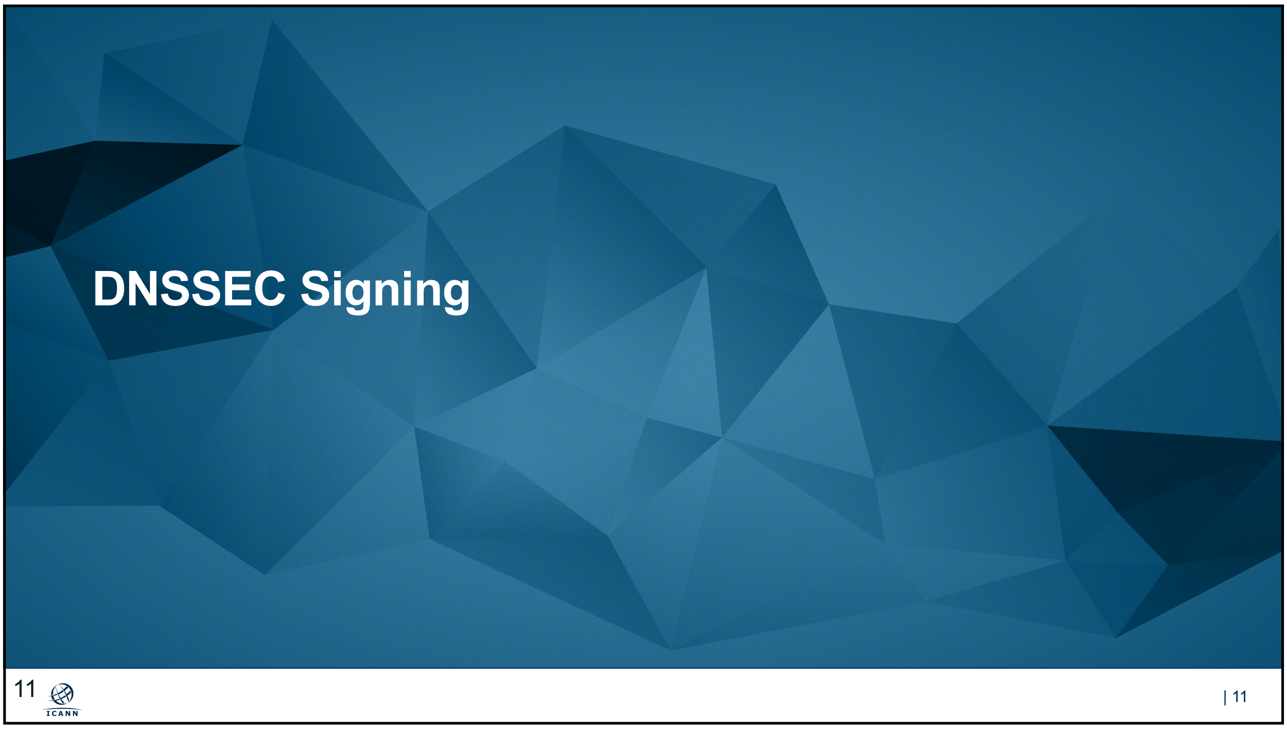
8

A More Secure DNS Ecosystem

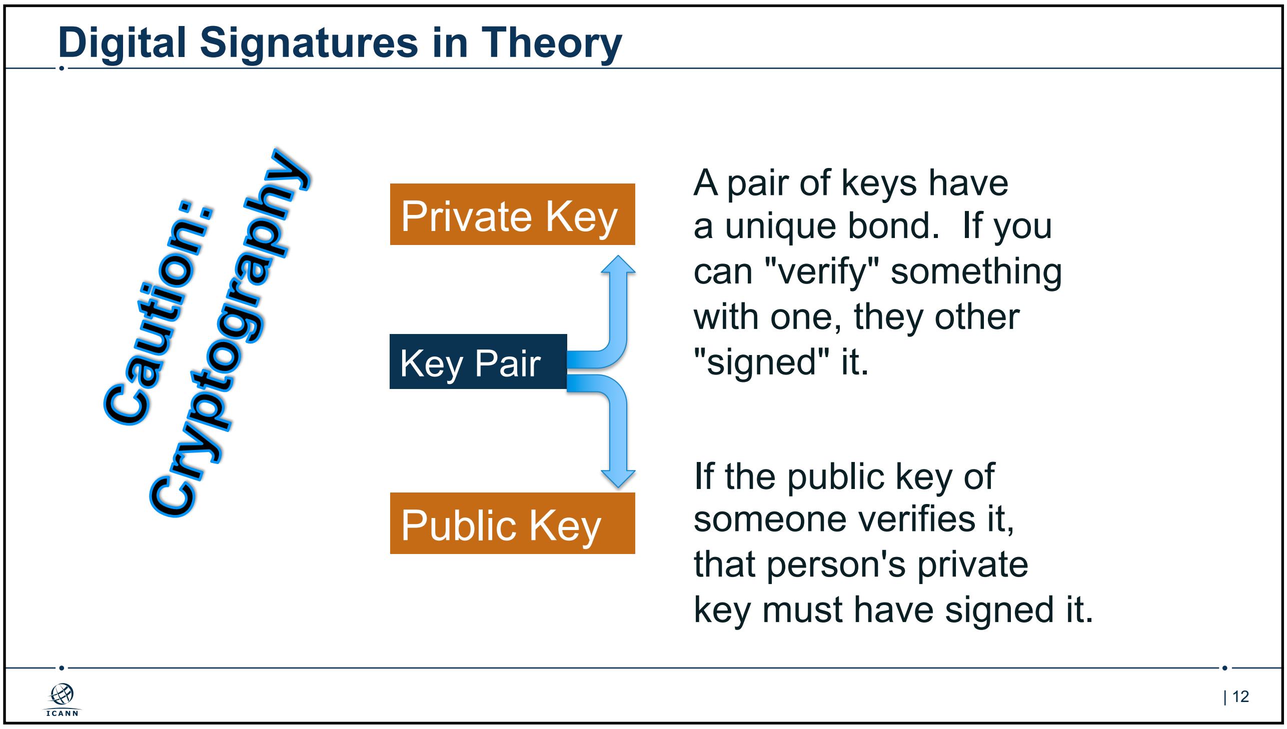


What DNSSEC Does

- DNSSEC uses public-key cryptography and digital signatures to provide:
  - Data origin authentication
    - "Did this response really come from the *example.com* zone authority?"
  - Data integrity
    - "Did an attacker (e.g., a man in the middle) modify the data in this response since the data was originally signed?"
- DNSSEC offers protection against spoofing of DNS data (and so, for attacks like cache-poisoning, etc.).



DNSSEC Signing



Digital Signatures in Theory

Caution: Cryptography

Private Key

Key Pair

Public Key

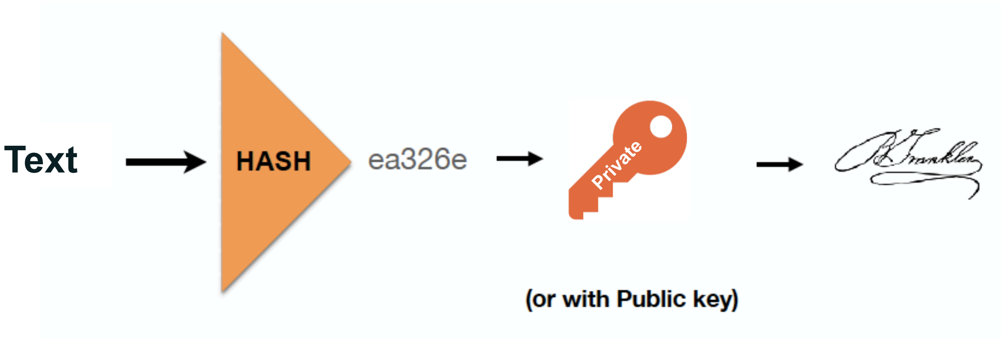
A pair of keys have a unique bond. If you can "verify" something with one, they other "signed" it.

If the public key of someone verifies it, that person's private key must have signed it.

Digital Signature

- We may combine *hash* with *private and public key*, to obtain a digital signature of any text

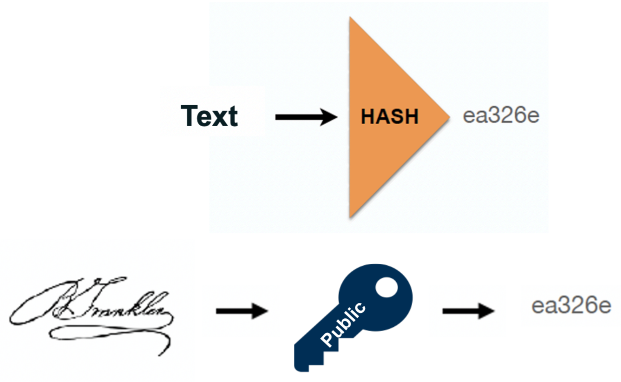
Hashing + Encrypt = Digital Signature



(or with Public key)

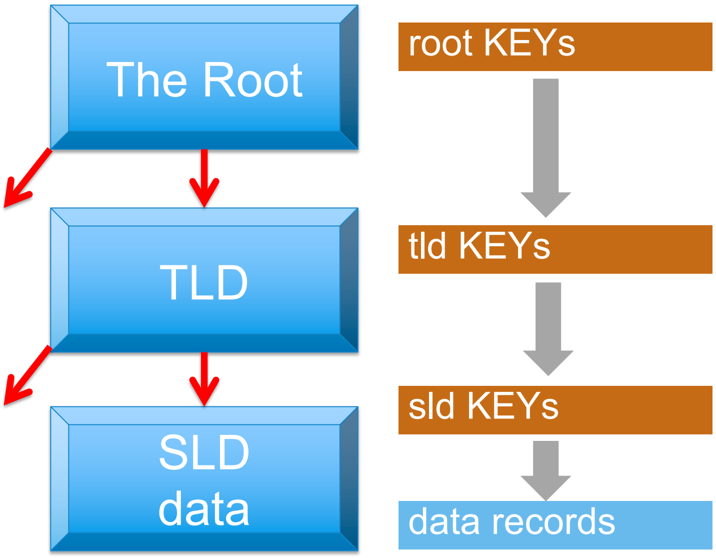
Digital Signature

- To verify the digital signature I need the ***text*** and the ***public key*** (or *private key* if signed with *public key*)



(or with private key)

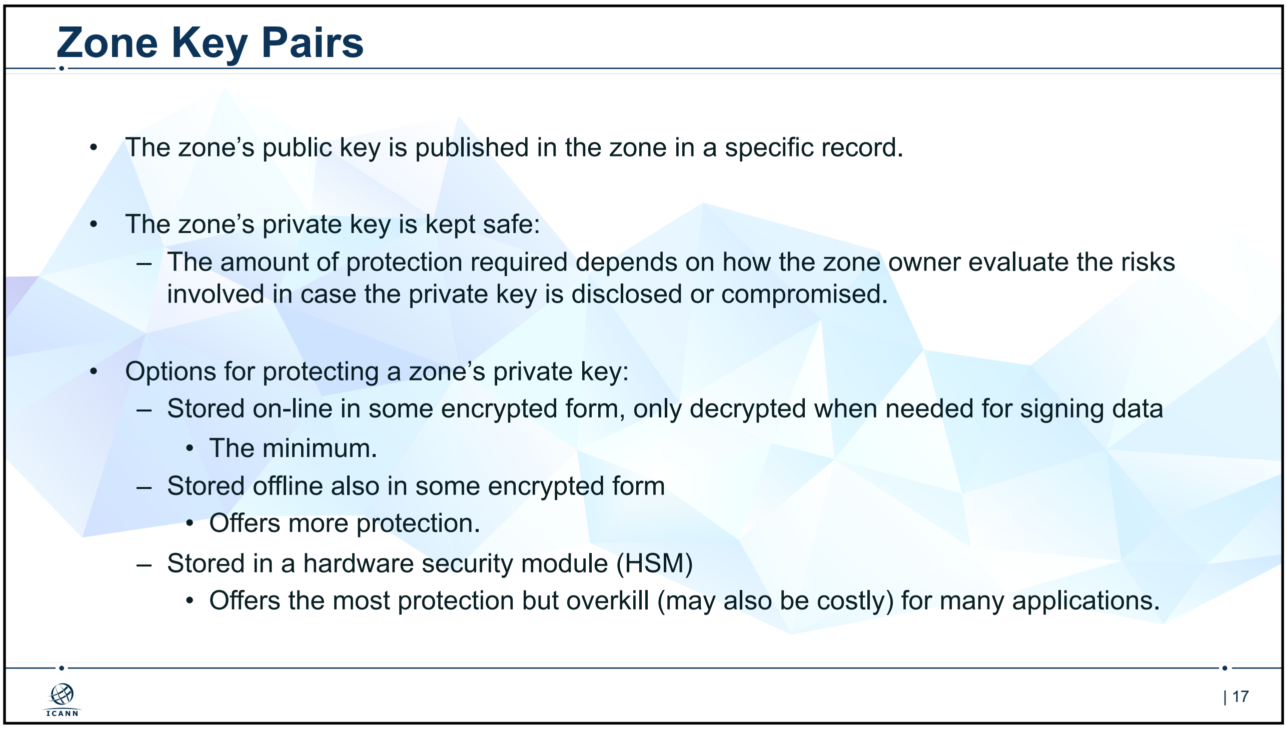
DNSSEC Chain of Keys In Theory



Making A Chain

- The root zone signs TLD keys
- A TLD (administrator) signs registrant keys
- A DNS zone administrator (registrant) signs their own data

