DNSSEC

Understanding DNSSEC…

ICANN Office of the CTO

Sept 2023

Introduction to DNSSEC

What Is DNSSEC?

DNSSEC stands for **Domain Name System (DNS) Security Extensions.**

- DNSSEC is a protocol that is currently being deployed to secure the DNS.
- DNSSEC adds security to the DNS by incorporating public key cryptography into the DNS hierarchy, resulting in a single, open, global Public Key Infrastructure (PKI) for domain names.
- DNSSEC is the result of over two decade of community-based, open standards development.
- Specified in RFCs 4033, 4034, 4035 and 5155

Timeline For Securing DNS With DNSSEC

- 1993: Discussion of secure DNS begins
- 1994: First draft of possible standard published
- 1997: RFC 2065 published
  DNSSEC is an IETF standard
- 1999: RFC 2535 published
  DNSSEC standard is revised
- 2005: Total rewrite of standards published
  RFCs 4033, 4034 and 4035
- July 2010: Root zone signed
- March 2011: *.com* zone signed
- 2012-: New gTLDs required to be signed with DNSSEC
- 2018: Root zone KSK rollover

What DNSSEC Does

- DNSSEC uses public-key cryptography and digital signatures to provide:

  Data Origin Authenticity

  - "Did this response really come from the *example.com* zone?"

  Data Integrity

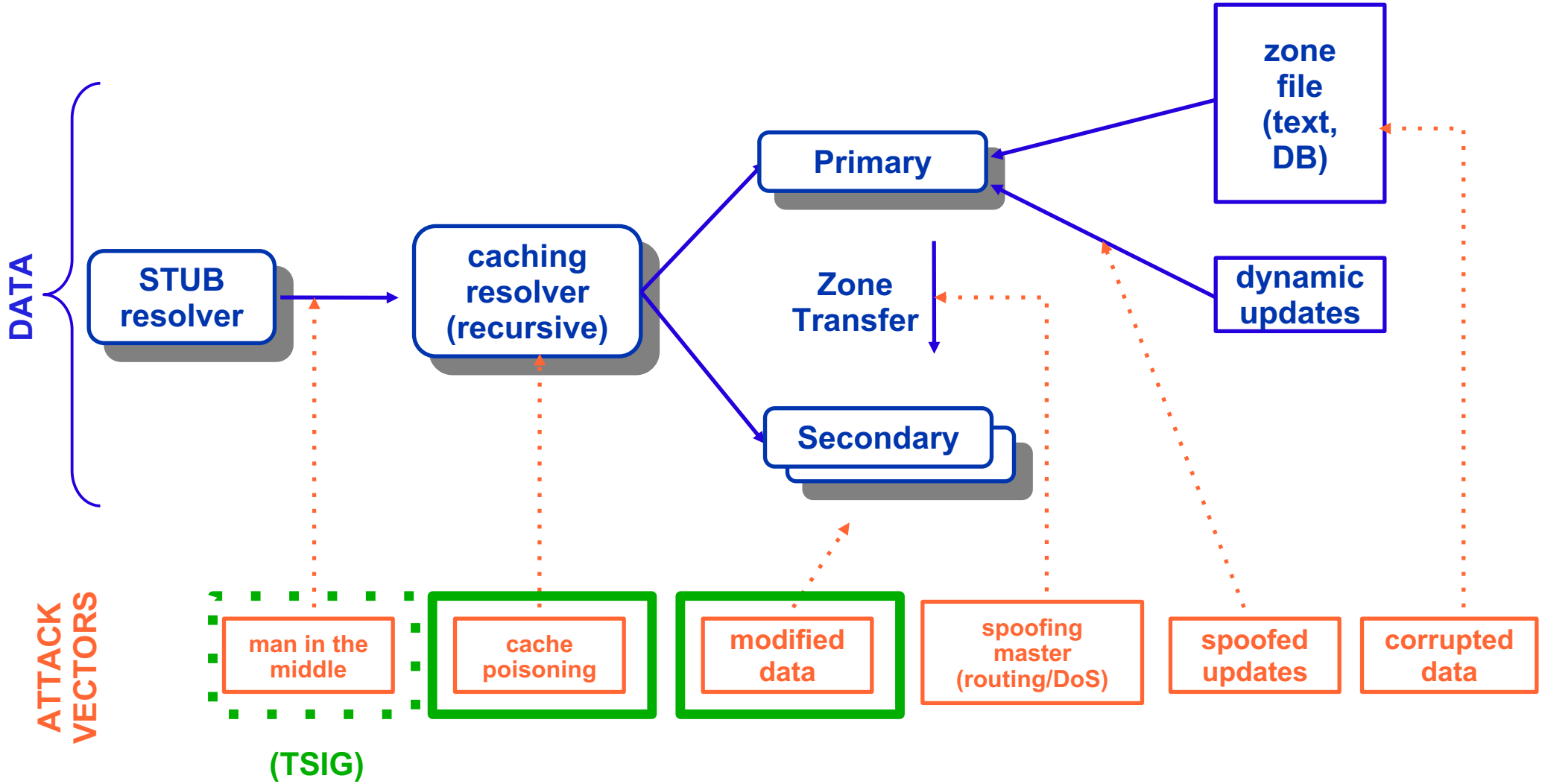
  - "Did an attacker (e.g., a man in the middle) modify the data in this response since the data was originally signed?"

- DNSSEC offers protection against spoofing of DNS data

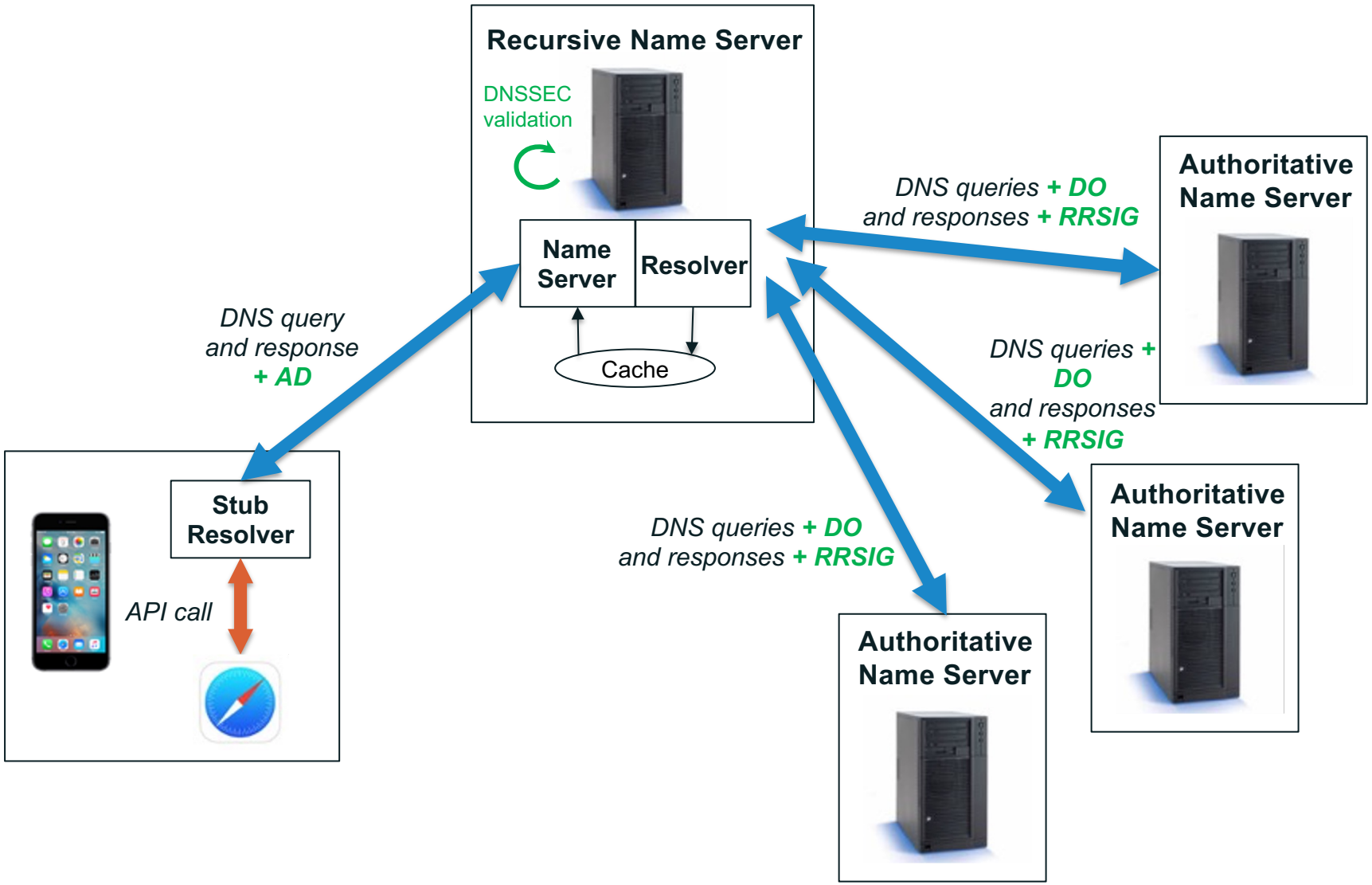
What DNSSEC Doesn't Do

- Provide any confidentiality for DNS data:
  - No encryption
  - Man in the middle-attack
  - DNS over HTTPS (DoH- RFC 8484) and DNS over TLS (DoT – RFC 7858) – more suited
- Address attacks against DNS software:
  - DDoS
  - BCP38

Protection by DNSSEC



DNS resolution process with DNSSEC



Who can implement DNSSEC ?

