DNS/DNSSEC Workshop

In conjunction with LKNOG8

13-16 August 2024

ICANN

Introductions – Trainers

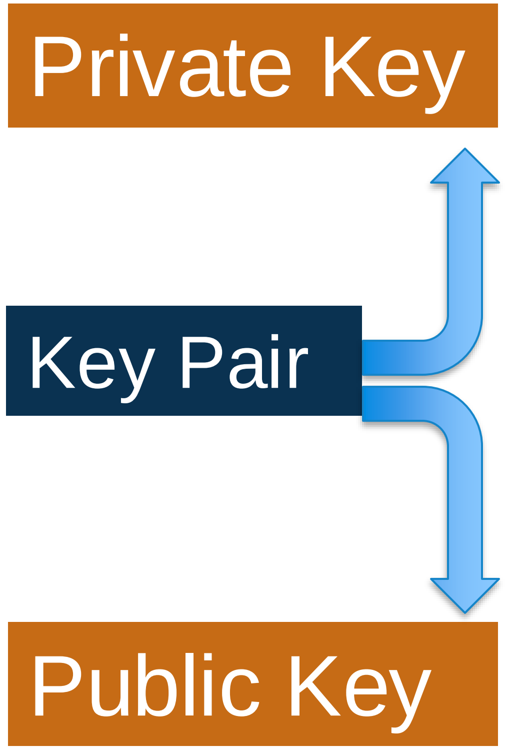
Champika Wijayatunga - *ICANN*
Sampath Hennayake – *LK Domain Registry*
Chamara Disanayake - *NSBM*
Pasan Ravinatha – *University of Moratuwa*

DNSSEC

ICANN

Digital Signatures in Theory

Caution: Cryptography



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.

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.).

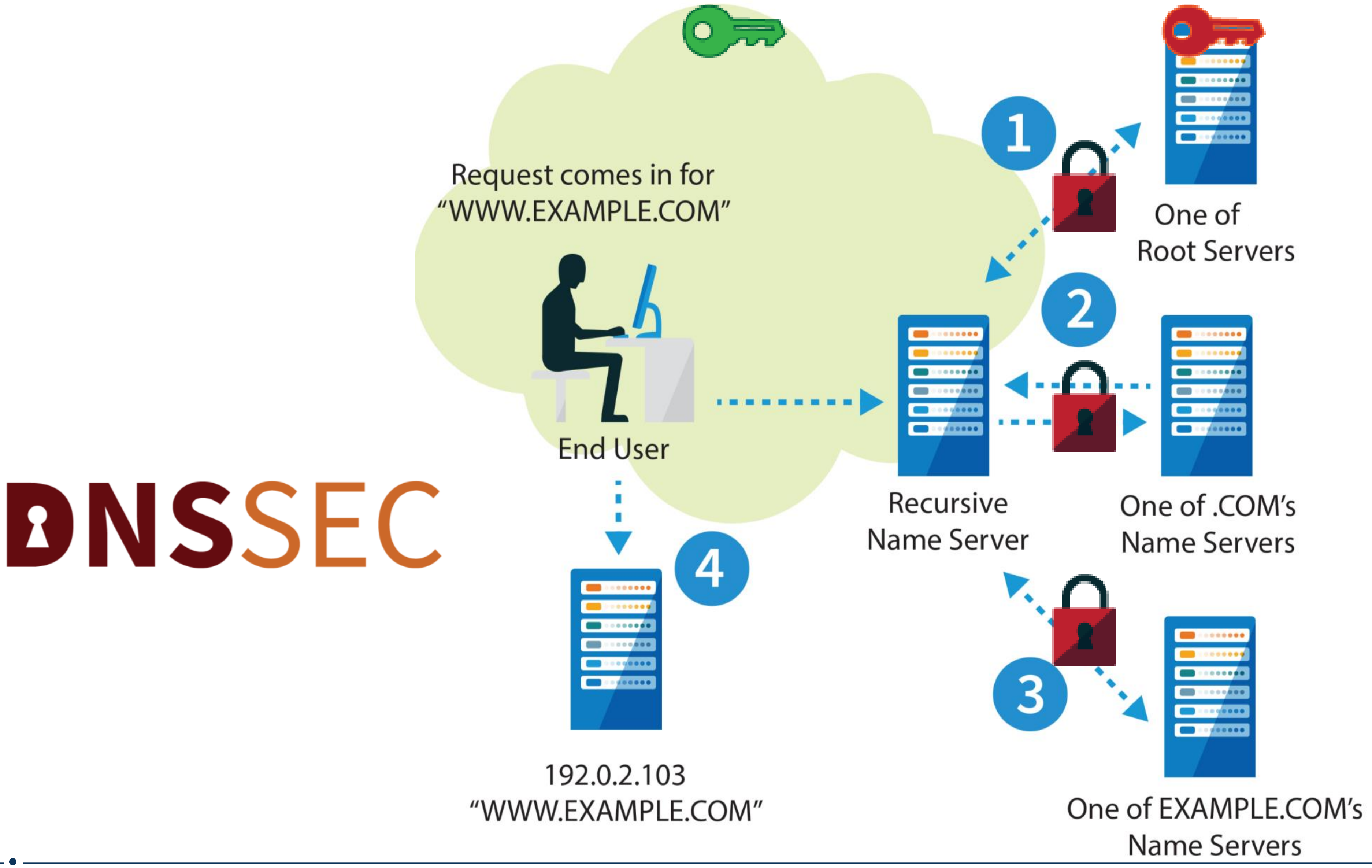
What DNSSEC Doesn't Do

- **DNSSEC does not**:
  - Provide any confidentiality for DNS data
    - No encryption.
    - Transferred data will be readable for person-in-the-middle.
  - Address attacks against DNS software
    - DDoS
    - “packets of death”
    - Etc.