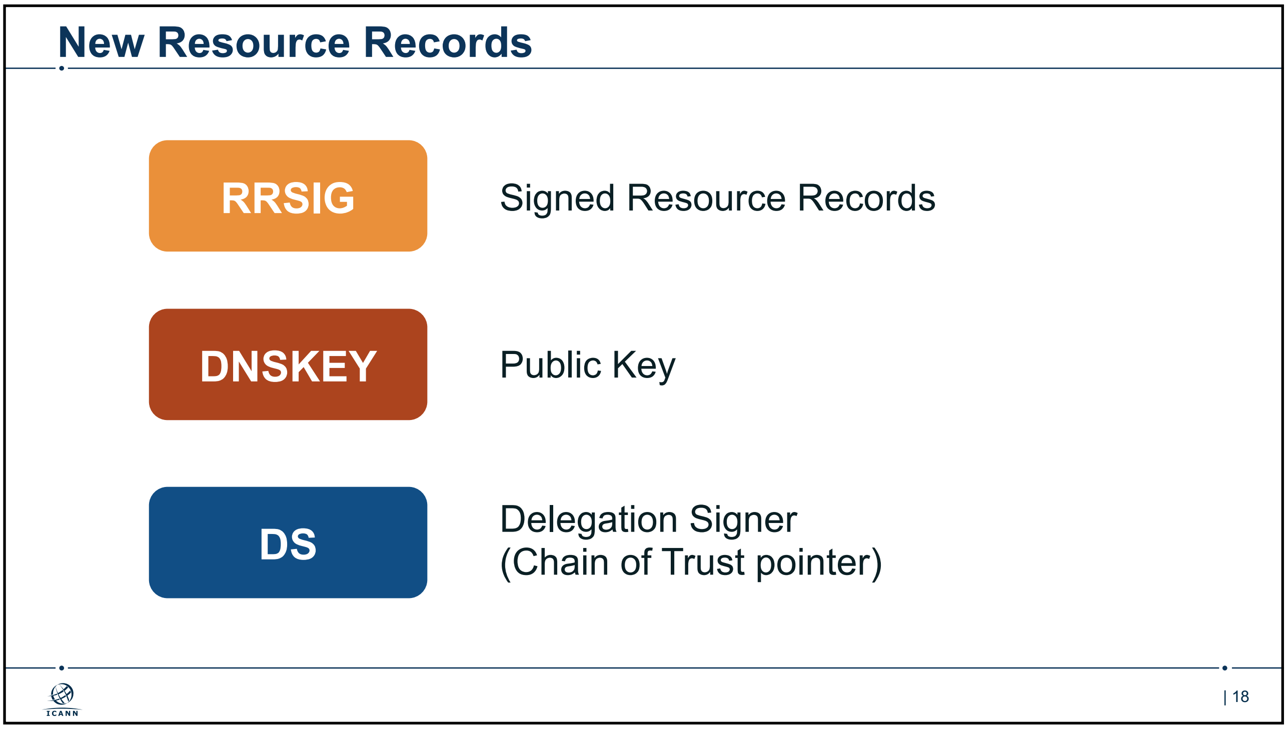
This creates a "chain" used in validation



Zone Key Pairs

- The zone’s public key is published in the zone in a specific record.
- The zone’s private key is kept safe:
  - The amount of protection required depends on how the zone owner evaluate the risks involved in case the private key is disclosed or compromised.
- Options for protecting a zone’s private key:
  - Stored on-line in some encrypted form, only decrypted when needed for signing data
    - The minimum.
  - Stored offline also in some encrypted form
    - Offers more protection.
  - Stored in a hardware security module (HSM)
    - Offers the most protection but overkill (may also be costly) for many applications.

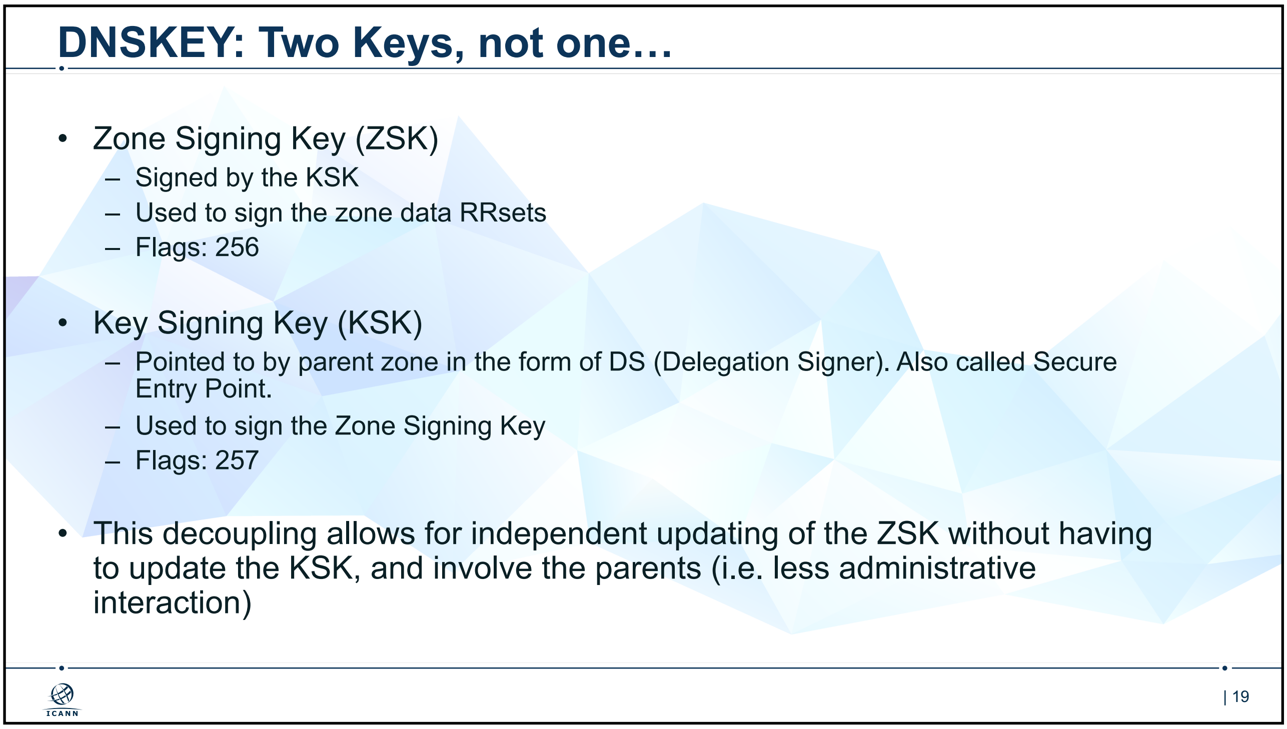
17



New Resource Records

RRSIG	Signed Resource Records
DNSKEY	Public Key
DS	Delegation Signer (Chain of Trust pointer)

18

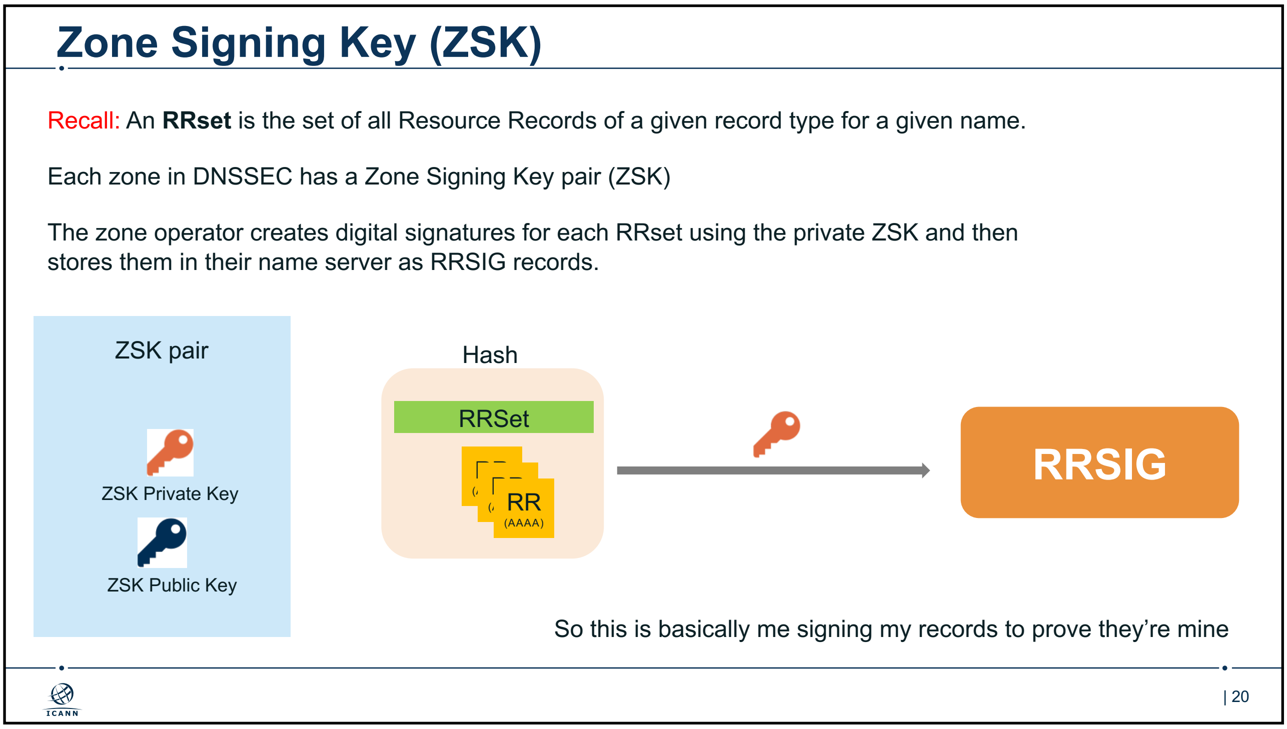


DNSKEY: Two Keys, not one…

- Zone Signing Key (ZSK)
  - Signed by the KSK
  - Used to sign the zone data RRsets
  - Flags: 256
- Key Signing Key (KSK)
  - Pointed to by parent zone in the form of DS (Delegation Signer). Also called Secure Entry Point.
  - Used to sign the Zone Signing Key
  - Flags: 257
- This decoupling allows for independent updating of the ZSK without having to update the KSK, and involve the parents (i.e. less administrative interaction)

| 19

19



Zone Signing Key (ZSK)

Recall: An **RRset** is the set of all Resource Records of a given record type for a given name.

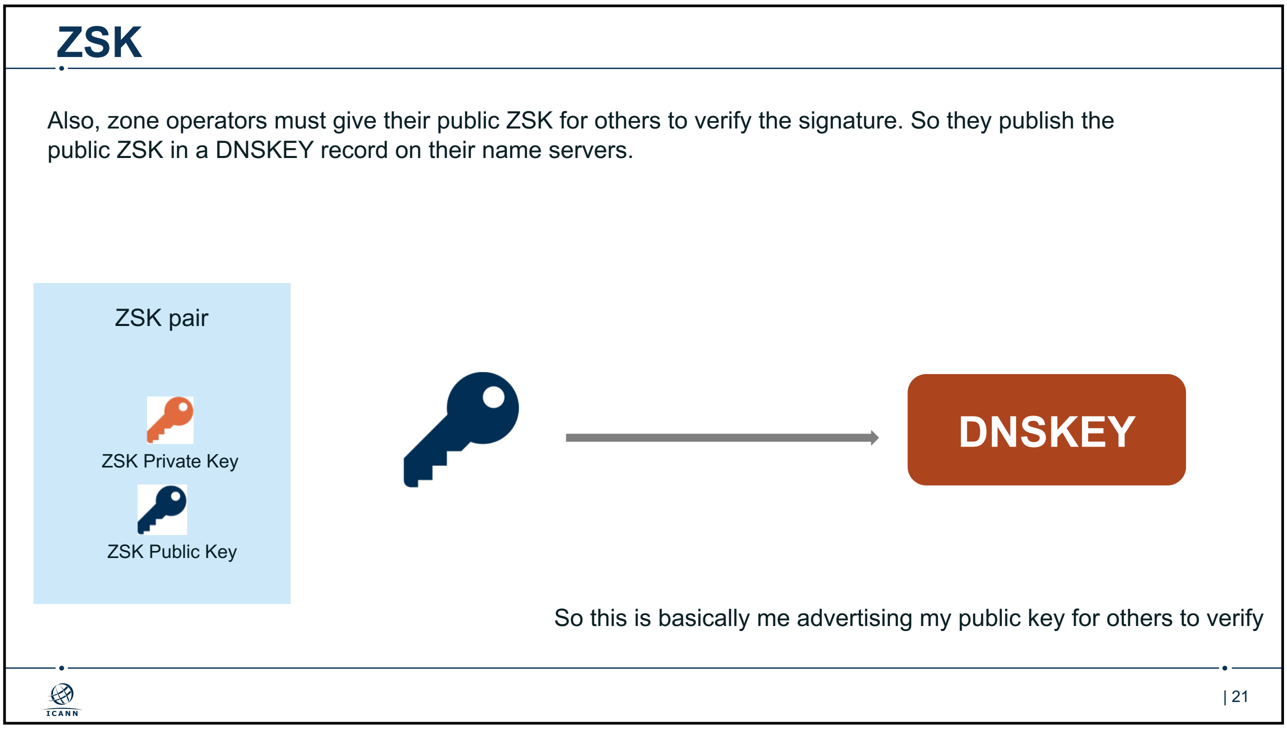
Each zone in DNSSEC has a Zone Signing Key pair (ZSK)

The zone operator creates digital signatures for each RRset using the private ZSK and then stores them in their name server as RRSIG records.

So this is basically me signing my records to prove they're mine

| 20

20



ZSK

Also, zone operators must give their public ZSK for others to verify the signature. So they publish the public ZSK in a DNSKEY record on their name servers.

ZSK pair

ZSK Private Key

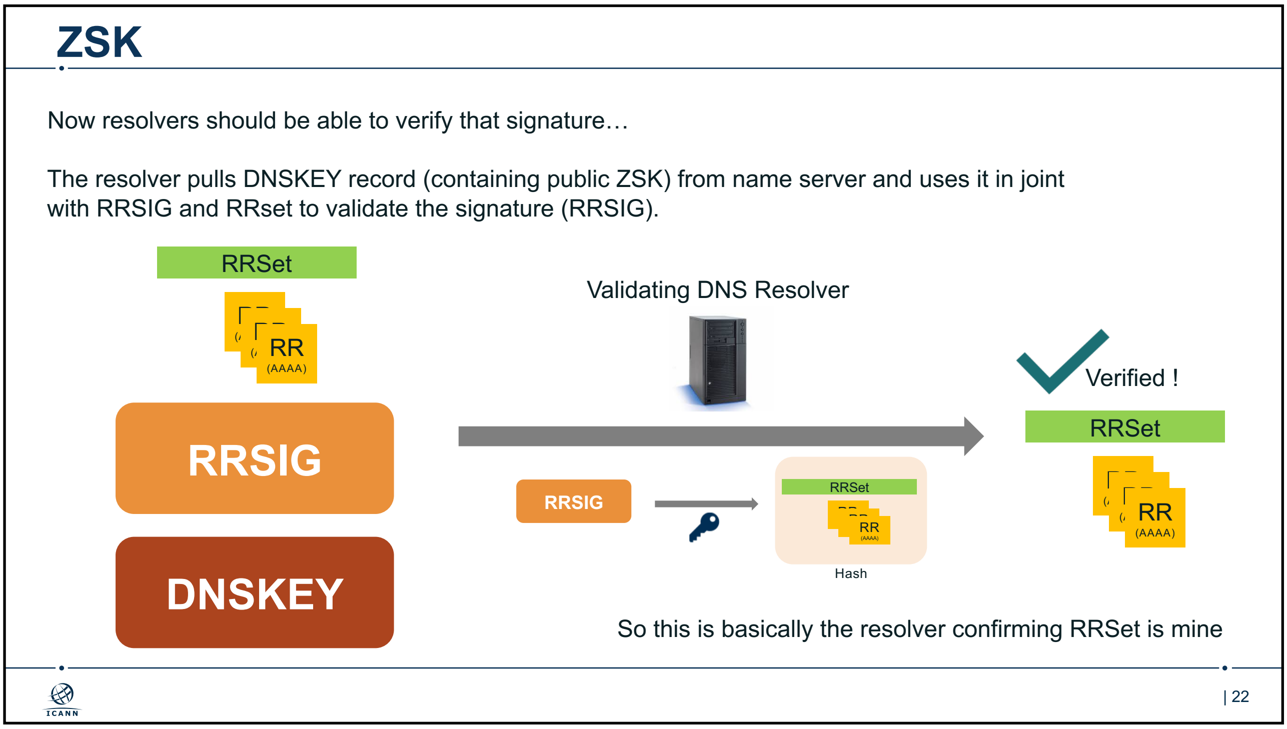
ZSK Public Key

DNSKEY

So this is basically me advertising my public key for others to verify

ICANN | 21

21



ZSK

Now resolvers should be able to verify that signature…

The resolver pulls DNSKEY record (containing public ZSK) from name server and uses it in joint with RRSIG and RRset to validate the signature (RRSIG).