- Enterprises – Sign their zones (NS) and validate DNS lookups (resolvers)
- TLD Operators – Sign the TLD
- Domain Name holders – Sign their zones
- Internet Service Providers – validate DNS lookups
- Hosting Provider – offer signing services to customers
- Registrars – accept DNSSEC records (e.g., DS)

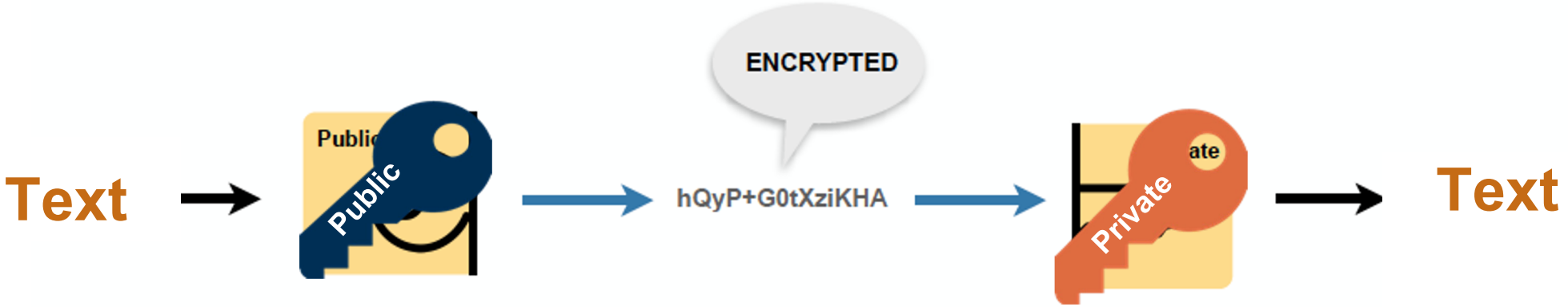
DNSSEC Signing
& a few Cryptography Basics …

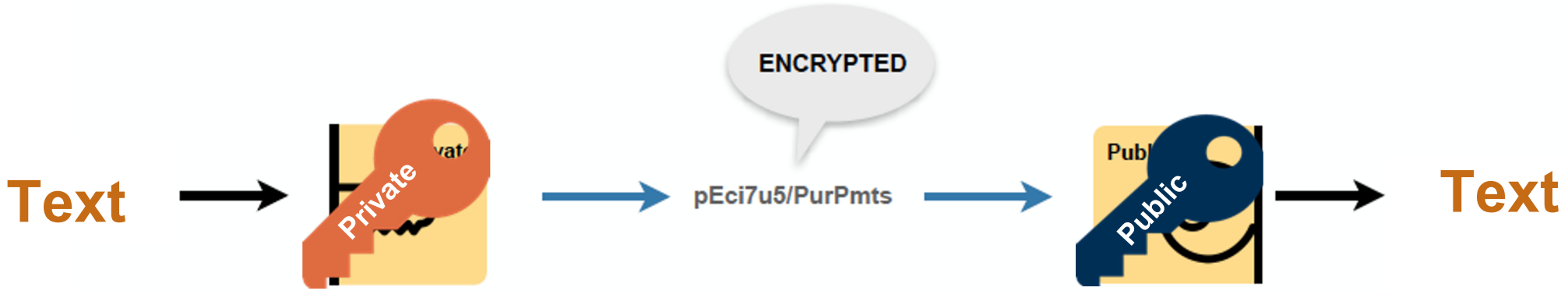
Cryptographic Basics

To provide this, we use

- Asymmetric cryptography
- Digital signatures

Private and Public Keys

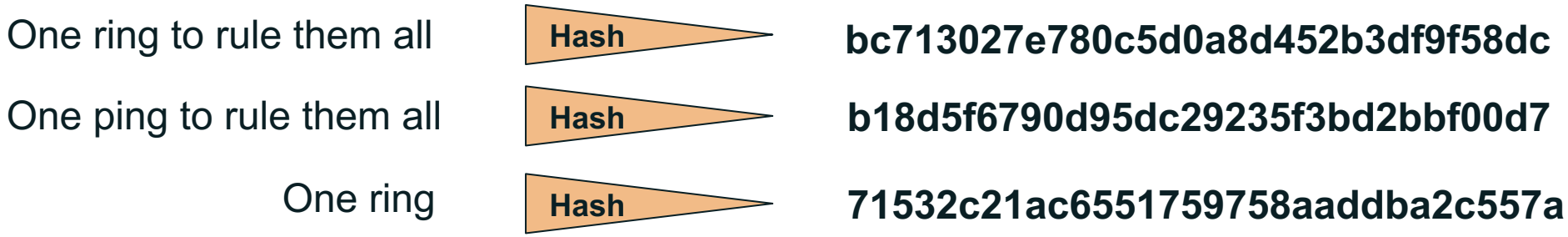




Hash Function

- A cryptographic hash algorithm produces a fixed-size output (fingerprint) called a hash or digest for any size input.

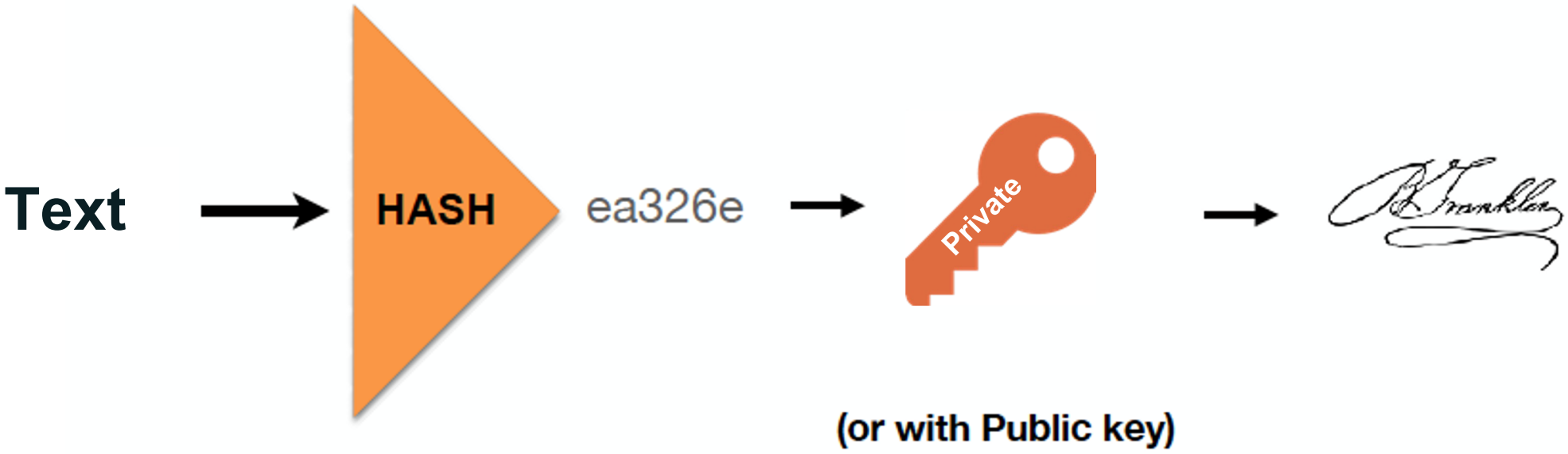
Example of MD5 digests (an MD5 hash is created by taking a string of an any length and encoding it into a 128-bit fingerprint):



Digital Signature

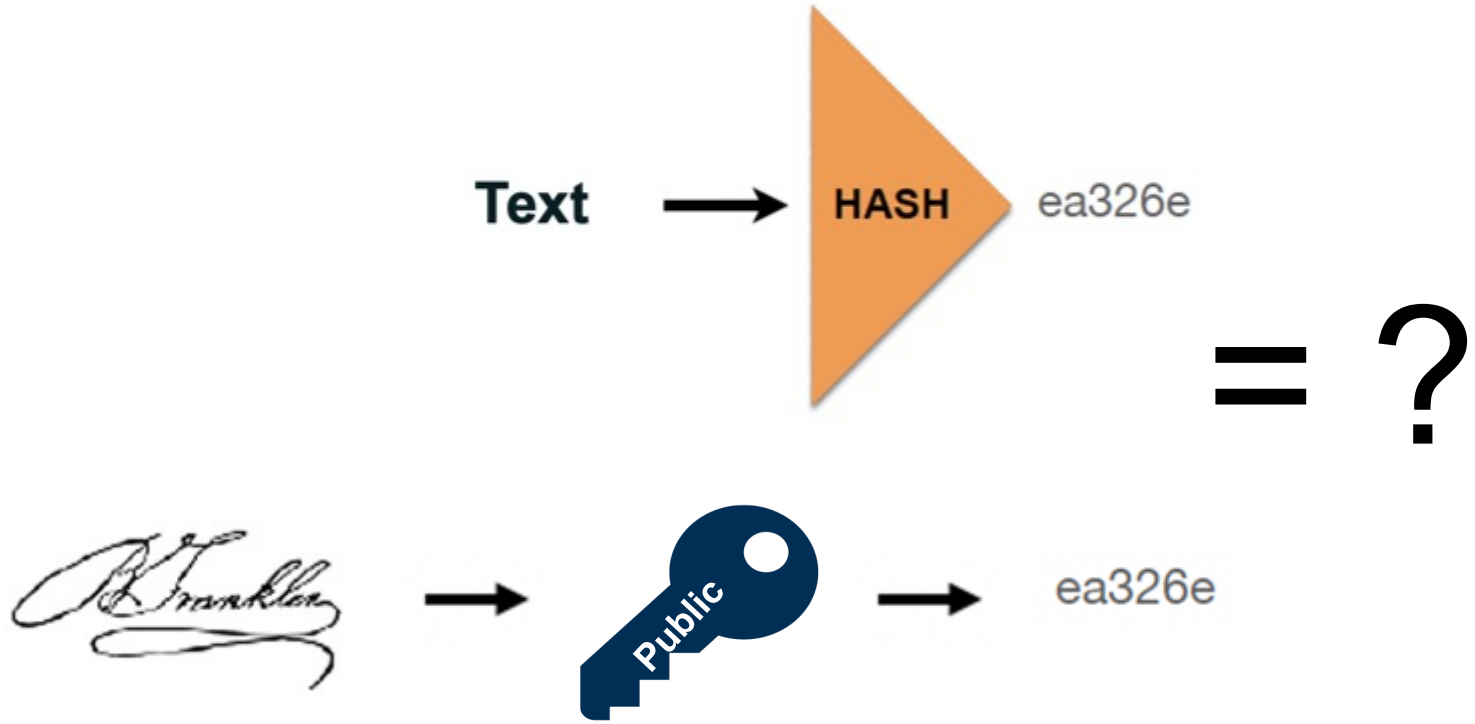
- **We may combine *hash* with *private and public key*, to obtain a digital signature of any text**

Hashing + Encrypt = Digital Signature



Validate the digital signature

Use normal clear text, signature and public keys to do two ways verification and compare both hashes.



So, DNSSEC !

Fasten your seatbelts …

Discussion: what are the RRs for DNSSEC ?

New Resource Records

RRSIG	Signed Resource Records
DNSKEY	Public Key
DS	Delegation Signer (Chain of Trust pointer)
NSEC	Next Secure (Proof of Non-existence)

Signing DNS Data

- In DNSSEC, each zone has a public/private key pair
- Data in the zone is signed with the private key
  - Signing the data is usually de-coupled from serving the data
  - The design allows data to be signed ahead of time rather than “on the fly” for each response
- Important: In DNSSEC, DNS *data* is signed, not DNS *messages*
  - Signing messages is called transaction security
  - A separate protocol called TSIG handles that

Two Keys: ZSK and KSK

- 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
- Zone Signing Key (ZSK)
  - Signed by the KSK
  - Used to sign the zone data RRsets
  - Flags: 256
- This decoupling allows for independent updating of the ZSK without having to update the KSK, and involve the parents (i.e. less administrative interaction)

Zone Key Pairs

- The zone's public key is published in the zone in a specific record (DNSKEY)
- 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.

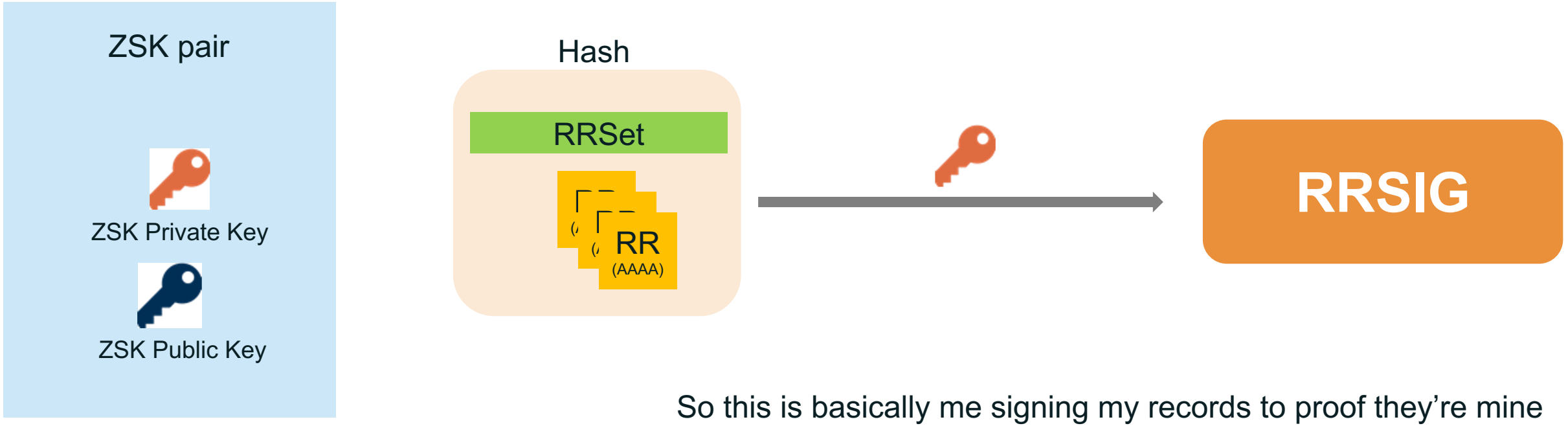
  Stored in a hardware security module (HSM)
  - Offers the most protection but overkill (may also be costly) for many applications.

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 proof they're mine

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

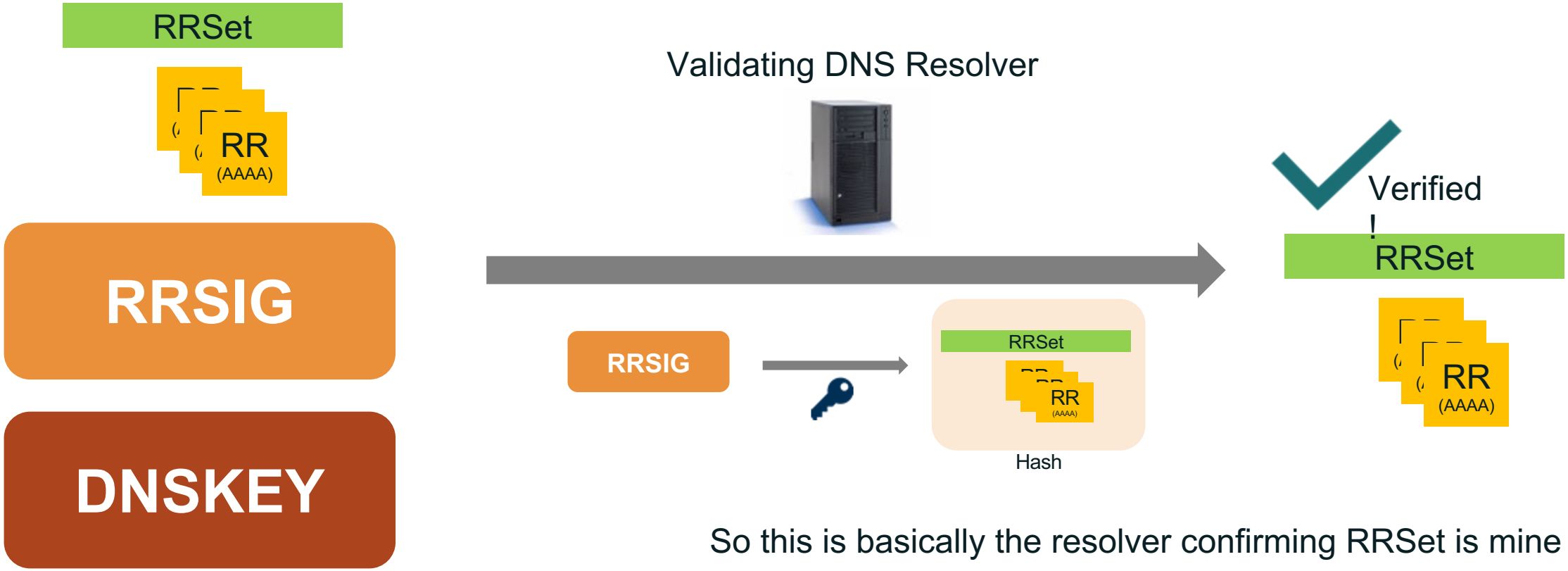
DNSKEY

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

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).