RRSet

RR (AAAA)

RRSIG

DNSKEY

Validating DNS Resolver

RRSIG

RRSet

RR (AAAA)

Hash

Verified !

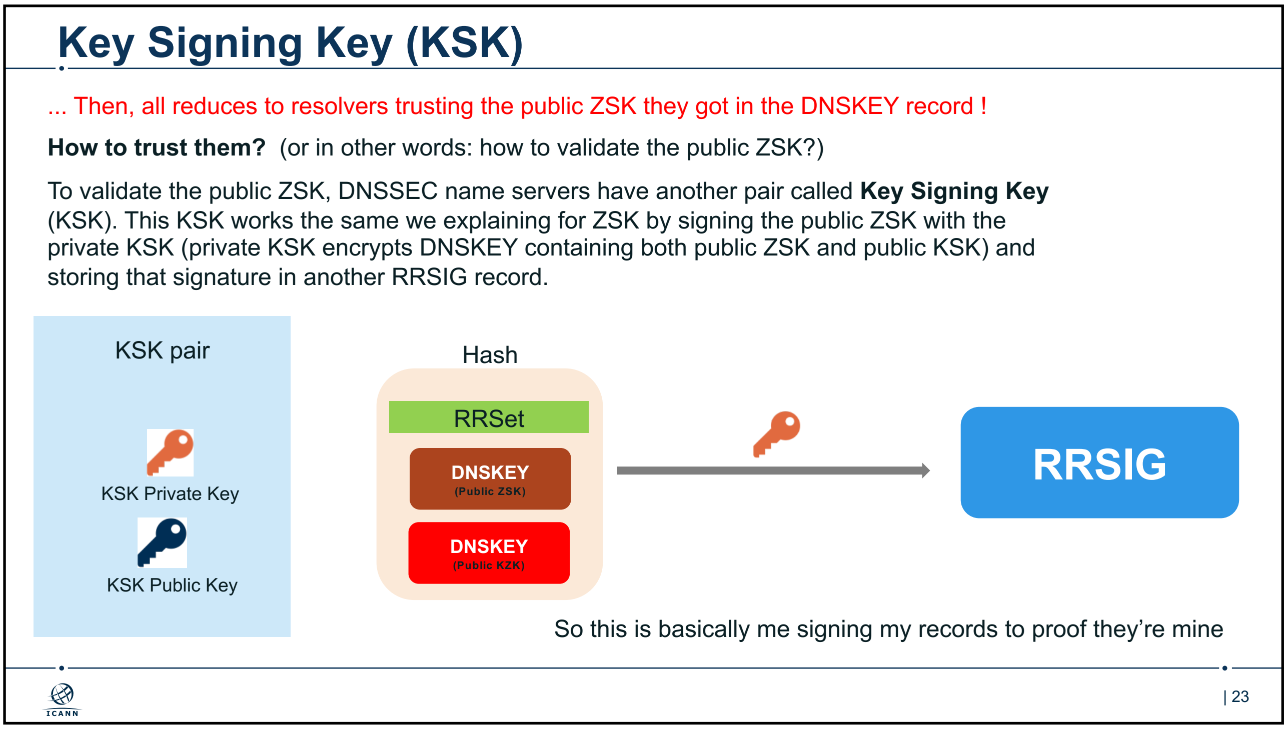
RRSet

RR (AAAA)

So this is basically the resolver confirming RRSet is mine

ICANN | 22

22



Key Signing Key (KSK)

... Then, all reduces to resolvers trusting the public ZSK they got in the DNSKEY record !

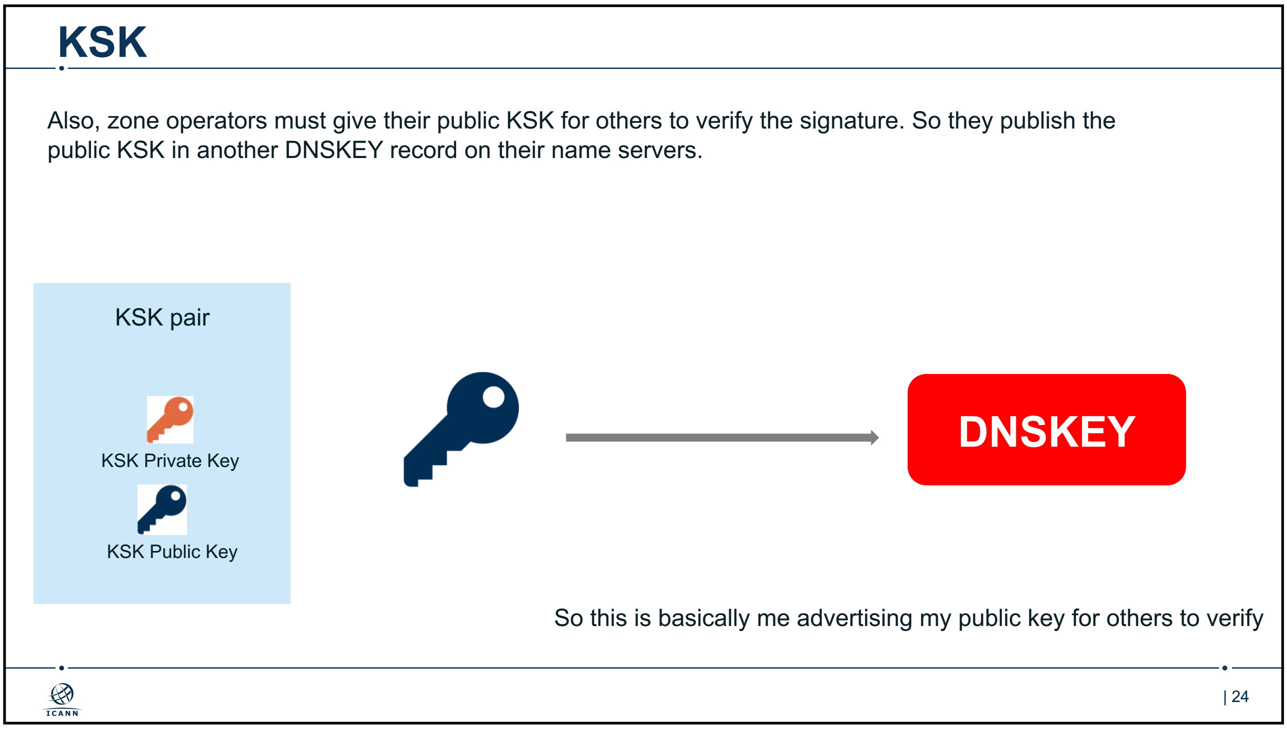
How to trust them? (or in other words: how to validate the public ZSK?)

To validate the public ZSK, DNSSEC name servers have another pair called **Key Signing Key** (KSK). This KSK works the same we explaining for ZSK by signing the public ZSK with the private KSK (private KSK encrypts DNSKEY containing both public ZSK and public KSK) and storing that signature in another RRSIG record.

So this is basically me signing my records to proof they're mine

| 23

23



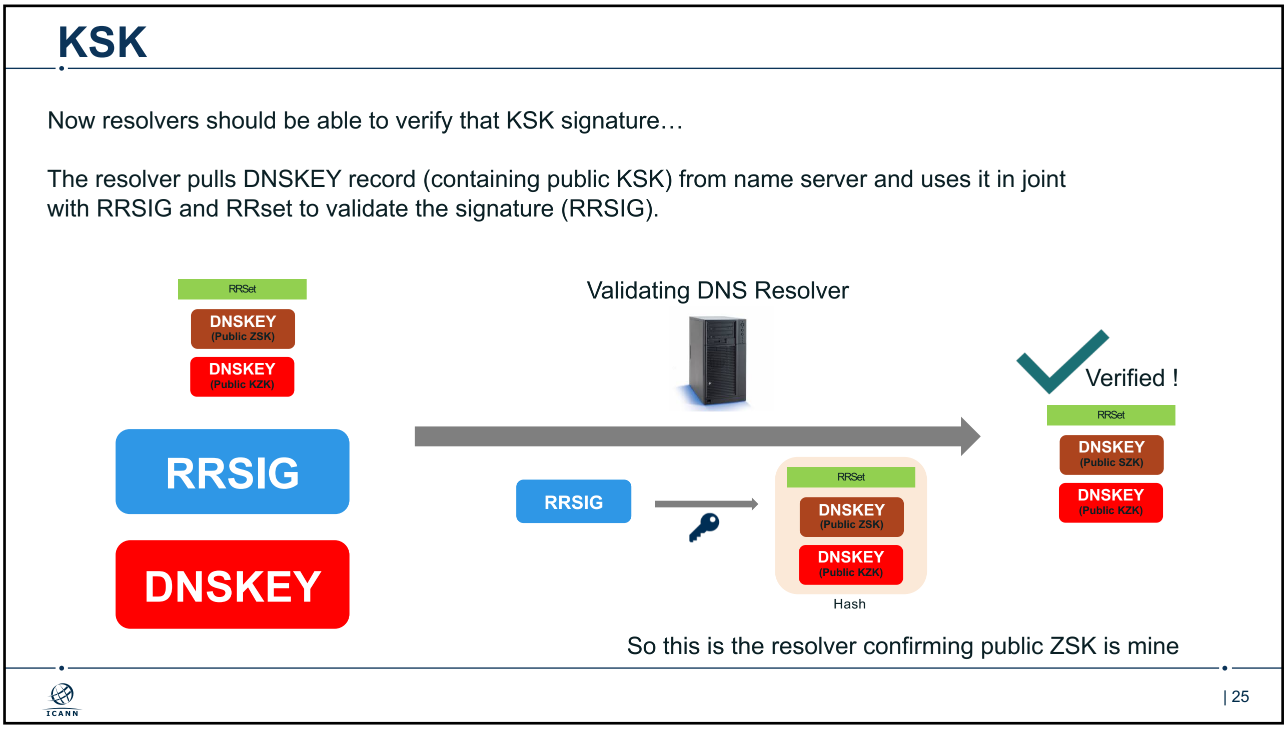
KSK

Also, zone operators must give their public KSK for others to verify the signature. So they publish the public KSK in another DNSKEY record on their name servers.

So this is basically me advertising my public key for others to verify

| 24

24

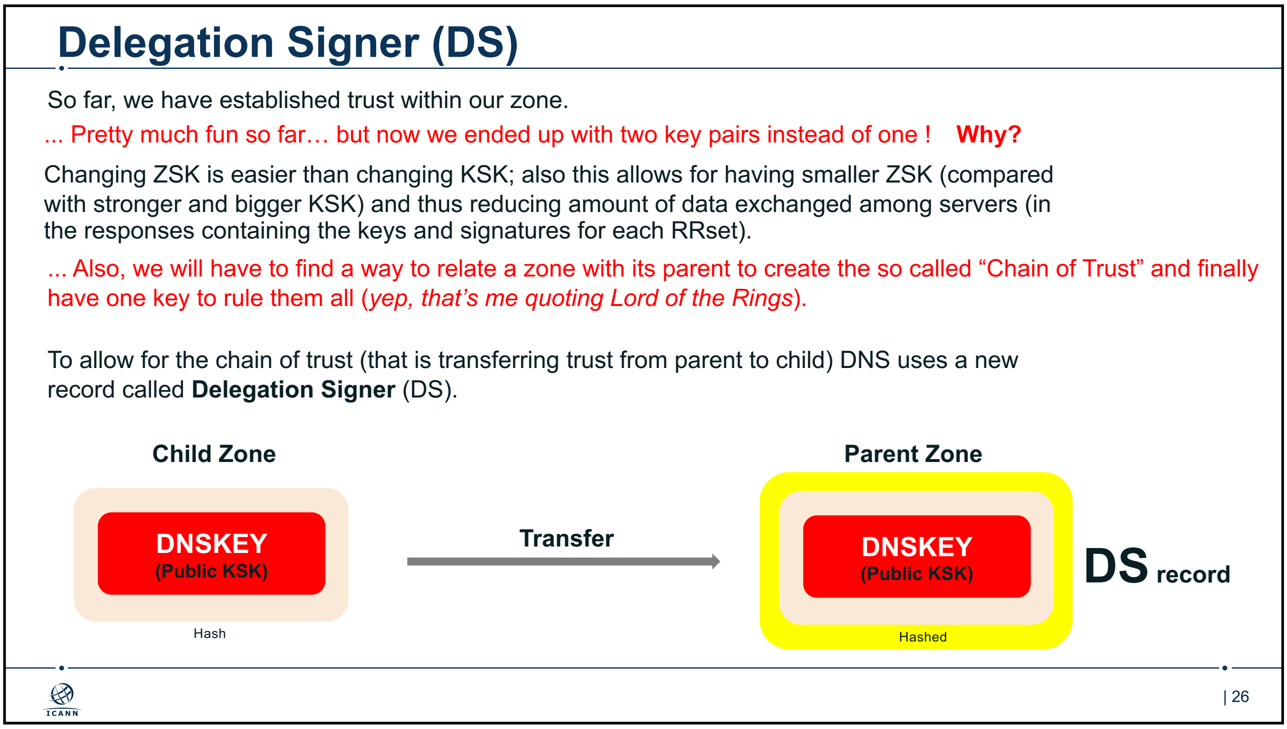


KSK

Now resolvers should be able to verify that KSK signature…

The resolver pulls DNSKEY record (containing public KSK) from name server and uses it in joint with RRSIG and RRset to validate the signature (RRSIG).