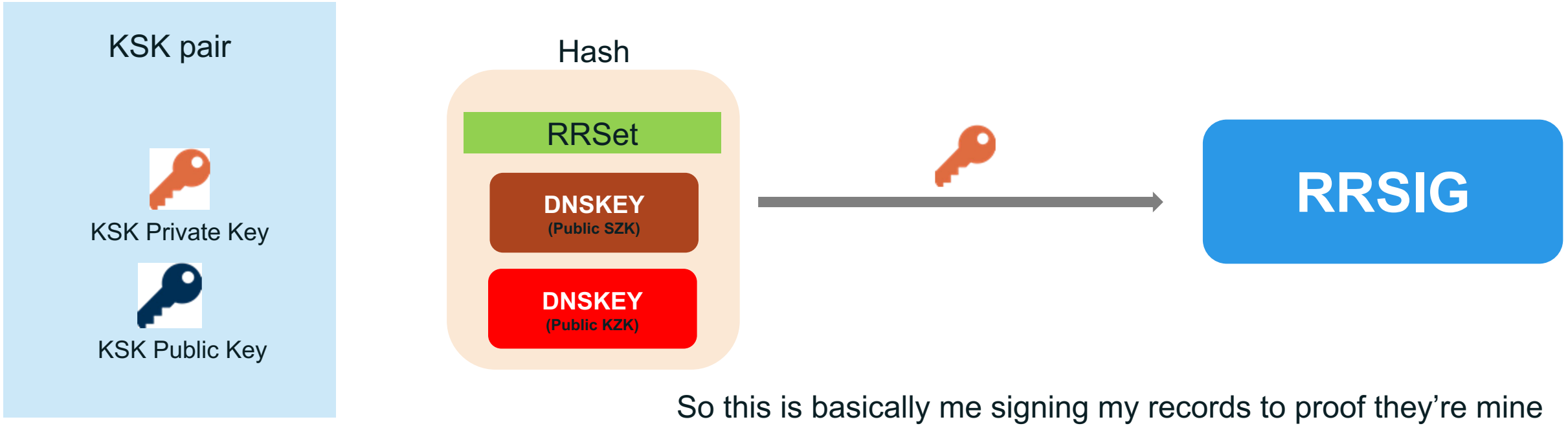
So this is basically the resolver confirming RRSet is mine

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

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.

KSK pair

KSK Private Key

KSK Public Key

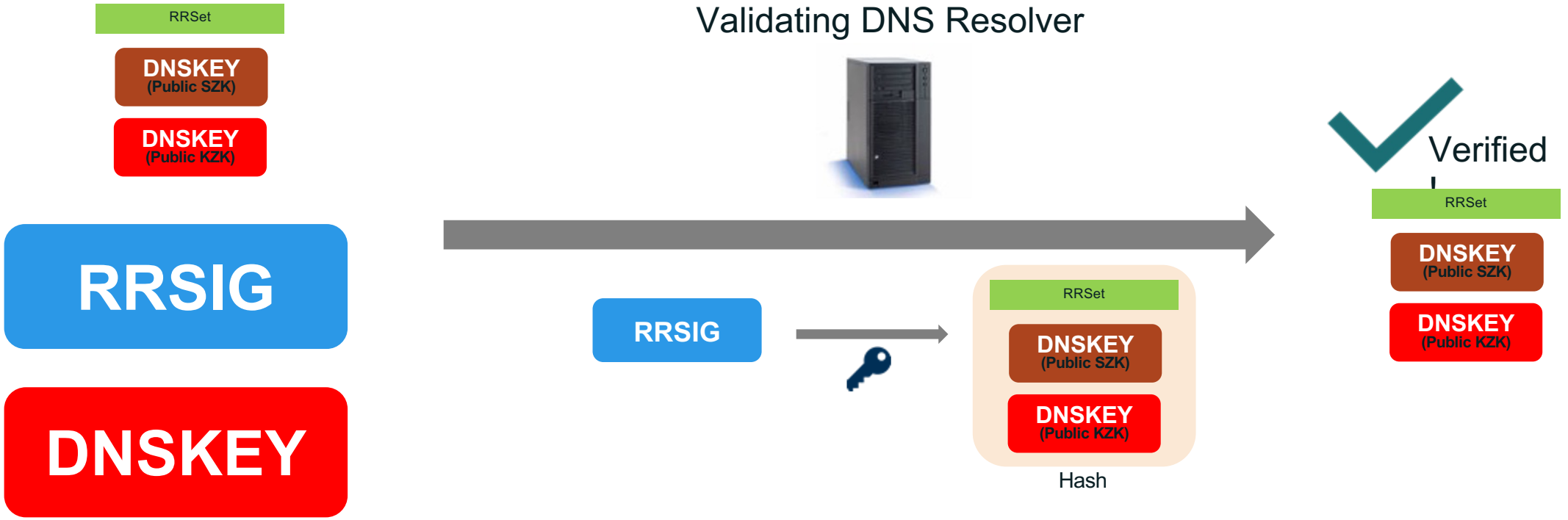
DNSKEY

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

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

DS

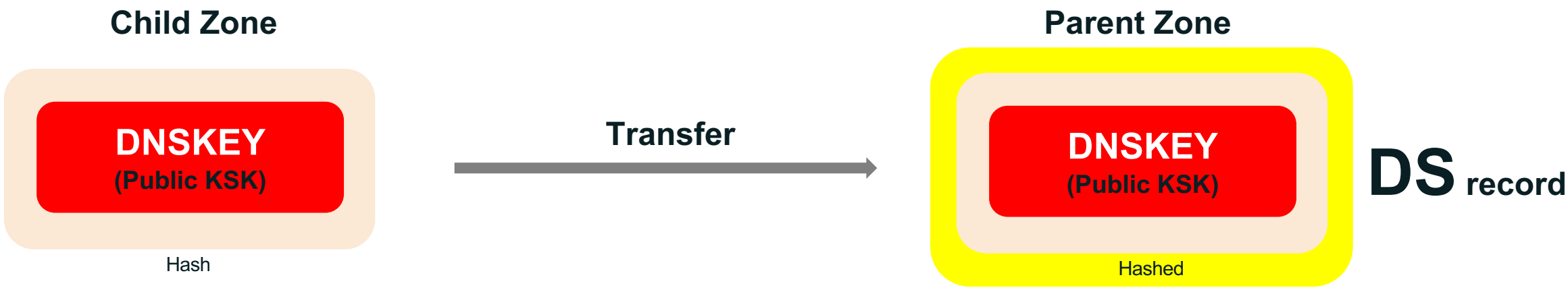
So far, we have established trust within our zone.

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

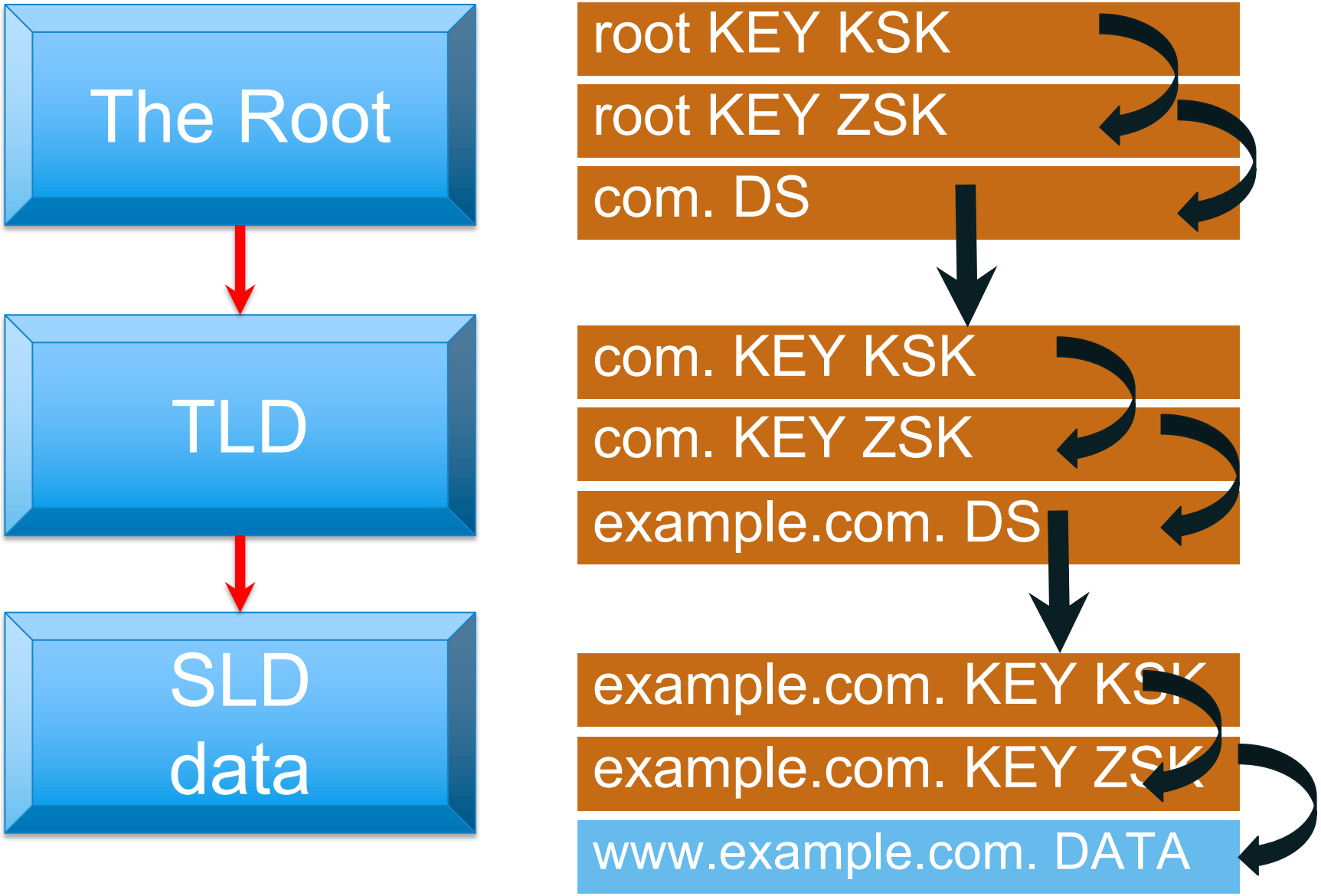
Changing ZSK is easier than changing KSK; also this allows for having smaller ZSK (compared with stronger and bigger KSK) and thus reducing amount of data exchanged among servers (in the responses containing the keys and signatures for each RRset).

... 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).



Signing Chain



Chain of Trust

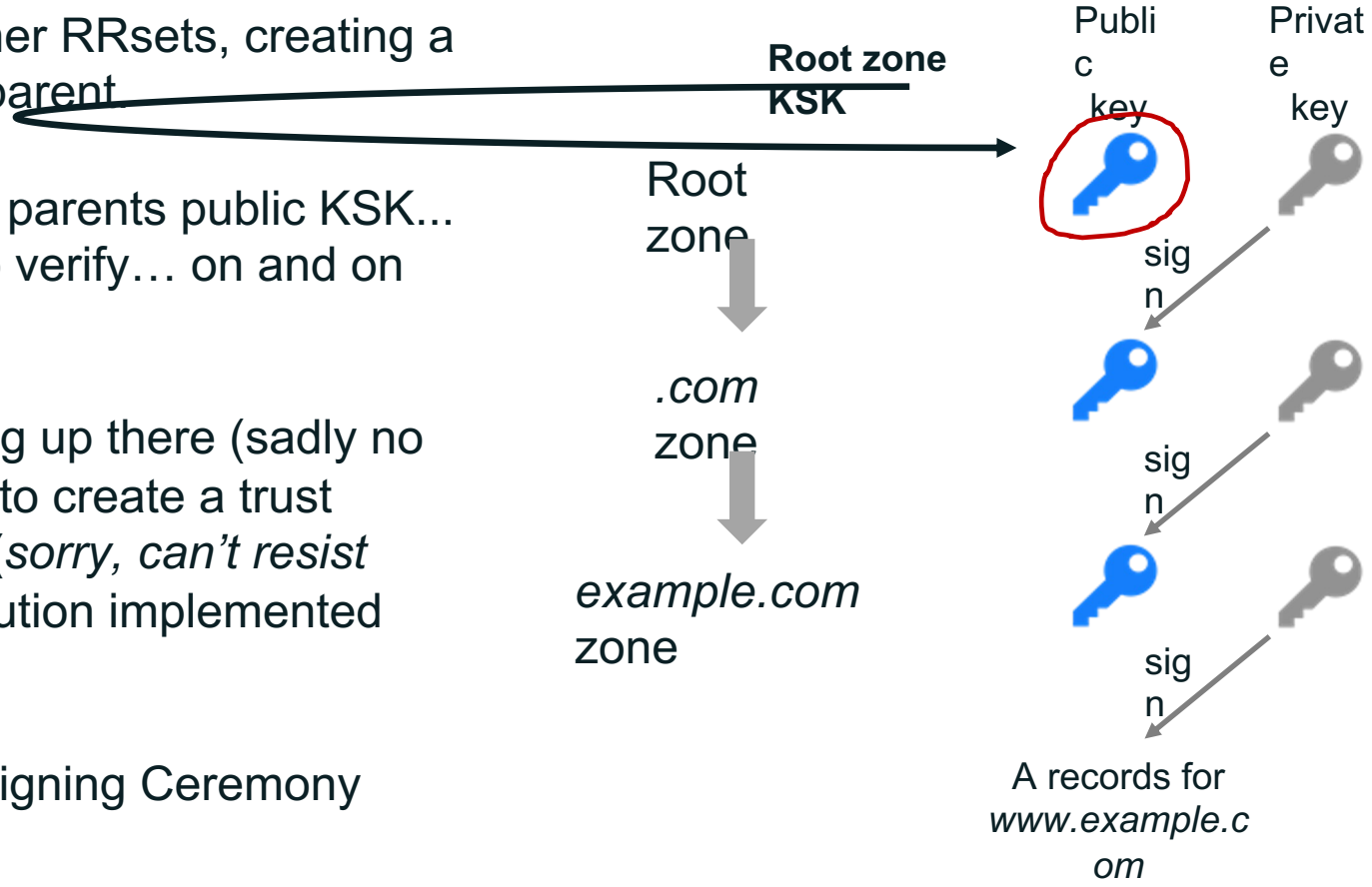
Finally, how do we trust DS record?

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" (*sorry, can't resist quoting LOTR again*)… and here it comes a solution implemented since 2010 called:

The Root Signing Ceremony



… to be continued …

More detail on DNSSEC RRs

The DNSKEY Record

- Fields:

  256 or **257**, the 16-bit flags field
  - Bit 7 is set to indicate a DNSSEC zone key
  - Bit 0 is set to indicate a key-signing key (KSK)

  3, the protocol octet
  - Will always be 3 to signify DNSSEC

  8, the algorithm number (RSA/SHA-256)

  And the public key itself, encoded in base64
  - 2048-bit RSA keys in this example

```
example.com.      600      DNSKEY   256 3 8 (
                  AwEAAdQdbS3W+EoxaGv21gOGGSUFHB6PNVNC
                  PecSLswQ7eKTVtPEYRd+VNDDRZShSOSNFDZq
                  eLcO66EO7N8E8udVxGMpBmk59V1YLGAOTIqW
                  J5132IGA9JgjSabtYtKU4kMbXqNKM8JrtlJd
                  sFF/nixVZzusEl1XZ1u38wozEu0uk39jo5ki
                  cju9o5UL2J+cXo7thBY8VRXibmCiz9FWB0G5
                  YH/YBgWdI8aFnojoPHbaMUr3G7MObahqCxzv
                  41EWPa9AsL97vKil71FD+Jt51Kzq6LcIK55F
                  P3I/oUQXGssJ0tINNnR7IVb8uwfo29w5p0DW
                  JG930HAltYPDav785Z6Gg8M=
                  ) ; ZSK; alg = RSASHA256; key id = 47265

                  600      DNSKEY   257 3 8 (
                  AwEAAbcTEHTvHv7TzxbeVhFSd9pCivORG73p
                  POTsT4WLB7FKtmLwJTXwaKS1bHcY+hm9TL8i
                  /H1LcecDEZjm9614I8fk61KwrH9Z7K0ibFrb
                  sBirNqXgS43IfRXU1ut4W8BHnOnrKtny2Djd
                  KtV46q9nFbzC4WKT/FT0CBGjcc8J5I8SYepO
                  J/R2jwFBHdvwNrVKz3tT0ndO0ceuLJOfWyfL
                  O/X2GQ6RwWHojjl9V0zpgHockIPtQ+EgSIqx
                  unD1Rv07Ezkd/5tlPBJIXjADL9IstSsaol8S
                  fgrtyLggM83sWzPTvnvqGrgQPmqKrnDsLugo
                  UPAYIYmgZ7TF2al5BbtZ4T0=
                  ) ; KSK; alg = RSASHA256; key id = 21700
```

The RRSIG Record

- Fields:
  - **A**, the type of records signed
  - **8**, the algorithm number (RSA/SHA-256)
  - **3**, the number of labels in the signed name
  - **600**, the original time to live on the records signed
  - **20200517225528**, when the signature expires
  - **20200417225528**, when the records were signed
  - **47265**, the key tag/key ID number
  - **example.com**, the signer's name (the zone name)
  - And the digital signature itself, in base64

```
www.example.com.    600  A       192.0.2.1
                    600  A       192.0.2.2
                    600  RRSIG   A 8 3 600 (
20200517225528 20200417225528 47265 example.com.
0Od1A6bCmBICLCqQqTpKRFeZrm4Lr/NqXOmg
KuM22cjllVLxpdgmwLiU7pTDo2FmOvaNPkgz
a2jhTgSOs6Yj6N0XnkV1e0u2n157YMg26xGv
GqJuPgLKql4KxMjngtdwNB5INQasohALjgAo
uTbu9mQQLdyLrkV54P5MUE7lOTFaliWEqW1e
Z/vdaYMc2yKb8CmOQwKxsoWlgnQTYO+lkLuZ
GGffjWH96p6mDbyl5UNA4umSDEqbVKs29Ldv
H7XGOEfkmkze4jSyVUMh57m1DV4ZVLuqx8bQ
YH9zTJPSqvizlSNkuVqssFwknCLwwSOb9FhS
Po9ylhJ9iRPdT34frg== )
```