So this is the resolver confirming public ZSK is mine



Delegation Signer (DS)

So far, we have established trust within our zone.

... Pretty much fun so far… but now we ended up with two key pairs instead of one ! **Why?**

Changing ZSK is easier than changing KSK; also this allows for having smaller ZSK (compared with stronger and bigger KSK) and thus reducing amount of data exchanged among servers (in the responses containing the keys and signatures for each RRset).

... Also, we will have to find a way to relate a zone with its parent to create the so called “Chain of Trust” and finally have one key to rule them all (*yep, that’s me quoting Lord of the Rings*).

To allow for the chain of trust (that is transferring trust from parent to child) DNS uses a new record called **Delegation Signer** (DS).

Chain of Trust

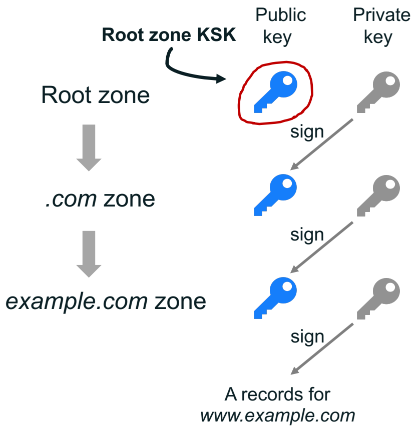
Finally, how do we trust DS record?

Well, we just sign DS record like we did with other RRsets, creating a corresponding RRSIG for the DS record in the parent.

We repeat the validation process and get to the parents public KSK... And again must go to that parent’s DS record to verify… on and on up to the DNS root.

Eventually, we get to the root and there's nothing up there (sadly no parent)… and so we must come with a solution to create a trust anchor for the root, a “one key to rule them all” … and here it comes a solution implemented since 2010 called:

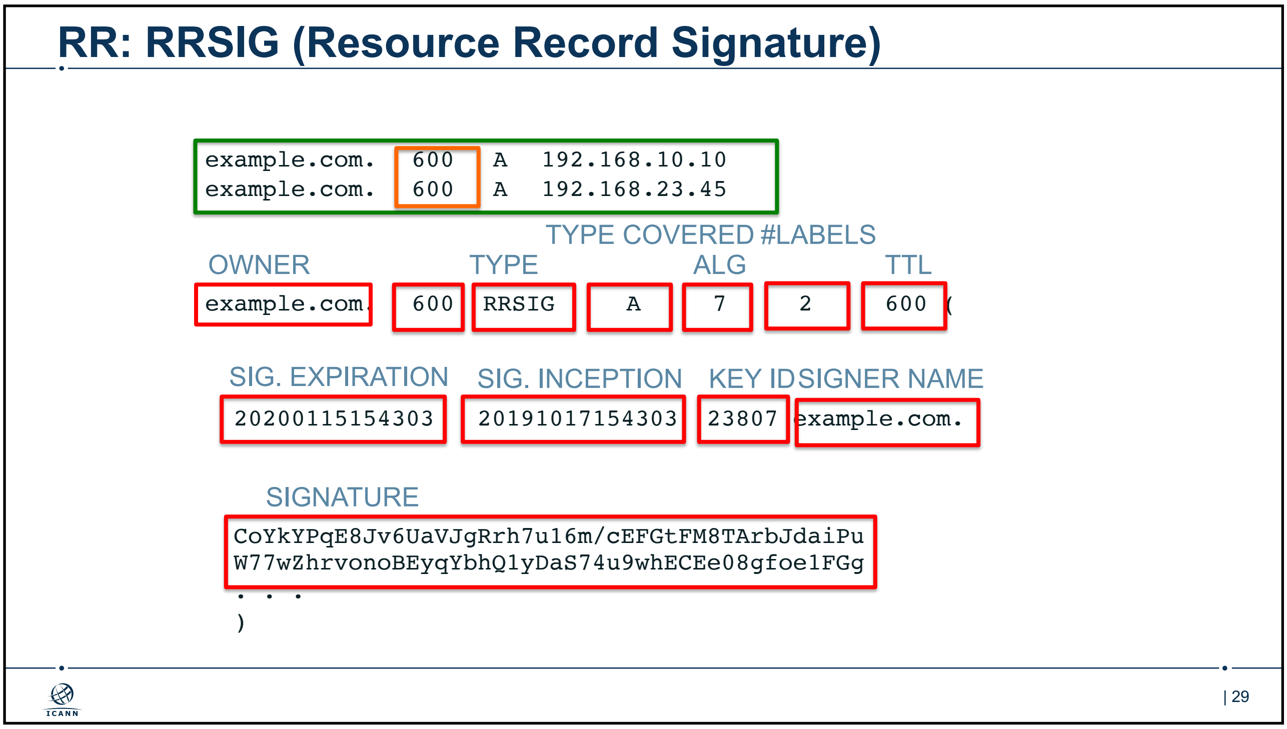
The Root Signing Ceremony



RR: DNSKEY

```
                           PROTOCOL
 OWNER             TYPE   FLAGS    ALGORITHM
example.com.  43200 DNSKEY  256   3   8 (
  AwEAAbinasY+k/9xD4MBBa3QvhjuOHIpe319SFbWYIRj   PUBLIC KEY
  /nbmVZfJnSw7By1cV3Tm7ZlLqNbcB86nVFMSQ3JjOFMr   (BASE64)
  ....) ; ZSK; key id = 23807   KEY ID
```

- FLAGS determines the usage of the key
- PROTOCOL is always 3 (DNSSEC)
- ALGORITHM can be (3: DSA/SHA-1, 5: RSA/SHA1, 8: RSA/SHA-256, 12: ECC-GOST)
  - http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml



RR: RRSIG (Resource Record Signature)

```
example.com.  600  A  192.168.10.10
example.com.  600  A  192.168.23.45
```

OWNER, TYPE, TYPE COVERED, ALG, #LABELS, TTL

```
example.com.  600  RRSIG  A  7  2  600 (
```

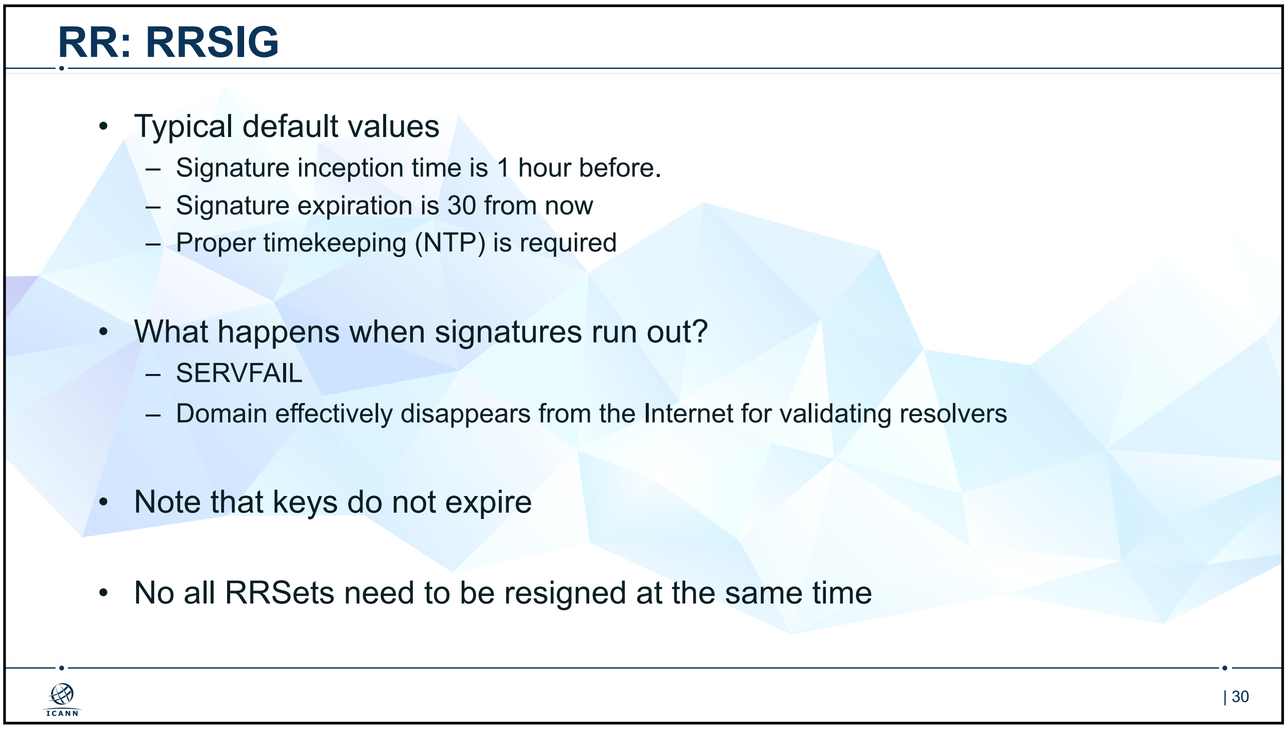
SIG. EXPIRATION, SIG. INCEPTION, KEY ID, SIGNER NAME

```
20200115154303  20191017154303  23807  example.com.
```

SIGNATURE

```
CoYkYPqE8Jv6UaVJgRrh7u16m/cEFGtFM8TArbJdaiPu
W77wZhrvonoBEyqYbhQ1yDaS74u9whECEe08gfoe1FGg
. . .
)
```

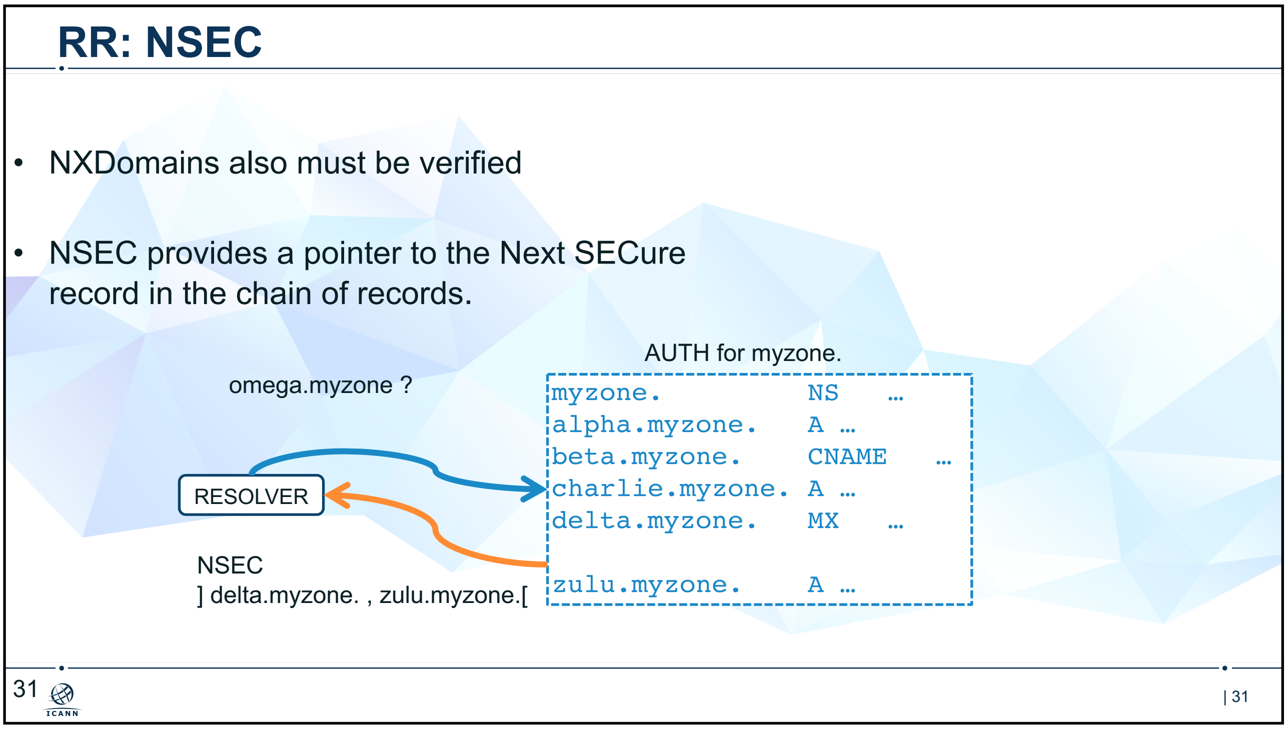
29



RR: RRSIG

- Typical default values
  - Signature inception time is 1 hour before.
  - Signature expiration is 30 from now
  - Proper timekeeping (NTP) is required
- What happens when signatures run out?
  - SERVFAIL
  - Domain effectively disappears from the Internet for validating resolvers
- Note that keys do not expire
- No all RRSets need to be resigned at the same time

30



RR: NSEC

- NXDomains also must be verified
- NSEC provides a pointer to the Next SECure record in the chain of records.

omega.myzone ?

RESOLVER

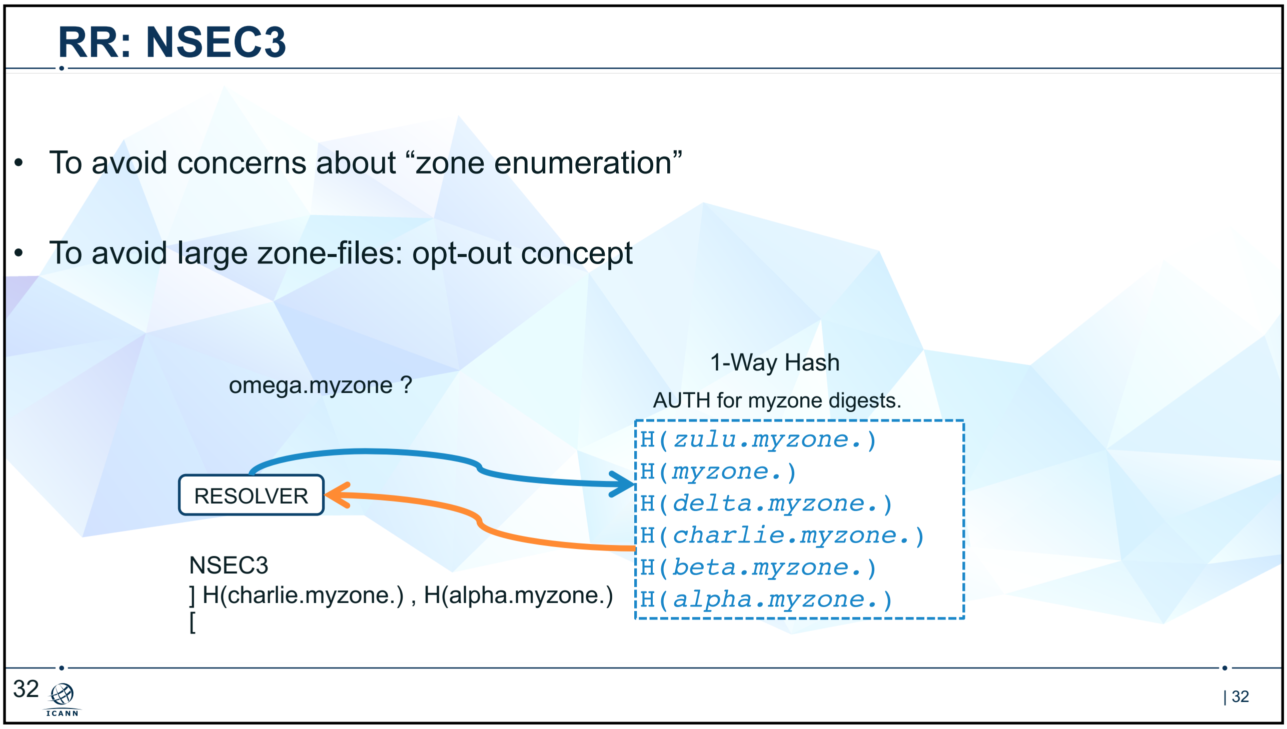
AUTH for myzone.

```
myzone.          NS    …
alpha.myzone.    A …
beta.myzone.     CNAME    …
charlie.myzone.  A …
delta.myzone.    MX    …

zulu.myzone.     A …
```

NSEC
] delta.myzone. , zulu.myzone.[

31



RR: NSEC3

- To avoid concerns about “zone enumeration”
- To avoid large zone-files: opt-out concept

omega.myzone ?

RESOLVER

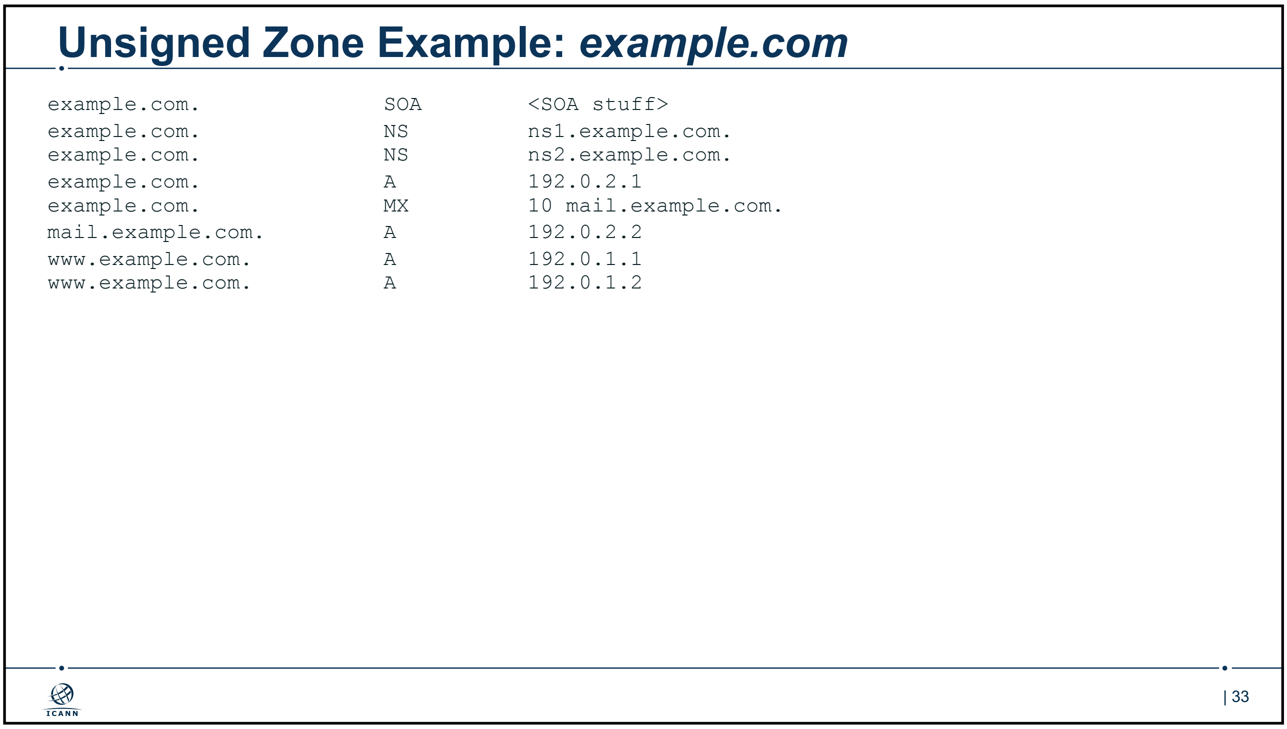
1-Way Hash

AUTH for myzone digests.

```
H(zulu.myzone.)
H(myzone.)
H(delta.myzone.)
H(charlie.myzone.)
H(beta.myzone.)
H(alpha.myzone.)
```

NSEC3
] H(charlie.myzone.) , H(alpha.myzone.)
[

32

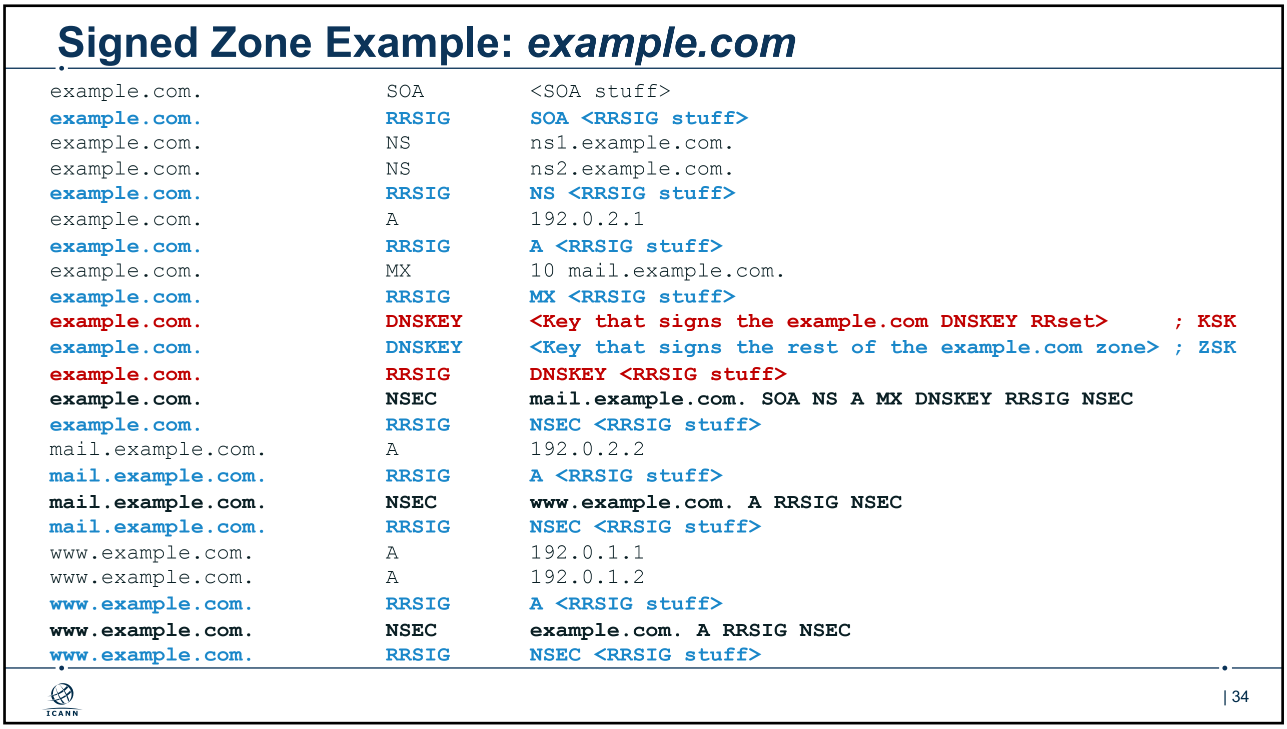


Unsigned Zone Example: *example.com*

```
example.com.              SOA       <SOA stuff>
example.com.              NS        ns1.example.com.
example.com.              NS        ns2.example.com.
example.com.              A         192.0.2.1
example.com.              MX        10 mail.example.com.
mail.example.com.         A         192.0.2.2
www.example.com.          A         192.0.1.1
www.example.com.          A         192.0.1.2
```

ICANN | 33

33

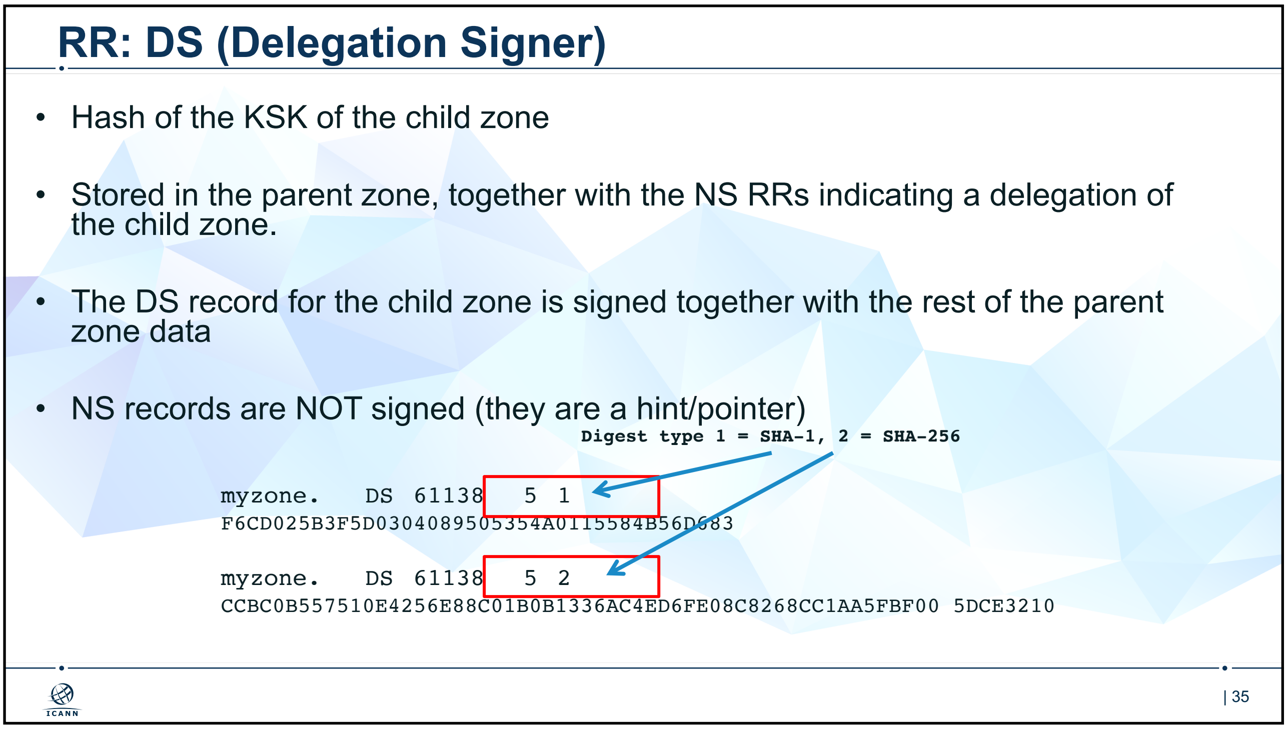


Signed Zone Example: *example.com*

```
example.com.              SOA       <SOA stuff>
example.com.              RRSIG     SOA <RRSIG stuff>
example.com.              NS        ns1.example.com.
example.com.              NS        ns2.example.com.
example.com.              RRSIG     NS <RRSIG stuff>
example.com.              A         192.0.2.1
example.com.              RRSIG     A <RRSIG stuff>
example.com.              MX        10 mail.example.com.
example.com.              RRSIG     MX <RRSIG stuff>
example.com.              DNSKEY    <Key that signs the example.com DNSKEY RRset>     ; KSK
example.com.              DNSKEY    <Key that signs the rest of the example.com zone> ; ZSK
example.com.              RRSIG     DNSKEY <RRSIG stuff>
example.com.              NSEC      mail.example.com. SOA NS A MX DNSKEY RRSIG NSEC
example.com.              RRSIG     NSEC <RRSIG stuff>
mail.example.com.         A         192.0.2.2
mail.example.com.         RRSIG     A <RRSIG stuff>
mail.example.com.         NSEC      www.example.com. A RRSIG NSEC
mail.example.com.         RRSIG     NSEC <RRSIG stuff>
www.example.com.          A         192.0.1.1
www.example.com.          A         192.0.1.2
www.example.com.          RRSIG     A <RRSIG stuff>
www.example.com.          NSEC      example.com. A RRSIG NSEC
www.example.com.          RRSIG     NSEC <RRSIG stuff>
```

ICANN | 34

34



RR: 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)