The DS Record

```
; This is an excerpt of the .com zone file
example.com.              NS      ns1.example.com.
                          NS      ns2.example.com.
                          DS      21700 8 1 (
                                  43839D3767944EDD08BA5F342A1F0526FDE1
                                  F2E0 )
                          DS      21700 8 2 (
                                  7C600DA93B9D0A6EAFC8DFA9C757D1CC59CD
                                  6281EFBAD75DA30FC5B1A121EDC4 )
                          RRSIG   DS 8 2 600 (
                                  20180518010942 20180418010942 22089 com.
                                  Lpcx20t+2K3svnR4/KAu7pUtBM90upIeUxF6
                                  k7USsg/usvLY2MXmUSTZo00jOD+5CNPMYiLq
                                  v/KwDjsxCfjZd25nWy0HLaNCF4kq/Hx7IkA3
                                  XxF7c/pjYHSIgGKQ5JdD1x+ns9XNeSxIy7Ic
                                  94Gp61SRFd87Mp6KNCbED3BGzmxMTHn4Yql2
                                  +TEfvmSHa4shxjtbZOtIFSNnzDKPTwcmtjHK
                                  m5WccKUXFrdEgUg03TsqJBDWnlzga7NdNITA
                                  tWgUKxALyycNGjla4shk6t4mTEpzFe631k2Q
                                  0vJamA+MfLZSz6ojT3SU7LyJrMO+RgaslqeE
                                  i4UWCs6+JOnLAnFKXQ== )
```

- DS record's fields:
  - **21700**, the key tag/key ID number (of the *example.com* KSK)
  - **8**, the algorithm number (RSA/SHA-256)
  - The DS digest type: **1** is SHA-1, **2** is SHA-256
  - And the digest, in hexidecimal

Proving Something Doesn't Exist

- Two kinds of "negative errors" in DNS when the queried RRset doesn't exist:
  - Name Error (NXDOMAIN)
  - "No such data" (NOERROR/0)
- How do you prove cryptographically that an RRset doesn't exist?
- Could sign negative responses "on the fly"
  - But the design of DNSSEC doesn't require the private key to be available when serving the zone
- Or sign something ahead of time: the **NSEC** record

The NSEC Record

- The NSEC record spans a gap between two domain names in a zone
- The NSEC record…
  - Resides at a given domain name
  - Specifies what types exist at the name
  - Points to the next name in the zone
- Notion of a “next” name implies a canonical order, which is introduced by DNSSEC
- Domain names in a zone are sorted by:
  - Shifting all characters to lowercase
  - Sorting non-existing bytes ahead of “0”
  - Sorting lexicographically from the highest-level label to the lowest-level

A Zone With NSEC Records Added

```
example.com.                    SOA             ns.example.com.
hostmaster.example.com.
                                2018041700 3600 600 86400 600
example.com.                    NS              ns.example.com.
example.com.                    A               10.0.0.1
example.com.                    MX              0 mail.example.com.
example.com.                    NSEC    east.example.com. A NS SOA MX NSEC
east.example.com.               NS              ns.east.example.com.
east.example.com.               NSEC    ns.east.example.com. NS NSEC
ns.east.example.com.    A               10.0.0.5
ns.east.example.com.    NSEC    ftp.example.com. A NSEC
ftp.example.com.                CNAME   www.example.com.
ftp.example.com.                NSEC    mail.example.com. CNAME NSEC
mail.example.com.               A               10.0.0.2
mail.example.com.               NSEC    ns.example.com. A NSEC
ns.example.com.                 A               10.0.0.1
ns.example.com.                 NSEC    www.example.com. A NSEC
west.example.com.               NS              ns.west.example.com.
west.example.com.               NSEC    ns.west.example.com. NS NSEC
ns.west.example.com.    A               10.0.0.4
ns.west.example.com.    NSEC    A NSEC
www.example.com.                A               10.0.0.3
www.example.com.                NSEC    example.com. A NSEC
```

NSEC 3

- **NSEC3** is an alternative to NSEC providing:
  - Non-enumerability
  - Opt-out
- Why the name NSEC3?
  - The name reflects the number of people who actually understand it
  - That was a joke
  - But NSEC3 is indeed very complicated

Non-Enumerability

- Stops zone enumeration via “zone walking” the NSEC chain
- Instead NSEC3 chain is a hash of names
- Example:

  Zone: *alpha.example*, *bravo.example*, *charlie.example*

  NSEC chain:

  - *alpha.example* → *bravo.example* → *charlie.example*

  NSEC3 chain:

  - HASH(*bravo*).*example* → HASH(*alpha*).*example* → HASH(*charlie*).*example*
  - *ACJENFKS.example* → *DGJRPFKDM.example* → *QVNRJVMD.example*

  (Note: hash names are examples only and shorter than an actual NSEC3 hash)

Opt-Out

- Standard DNSSEC:
  - Every name in a zone has an NSEC
    - Including delegations (NS RRsets)
- Opt-Out DNSSEC:
  - Only secure delegations have an NSEC
    - Delegations to zones that are signed
    - i.e., delegations that also have a DS RRset
- Much better for large zones like *.com*
  - Many names, but few secure delegations
  - Much shorter NSEC3 chain than if there were an NSEC chain
  - Fewer signatures
  - Smaller signed zone

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
```

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>
```

Time for practice: play with DNSSEC resource records

- CLI : dig
- Web : https://www.digwebinterface.com/

DNSSEC Validation

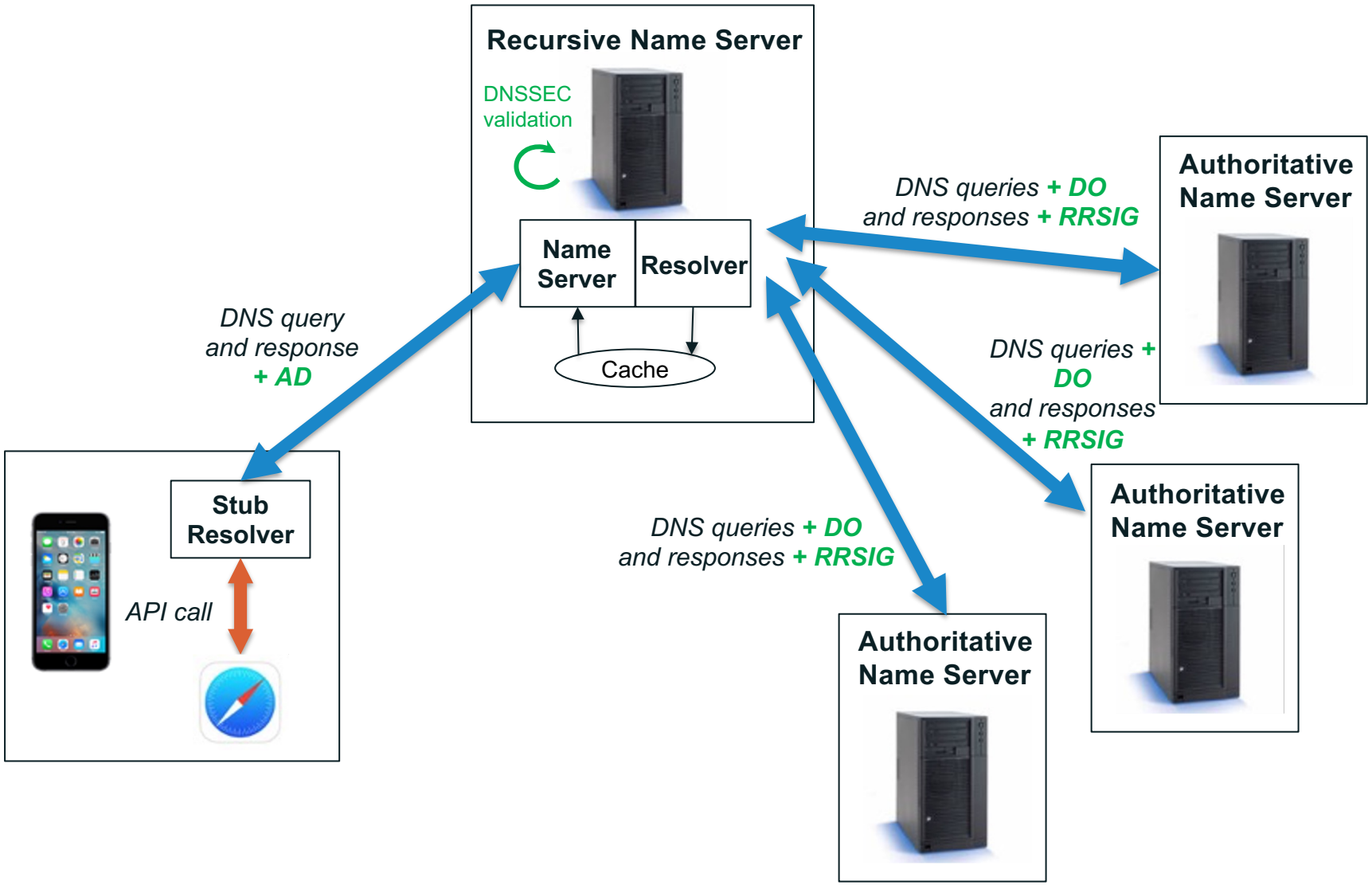
DNSSEC enabled - resolvers in action

DNSSEC Validation

- DNSSEC validation is the process of **checking the signatures** on DNSSEC data that help to verify authenticity and integrity of signed zones.
- The DNSSEC signatures data come alongside with the DNS response for domains that are signed.
- DNSSEC validation protects your customers/users from being redirected to a wrong/fake destination (web site, online service, …)
- Validation can occur in applications, stub resolvers or recursive resolvers. Most validation today occurs in recursive resolvers.
- Trust Anchor: To perform DNSSEC validation, you have to trust somebody (some zone's key). **Root Zone KSK is the most important trust Anchor on the Internet. You can view root key signing ceremony on YouTube.**
- What happens when validation fails?