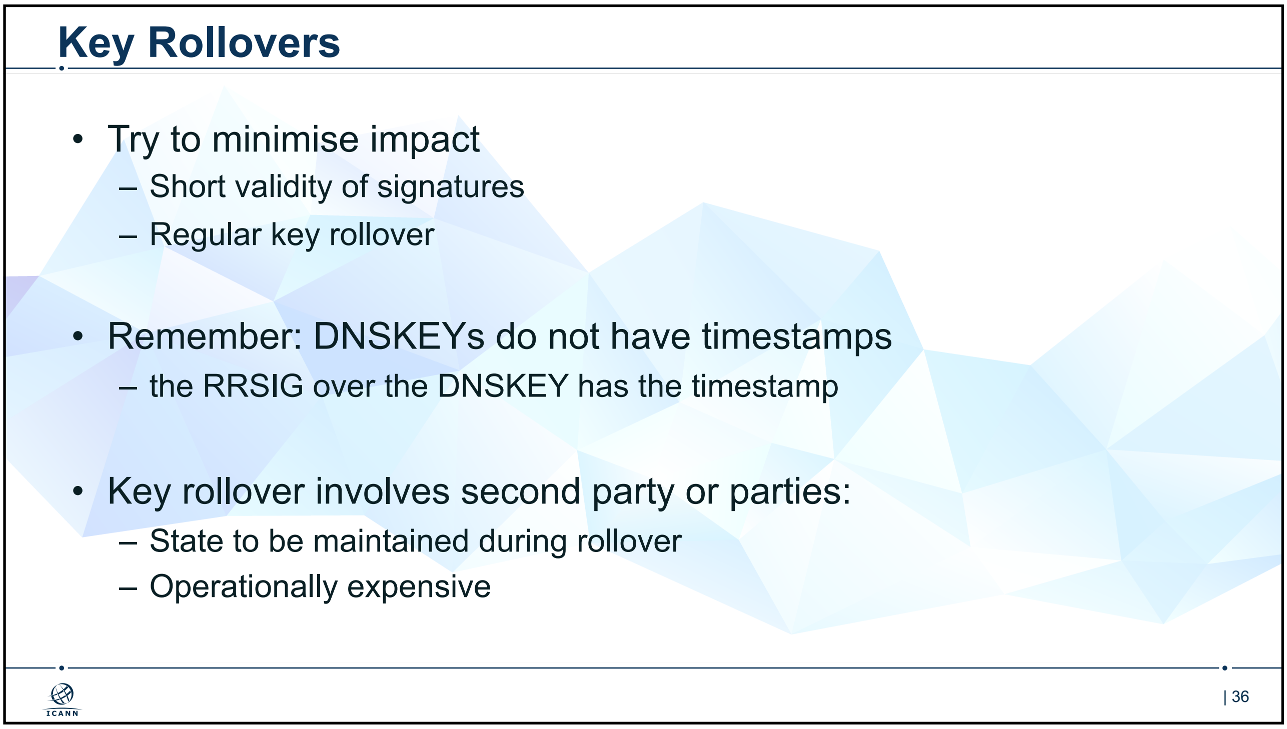
Digest type 1 = SHA-1, 2 = SHA-256

```
myzone.   DS  61138   5  1
F6CD025B3F5D0304089505354A0115584B56D683

myzone.   DS  61138   5  2
CCBC0B557510E4256E88C01B0B1336AC4ED6FE08C8268CC1AA5FBF00 5DCE3210
```

| 35

35

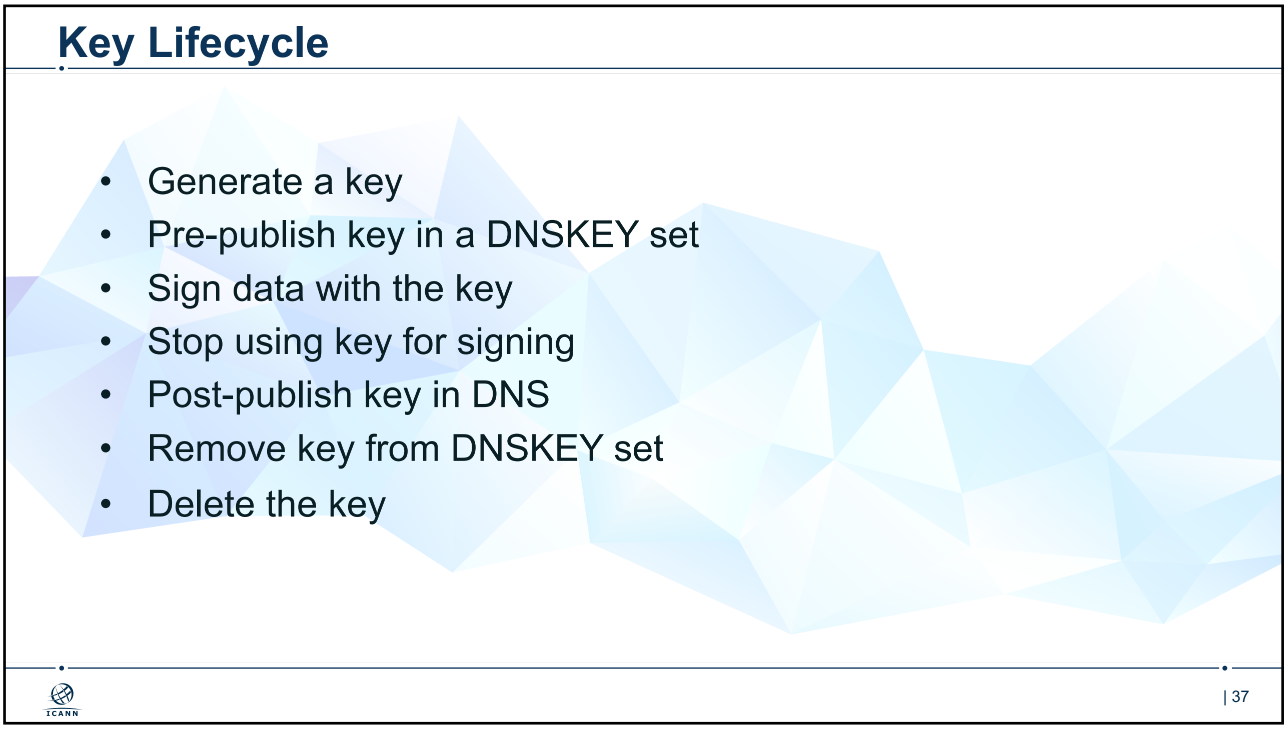


Key Rollovers

- Try to minimise impact
  - Short validity of signatures
  - Regular key rollover
- Remember: DNSKEYs do not have timestamps
  - the RRSIG over the DNSKEY has the timestamp
- Key rollover involves second party or parties:
  - State to be maintained during rollover
  - Operationally expensive

| 36

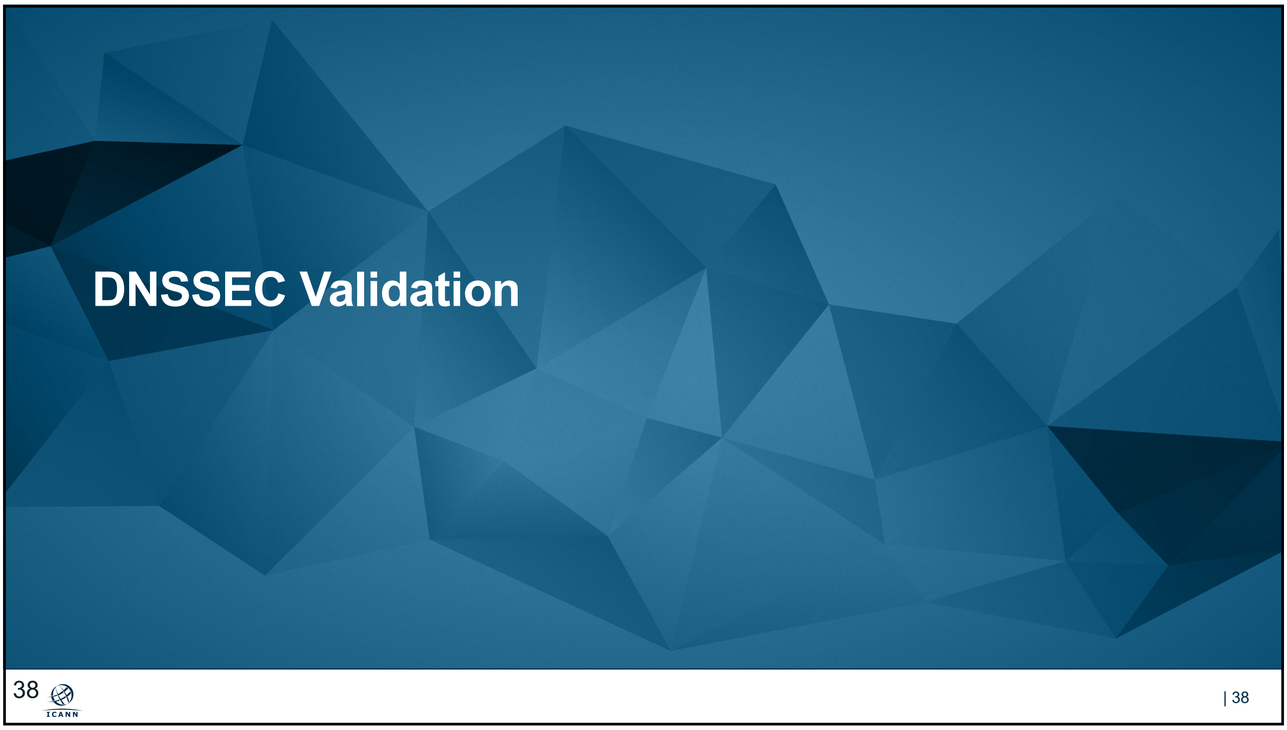
36



Key Lifecycle

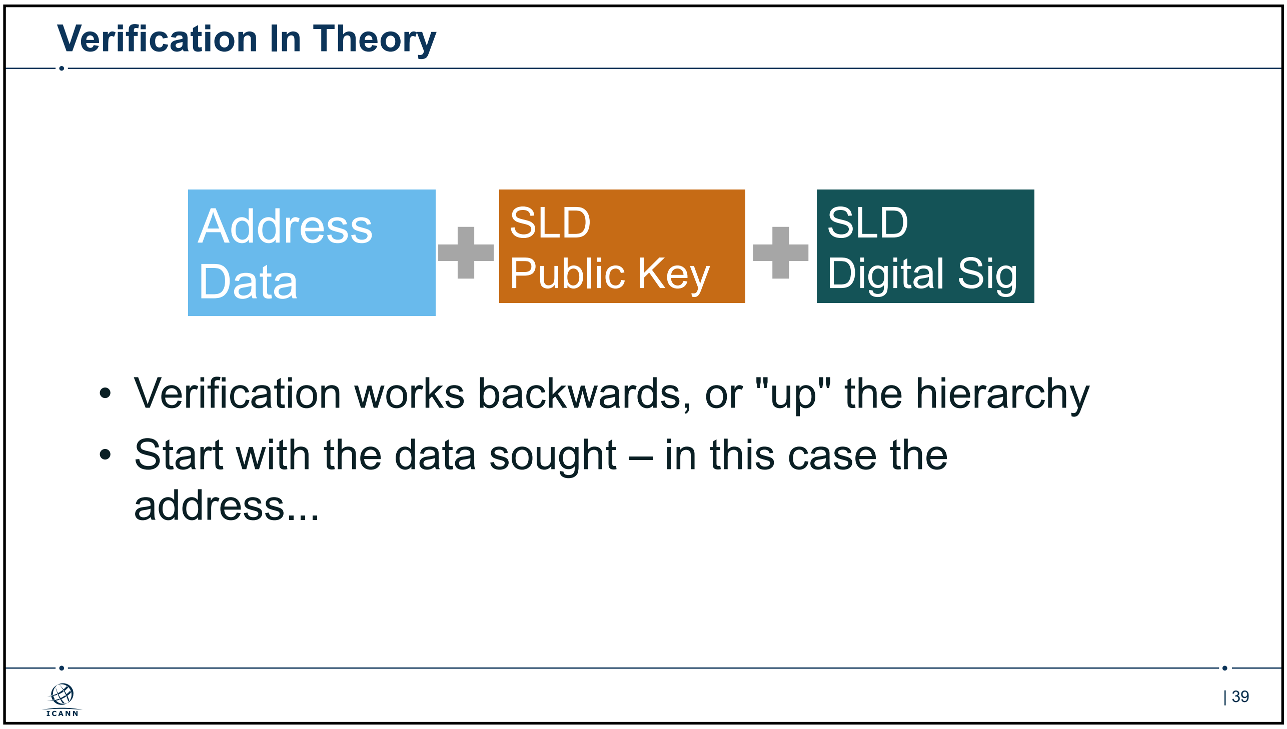
- Generate a key
- Pre-publish key in a DNSKEY set
- Sign data with the key
- Stop using key for signing
- Post-publish key in DNS
- Remove key from DNSKEY set
- Delete the key

37



DNSSEC Validation

38



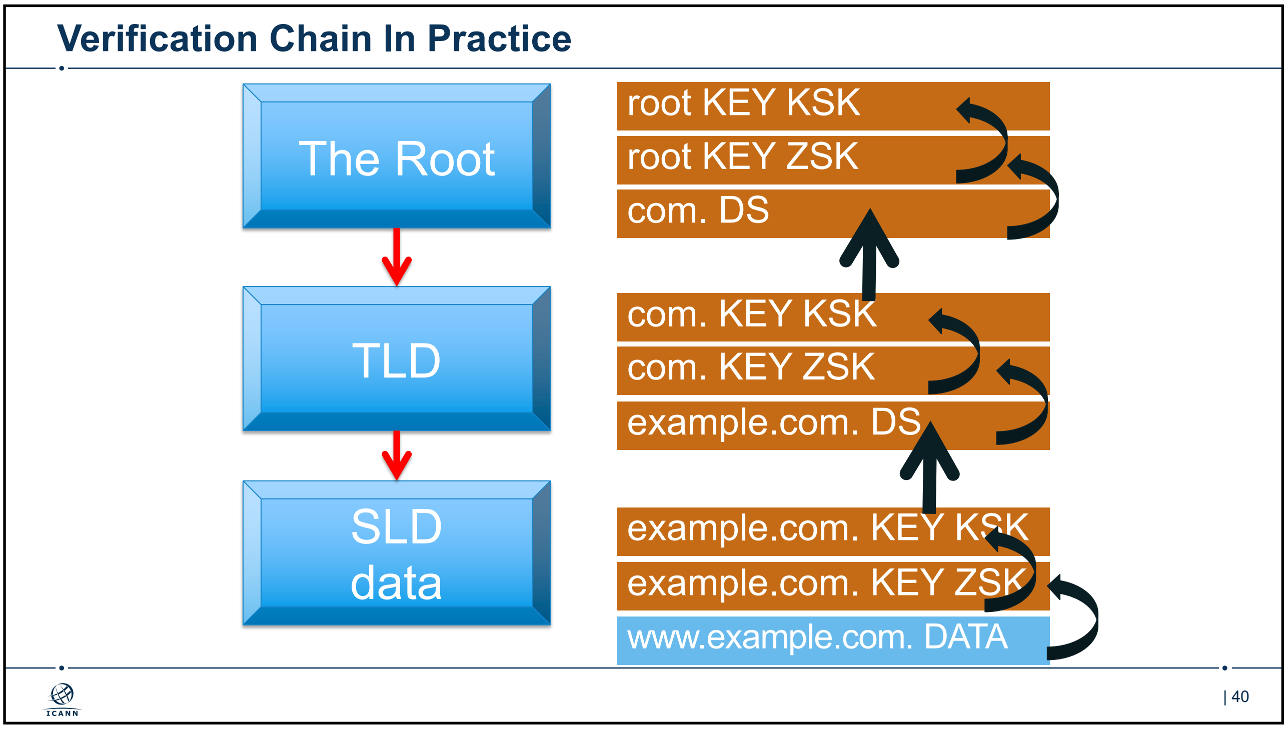
Verification In Theory

Address Data + SLD Public Key + SLD Digital Sig

- Verification works backwards, or "up" the hierarchy
- Start with the data sought – in this case the address...

| 39

39



Verification Chain In Practice

The Root

TLD

SLD data

root KEY KSK

root KEY ZSK

com. DS

com. KEY KSK

com. KEY ZSK

example.com. DS

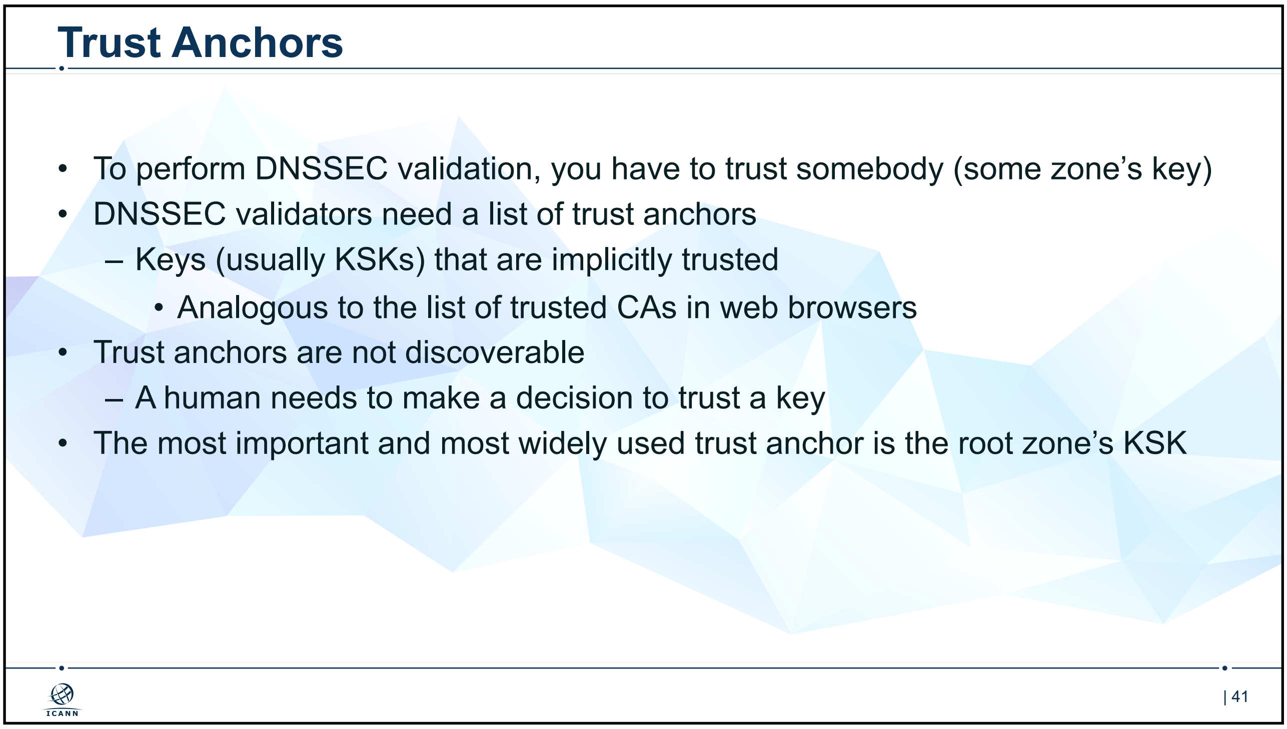
example.com. KEY KSK

example.com. KEY ZSK

www.example.com. DATA

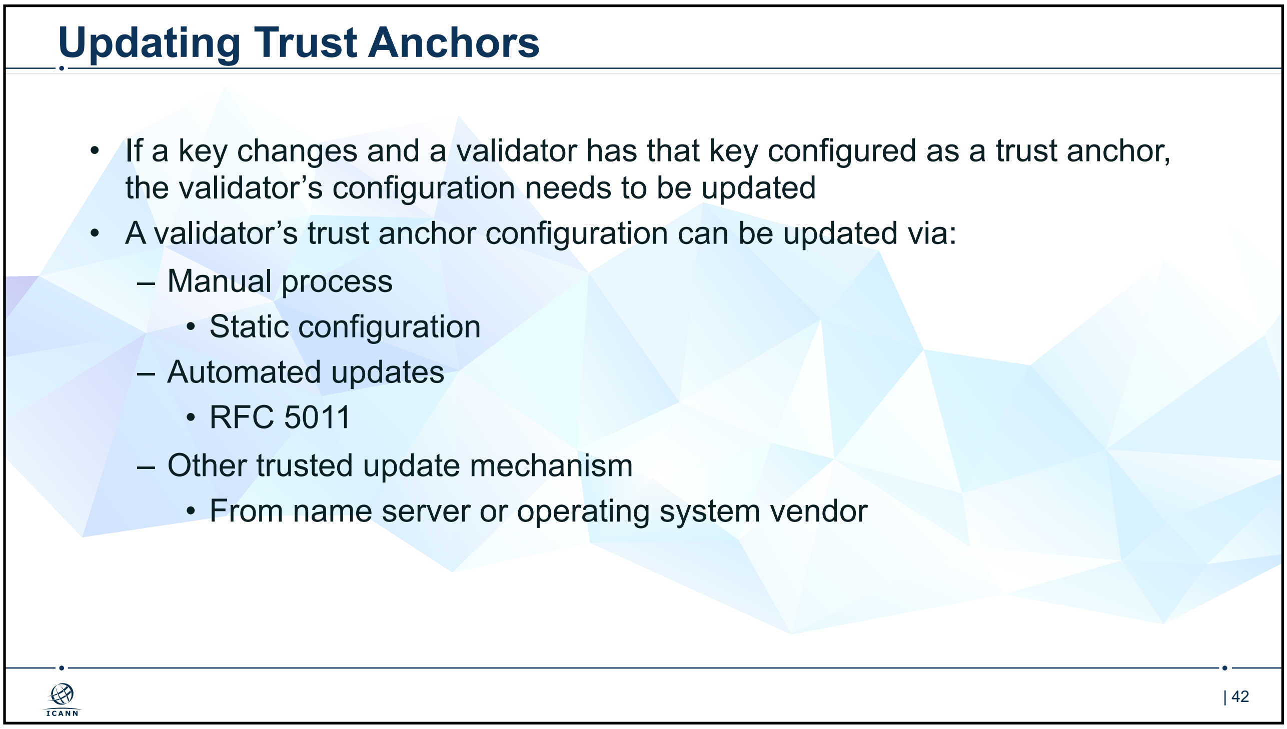
| 40

40



Trust Anchors

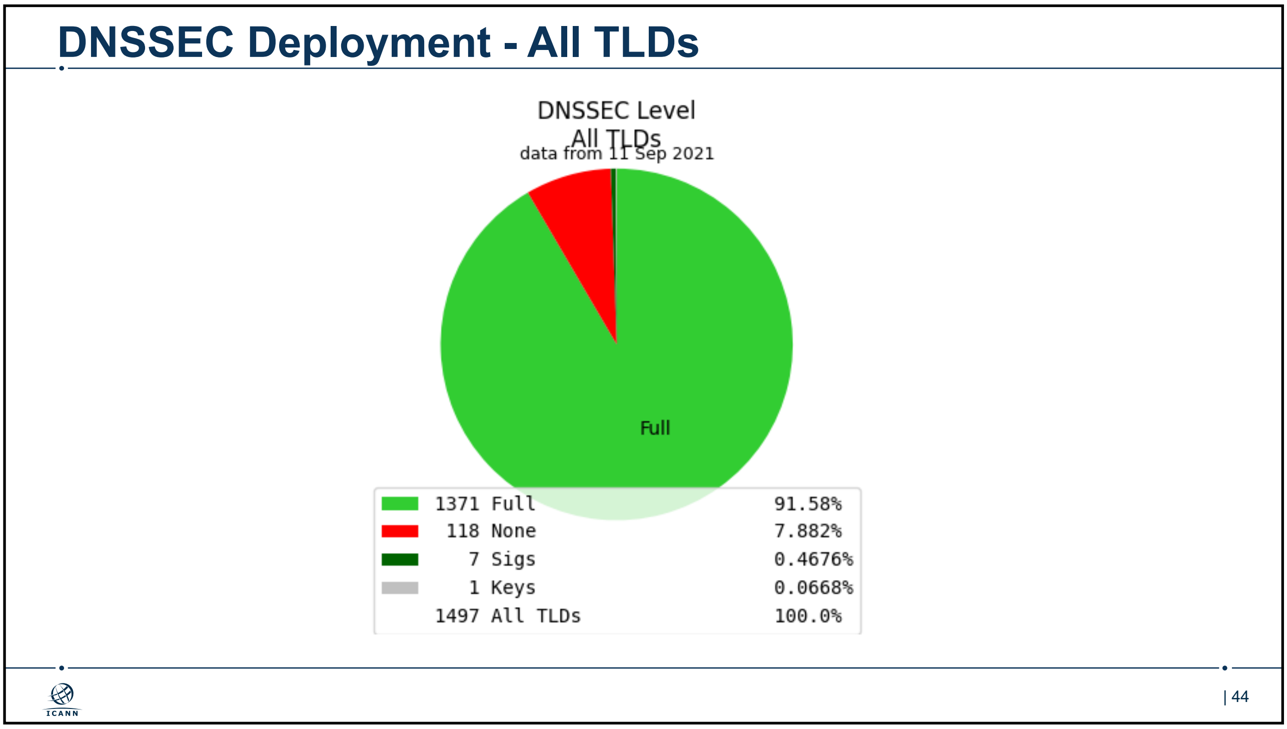
- To perform DNSSEC validation, you have to trust somebody (some zone's key)
- DNSSEC validators need a list of trust anchors
  - Keys (usually KSKs) that are implicitly trusted
    - Analogous to the list of trusted CAs in web browsers
- Trust anchors are not discoverable
  - A human needs to make a decision to trust a key
- The most important and most widely used trust anchor is the root zone's KSK



Updating Trust Anchors

- If a key changes and a validator has that key configured as a trust anchor, the validator's configuration needs to be updated
- A validator's trust anchor configuration can be updated via:
  - Manual process
    - Static configuration
  - Automated updates
    - RFC 5011
  - Other trusted update mechanism
    - From name server or operating system vendor

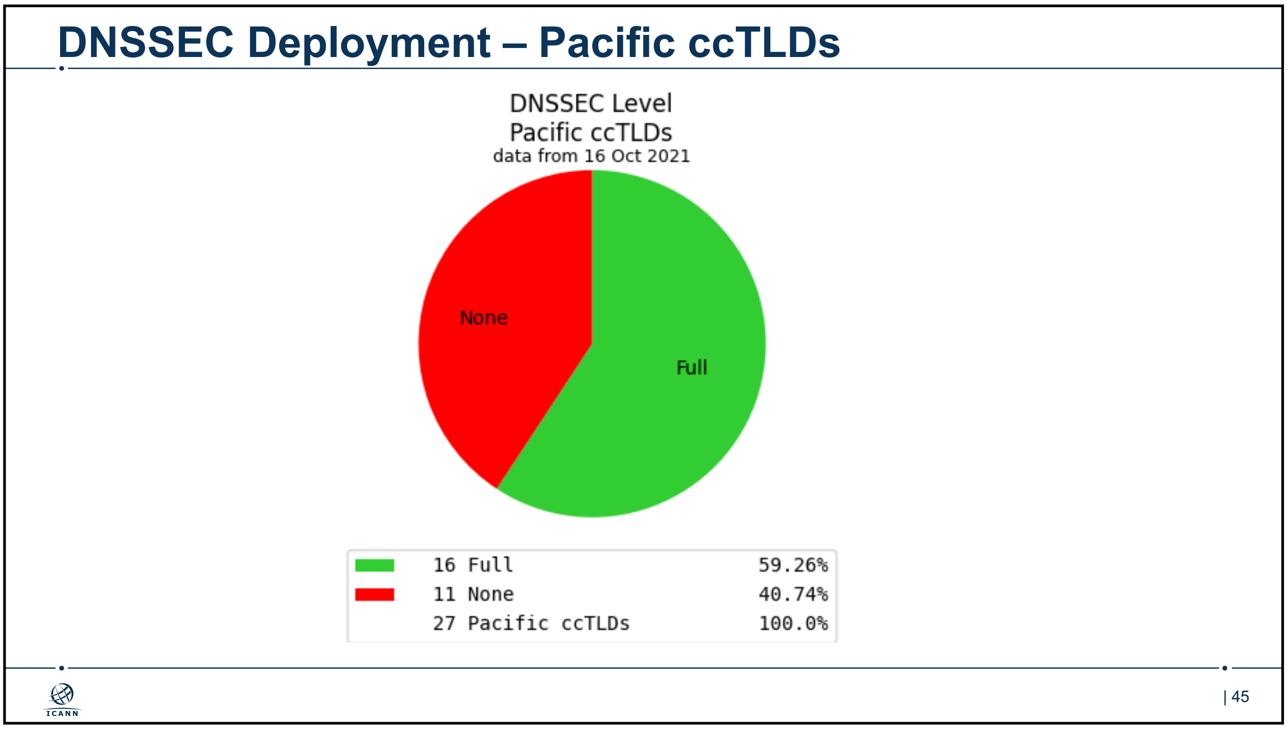
DNSSEC Deployment



DNSSEC Deployment - All TLDs

DNSSEC Level
All TLDs
data from 11 Sep 2021

🟩	1371	Full	91.58%
🟥	118	None	7.882%
🟩	7	Sigs	0.4676%
▨	1	Keys	0.0668%
	1497	All TLDs	100.0%