  The recursive resolver protects the user by sending a "SERVFAIL" error response.

DNS resolution process with DNSSEC



Chain of Trust

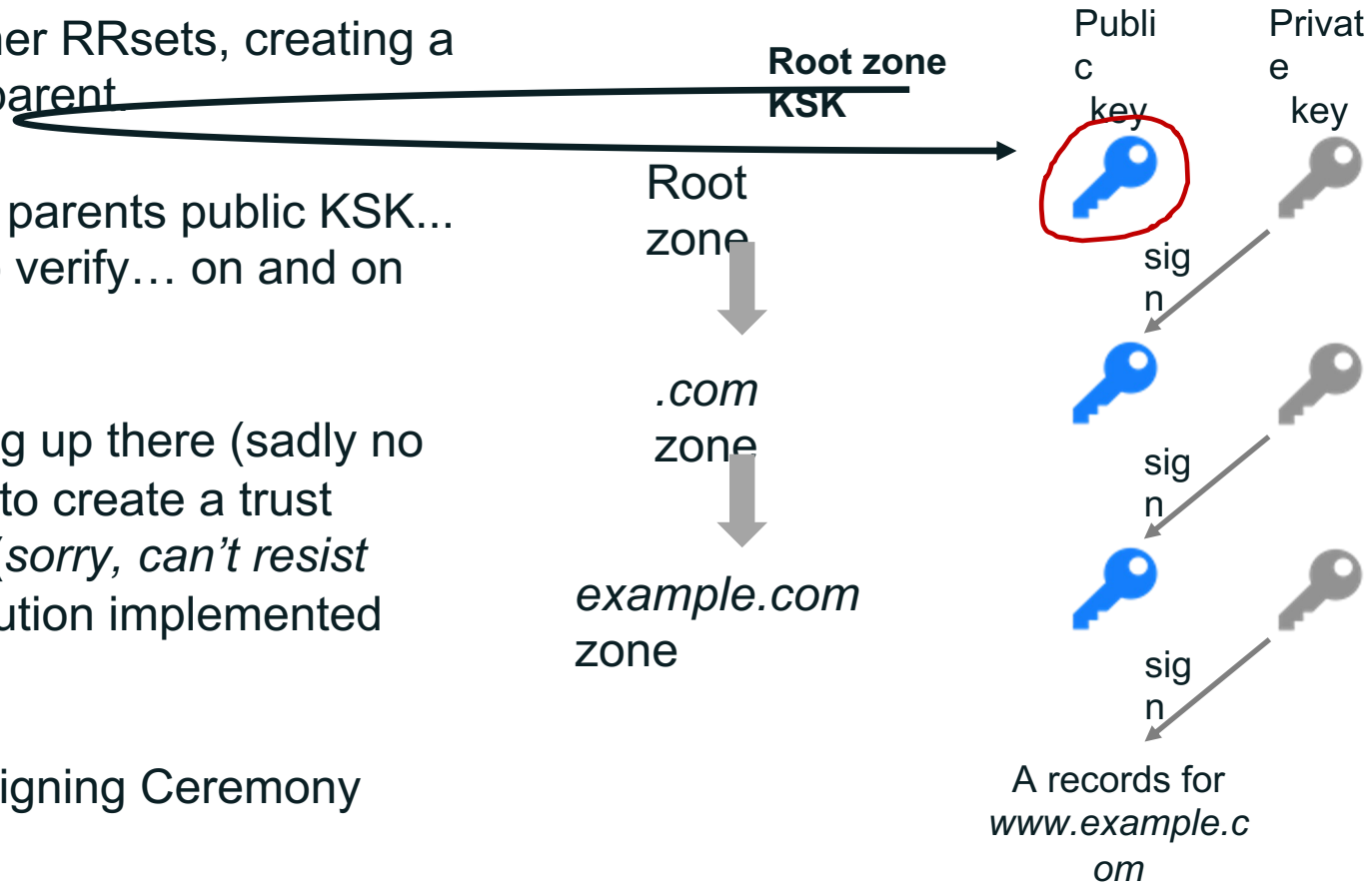
Finally, how do we trust DS record?

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" (*sorry, can't resist quoting LOTR again*)… and here it comes a solution implemented since 2010 called:

The Root Signing Ceremony

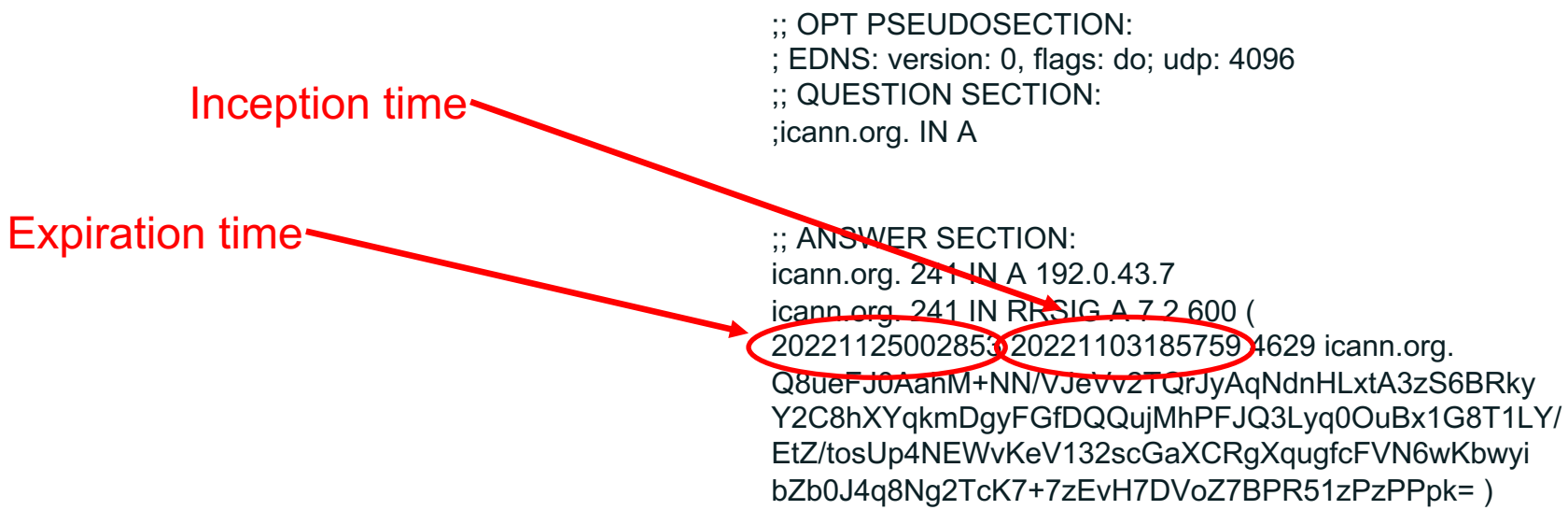


Some hardware & network considerations

- **System memory**: DNSSEC generates larger response sets and therefore takes up more memory space. It is good practice to have more detailed monitoring of memory occupation
- **CPU**: DNS response validations usually lead to increased CPU usage (without an immediate need to change the machine).
- **Network interfaces**: Although DNSSEC increases the total amount of DNS traffic, it is unlikely that you will need to update the network interfaces on the name server. It is also a good practice to do a detailed monitoring of the traffic increase due to the activation of the validation.
- **Large UDP packets**: Some network equipment, such as firewalls, can make assumptions about the size of UDP DNS packets and incorrectly reject DNS traffic that appears "too large". You should check EDNS configuration.
- **Check network connectivity over TCP port 53:** this may mean updating firewall or ACL policies on routers.

Some hardware & network considerations (2)

- **Make sure the Network Time Protocol (NTP) is working properly**: DNSSEC make use of real clock time and date when validating the signed RRs (RRSIG) so if the resolver date/time is wrong, it won't verify the signatures correctly and may lead to attack vectors exploiting a non sync resolver. Good practice is to have the resolver use a trusted NTP service and make sure it always has network connectivity to it and is sync.



Enabling DNSSEC Validation

What do you need to enable DNSSEC validation ?

- If you run your own DNS recursive resolvers (open source or commercial) within your network, activate DNSSEC validation is usually simple and does not require a new investment. Most of the softwares already have it embedded, you just need to perform some verification before activating in the configuration. Those verifications are :

  hardware resources utilization (memory, CPU) and network bandwidth utilization.

  server clock synchronization: NTP

  current trust anchors in the system

  EDNS

  TCP port 53 should be open

  Explicitly exclude forward-zones (if you have any)

- If you are using external recursive resolvers, make sure that they are DNSSEC validating. If not, you can refer to their administrators and suggest them to activate it.