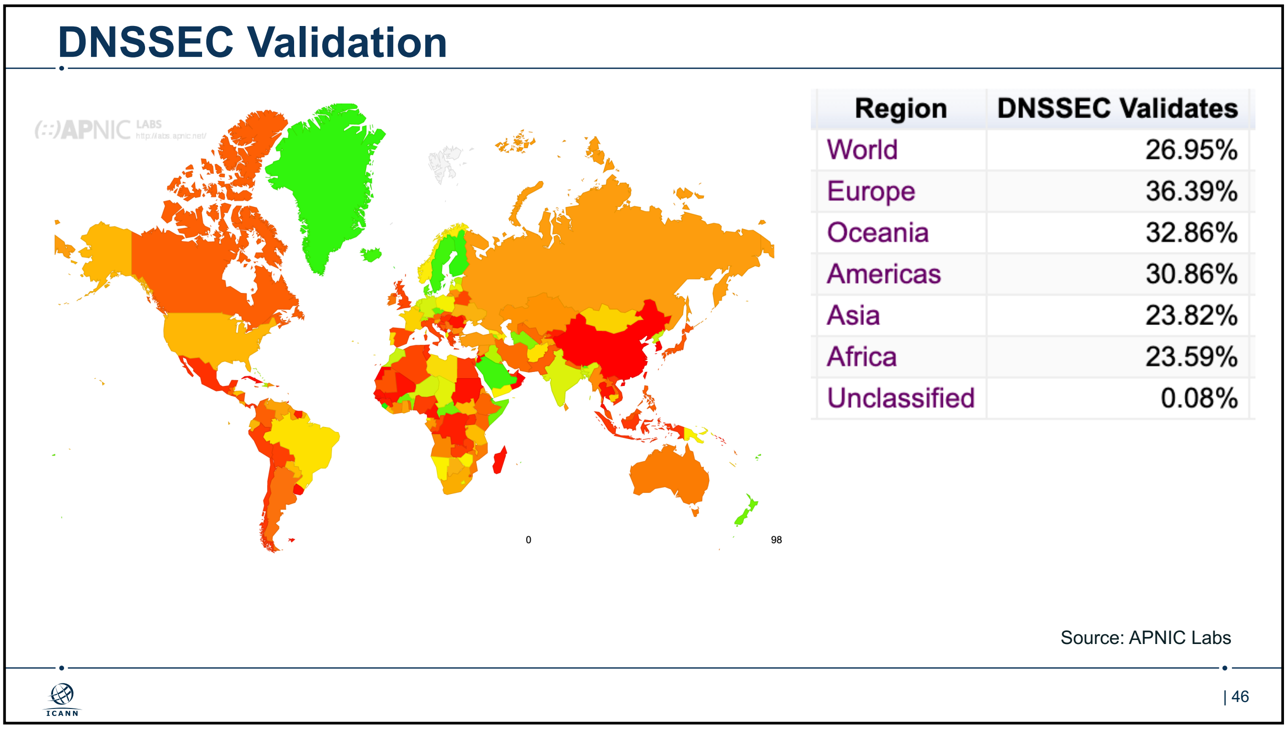
DNSSEC Deployment – Pacific ccTLDs

DNSSEC Level
Pacific ccTLDs
data from 16 Oct 2021

🟩	16 Full	59.26%
🟥	11 None	40.74%
	27 Pacific ccTLDs	100.0%

| 45

45



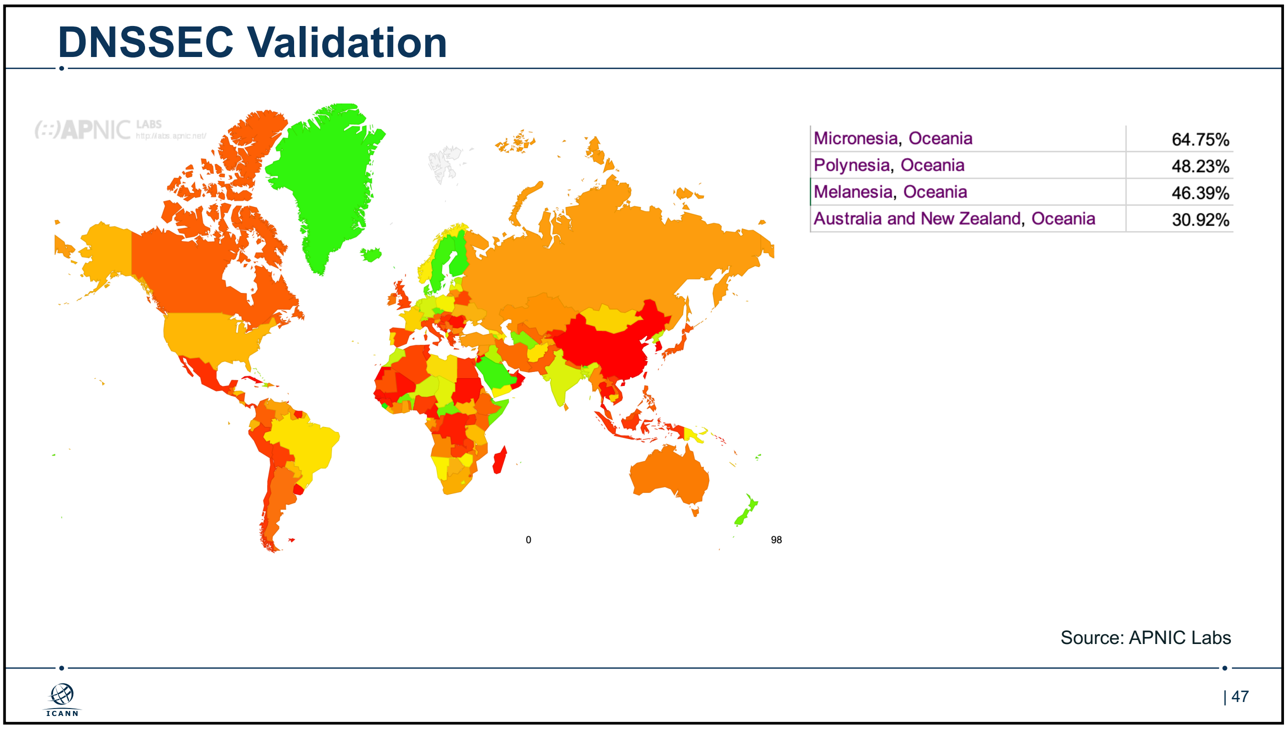
DNSSEC Validation

Region	DNSSEC Validates
World	26.95%
Europe	36.39%
Oceania	32.86%
Americas	30.86%
Asia	23.82%
Africa	23.59%
Unclassified	0.08%

Source: APNIC Labs

| 46

46

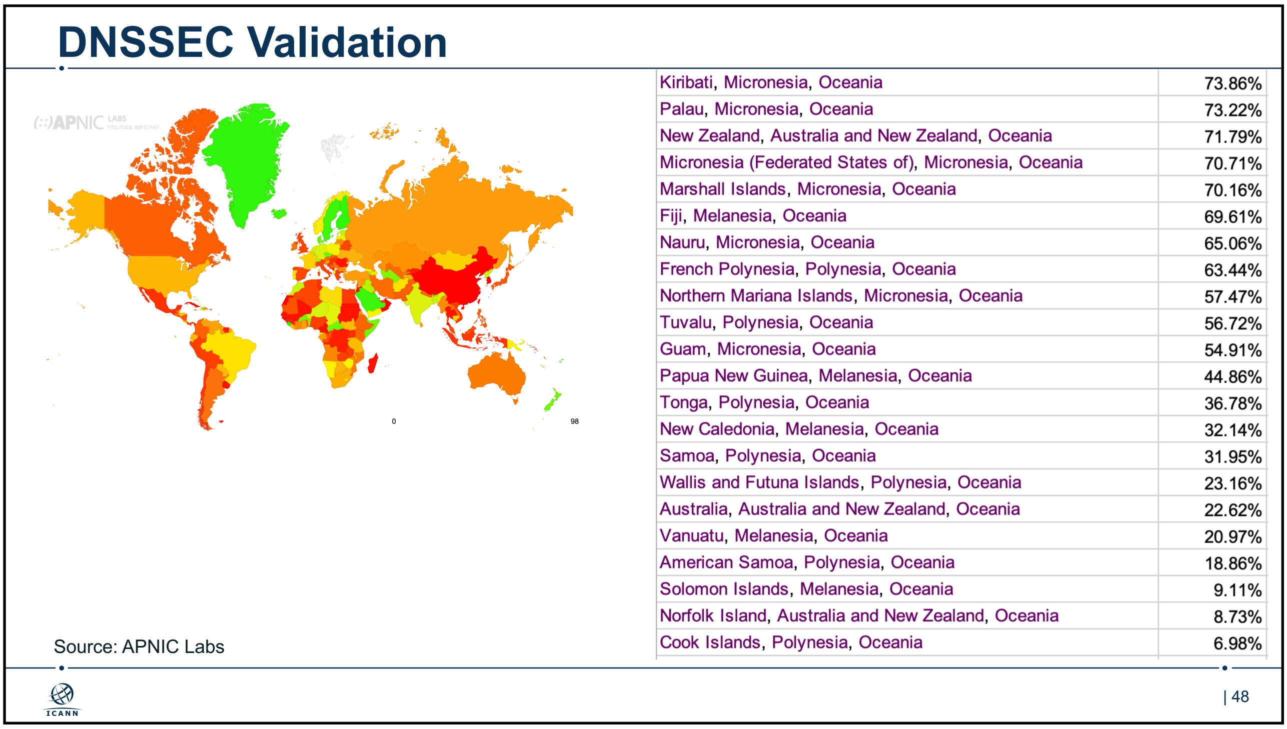


DNSSEC Validation

Micronesia, Oceania	64.75%
Polynesia, Oceania	48.23%
Melanesia, Oceania	46.39%
Australia and New Zealand, Oceania	30.92%

Source: APNIC Labs

47



DNSSEC Validation

Kiribati, Micronesia, Oceania	73.86%
Palau, Micronesia, Oceania	73.22%
New Zealand, Australia and New Zealand, Oceania	71.79%
Micronesia (Federated States of), Micronesia, Oceania	70.71%
Marshall Islands, Micronesia, Oceania	70.16%
Fiji, Melanesia, Oceania	69.61%
Nauru, Micronesia, Oceania	65.06%
French Polynesia, Polynesia, Oceania	63.44%
Northern Mariana Islands, Micronesia, Oceania	57.47%
Tuvalu, Polynesia, Oceania	56.72%
Guam, Micronesia, Oceania	54.91%
Papua New Guinea, Melanesia, Oceania	44.86%
Tonga, Polynesia, Oceania	36.78%
New Caledonia, Melanesia, Oceania	32.14%
Samoa, Polynesia, Oceania	31.95%
Wallis and Futuna Islands, Polynesia, Oceania	23.16%
Australia, Australia and New Zealand, Oceania	22.62%
Vanuatu, Melanesia, Oceania	20.97%
American Samoa, Polynesia, Oceania	18.86%
Solomon Islands, Melanesia, Oceania	9.11%
Norfolk Island, Australia and New Zealand, Oceania	8.73%
Cook Islands, Polynesia, Oceania	6.98%

Source: APNIC Labs

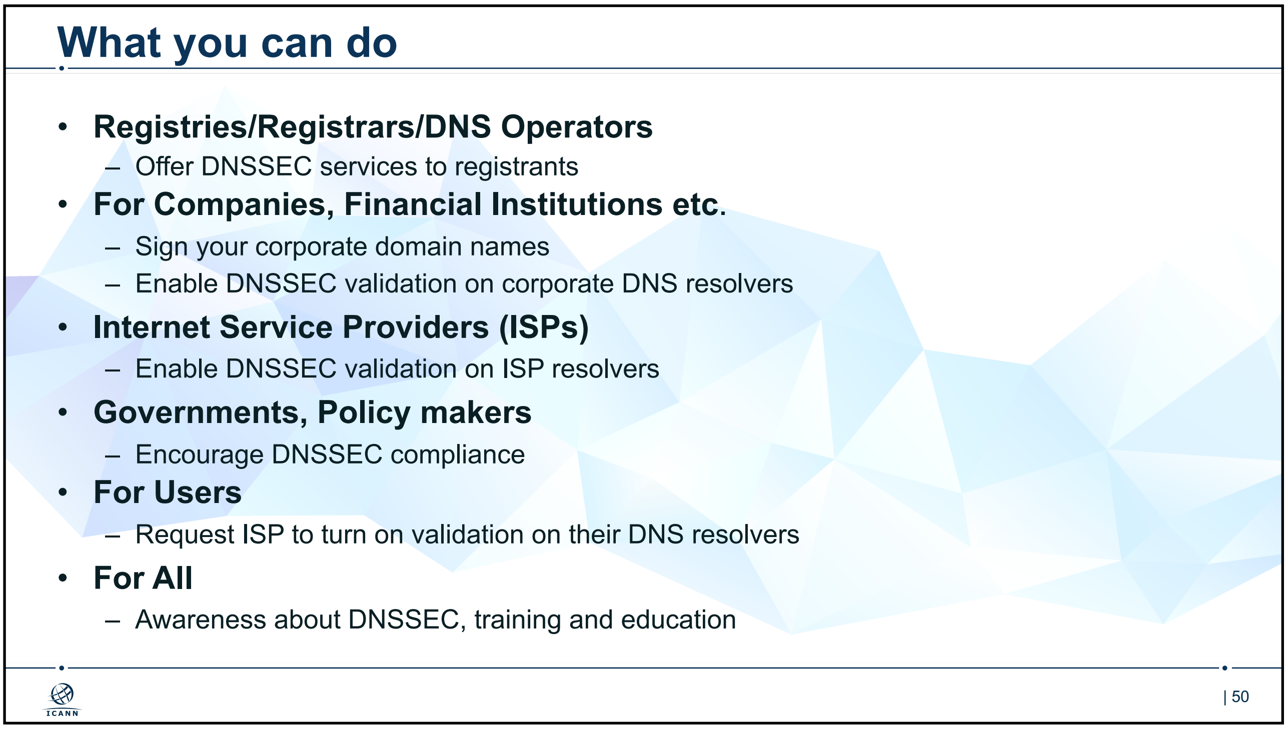
48

What you can do

49

| 49

49



What you can do

- **Registries/Registrars/DNS Operators**
  - Offer DNSSEC services to registrants
- **For Companies, Financial Institutions etc.**
  - Sign your corporate domain names
  - Enable DNSSEC validation on corporate DNS resolvers
- **Internet Service Providers (ISPs)**
  - Enable DNSSEC validation on ISP resolvers
- **Governments, Policy makers**
  - Encourage DNSSEC compliance
- **For Users**
  - Request ISP to turn on validation on their DNS resolvers
- **For All**
  - Awareness about DNSSEC, training and education

| 50

50

Engage with ICANN – Thank You and Questions

One World, One Internet

Visit us at **icann.org**

Email: champika.wijayatunga@icann.org

- @icann
- facebook.com/icannorg
- youtube.com/icannnews
- flickr.com/icann
- linkedin/company/icann
- slideshare/icannpresentations
- soundcloud/icann

| 51

51