Enable DNSSEC Validation in BIND 9.11+

On /etc/bind/named.conf.options :

dnssec-validation auto

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation auto;

        listen-on-v6 { any; };
};
~
```

Enable DNSSEC Validation in Unbound 1.7+

1) Download root-key trust anchor:
unbound-anchor

2) On /etc/unbound/unbound.conf.d/root-auto-trust-anchor-file.conf :
Uncomment the line:
 # auto-trust-anchor-file: "/var/lib/unbound/root.key"

To:
auto-trust-anchor-file: "/var/lib/unbound/root.key”

3) Restart Unbound

Enable DNSSEC Validation in Infoblox

- Infoblox DNSSEC deployment Guide (signing and validation): https://www.infoblox.com/wp-content/uploads/infoblox-deployment-guide-dnssec.pdf

DNSSEC validation

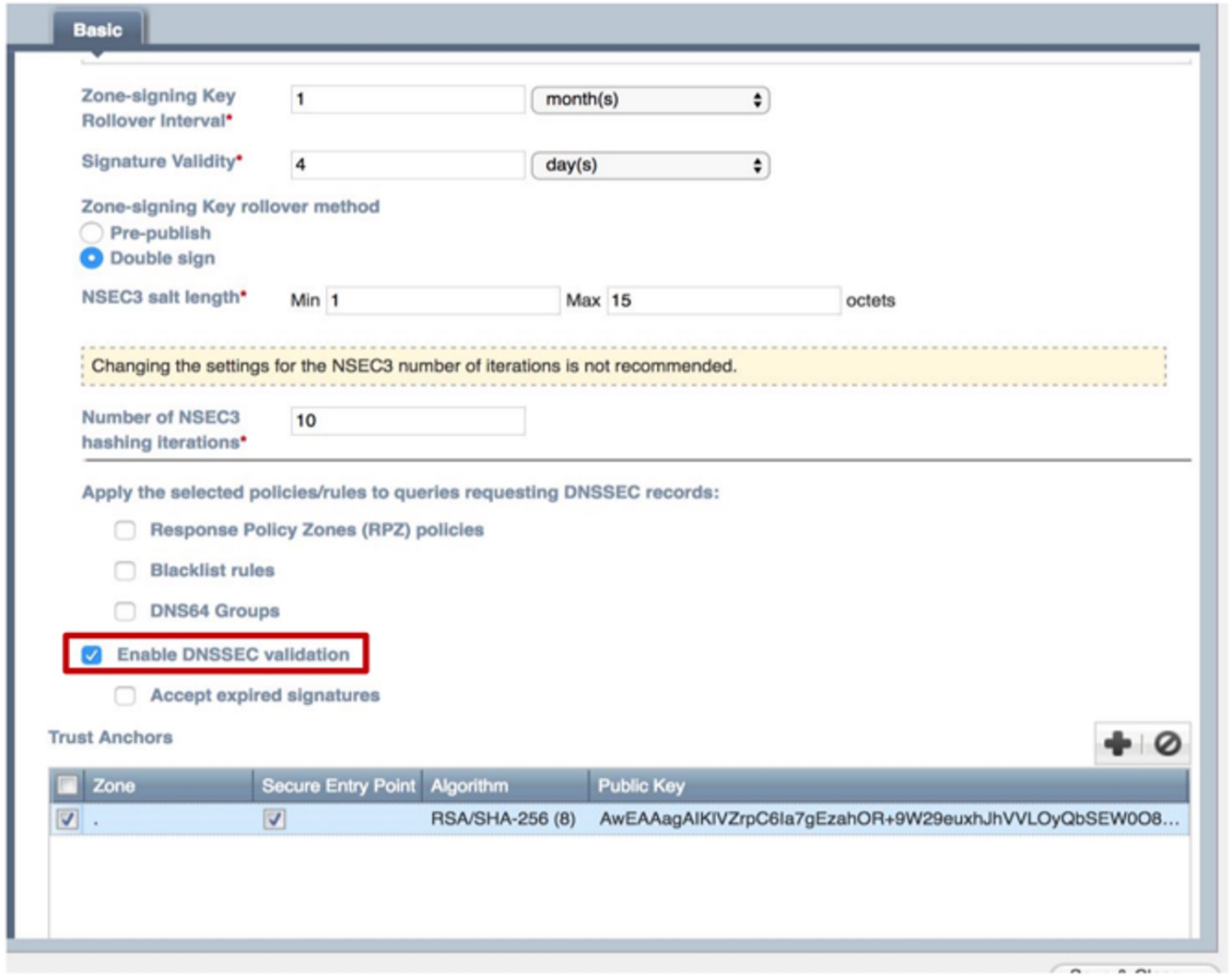
Prerequisites

1. EDNS0 must be enabled and supported by your networking equipment.
    a. Check the section Troubleshooting for a quick method on how to test if your environment supports EDNS0.
2. Recursion must be enabled.

Steps to enable DNSSEC Validation

1. Go to **Data Management > DNS > Grid properties**
2. Toggle advanced on (if not already enabled)
3. Click on DNSSEC
4. Check the Enable DNSSEC box
5. Scroll down and check the Enable DNSSEC validation checkbox
6. Once you have enabled the feature, you will need to obtain the root key(s) in a secure way and enter it/them under Trust Anchors

Enable DNSSEC Validation in Infoblox



Enable DNSSEC Validation in Infoblox

- Ascertain Root Key (Trust Anchor):

  AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjFFVQUTf6v58fLjwBd0YI0
  EzrAcQqBGC
  zh/RStIoO8g0NfnfL2MTJRkxoXbfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efuc
  p2gaDX6RS6
  CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpzW5hOA2hzCTMjJPJ8LbqF6dsV6
  DoBQzgul0s
  GIcGOYl7OyQdXfZ57relSQageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhYB4N7knNnu
  lqQxA+Uk1ihz 0=

- Add this key under Trust Anchors for “.” and set the algorithm to 8

Test your Resolver is Validating

- Do you get the **ad** bit?

```
root@resolv2:~# dig @127.0.0.1 icann.org +dnssec +multiline

; <<>> DiG 9.16.1-Ubuntu <<>> @127.0.0.1 icann.org +dnssec +multiline
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3195
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;icann.org.                     IN A

;; ANSWER SECTION:
icann.org.              600 IN A 192.0.43.7
icann.org.              600 IN RRSIG A 7 2 600 (
                                20210515183326 20210424162304 54555 icann.org.
                                uUSoNscydwnlVsuT/hk3Fi/aZ3ubozLV/AQQis+lWuor
                                0zMTNXQvk8Vgz0jdYdgBhbFSXa0igdYzewYnkMO6PM2B
                                pIF34IoJ/0ePojRpSqaFL+w6mlIQ7iDKOBwyFBAQ0RQ7
                                FJTJtWKp/WsOnneNMkp81gQviouuTE9EK94Ntps= )

;; Query time: 167 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May 04 10:03:11 UTC 2021
;; MSG SIZE  rcvd: 223
```

State of DNSSEC Deployment

State of DNSSEC validation in your country

https://stats.labs.apnic.net/dnssec

State of DNSSEC Validation

- Most validation today occurs in recursive resolvers
- **Bad News:**
  30.22% of DNS responses are validated according to APNIC Labs*
  Too many resolvers do not validate DNS answers
  . . And not enough domains are signed
- ICANN has a mandate in our strategic plan for 2021-2025 to significantly increase DNSSEC adoption, including convincing DNS resolver vendors to ship their software with DNSSEC validation turned-on by default

https://stats.labs.apnic.net/dnssec/

Time for practice !

- Configure your resolvers to be validating resolvers: lab DNSSEC validation
- Test your validating resolvers with signed and non signed domains.
- Discuss the differences you notice. Tip: check the flags; use the +dnssec option in dig commands.

OCTO-029: a guidebook for DNSSEC deployment

- The guidebook aims to assist ccTLD registry operators in understanding DNSSEC deployment at a TLD.
- Target audience:

  TLD registry managers, staff, registrars, registrants

  Operator who administers zones

  Anyone willing to have an overview of how to deploy DNSSEC on a ccTLD.

  Either you are already DNSSEC signed or not, you can glean insights on current DNSSEC operational best practices from this guidebook.

- DNSSEC deployment checklist: a list of adjustable action items that aims to simplify your journey into DNSSEC deployment.
- Download the guidedebook at : https://www.icann.org/en/system/files/files/octo-029-12nov21-en.pdf

DNSSEC deployment high overview

1. Working DNS infrastructure (with best operational practices).
2. Signing architecture: in-line signing, bump in the wire, etc.
3. Signing mode: manual, semi-automated, automated, etc.
4. Choose parameters: lifetimes (keys, signatures, etc.), algorithms, key sizes, etc.
5. Generate keys
6. Sign and Test
7. Write procedures: normal key rollovers, emergency, keys management, DPS, etc.
8. Test and document
9. Share DS with parent
10. Monitor
11. Publish DPS
12. Plan rollover: ZSK, KSK (manual, semi-automated, automated).
13. DNSSEC deployment guidebook: OCTO 029 ?

Time for practice !

1. Sign your zone: lab zone signing and lab send DS to root zone.
2. Once DS is publish in root zone, confirm you get the AD flag for your DS and your zone records using your internal resolvers.
3. Lab ZSK rollover.

Engage with ICANN – Thank You and Questions

One World, One Internet

Visit us at **icann.org**

@icann

facebook.com/icannorg

youtube.com/icannnews

flickr.com/icann

linkedin/company/icann

slideshare/icannpresentations

soundcloud/icann