Domain Name System Security Extensions (DNSSEC) - Benefits

- Adds security to the DNS by incorporating public key cryptography.
- Provides assurance to users that the DNS data they get is **valid and true**.
- Helps prevent DNS threats and abuses (cache poisoning, redirection to fake destination, etc.) by verifying and confirming authenticity and integrity of DNS data.
- Protects your digital integrity and your business, protects your customers online.
- Complementary to other technologies like SSL widely used to secure web communications.

DNS Security Extensions



How does DNSSEC work ?

Two actions are required :

- Registrants (domain name holder) should **sign their domain**: the domain administrator generates and maintains the cryptographic keys and signatures for the domain.
- DNS operators, ISPs, mobile operators, hosting providers, IT services,etc. should **activate DNSSEC validation** (verifies the authenticity and integrity of DNS responses from signed domains) in their recursive resolvers: system administrators should enter the server configuration and turn on the functionality.

Who should implement DNSSEC ?

- Registry operator (TLD): ccTLD and IDN ccTLD Registry Operators.
- Companies and business entities:
  - Sign your domains or get them signed.
  - Activate DNSSEC validation on your recursive resolvers.
- ISPs, Mobile Operators, hosting providers etc:
  - Activate DNSSEC validation on your recursive resolvers.
  - Sign your domains and the ones you host for your customers.
  - Accept DNSSEC records such as DS and push to the registry (registrars).
- Registrants: sign your domains or get them signed.

DNSSEC Signing: Technical High Level Overview

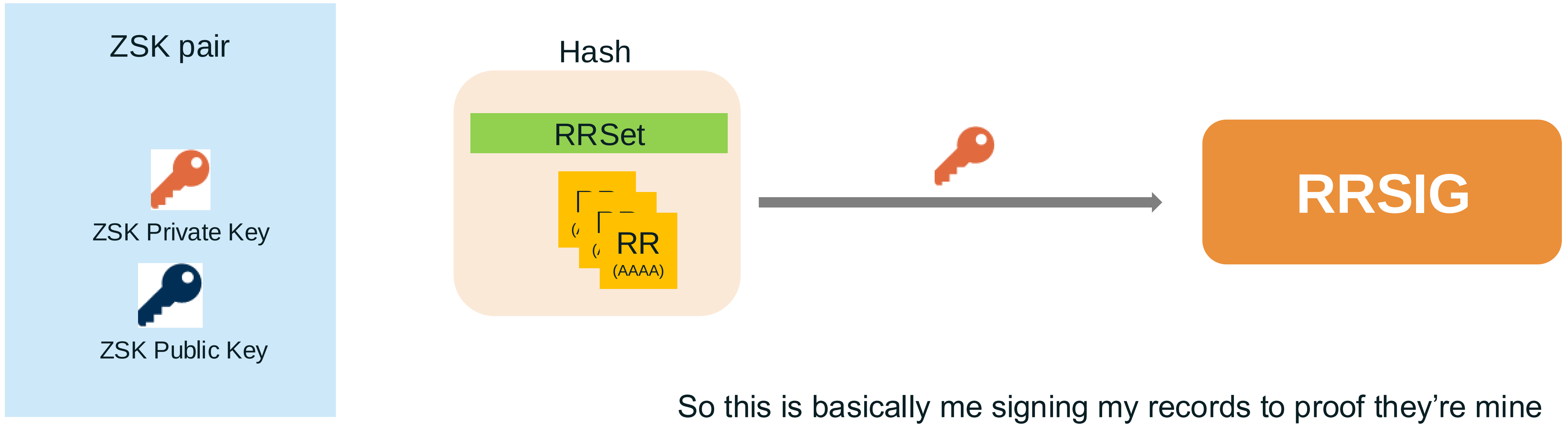
- What/how is the existing DNS infrastructure ?
- Plan and get prepared
- Involve partners: 3rd Party, registrars,
- DNSSEC software solution (OpenDNSSEC, Bind, …), architecture, signing methodology, key generation and management.
- Generate DNSSEC signing keys.
- Test signing and plan for signing in production.
- Sign and when comfortable, upload DS to parent zone: your zone is officially signed.
- Refresh signatures and keys as per best practices and your operational constraints.
- Update Business Continuity Plans
- Monitor, analyze, improve, implement, monitor.

New Resource Records

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

ZSK

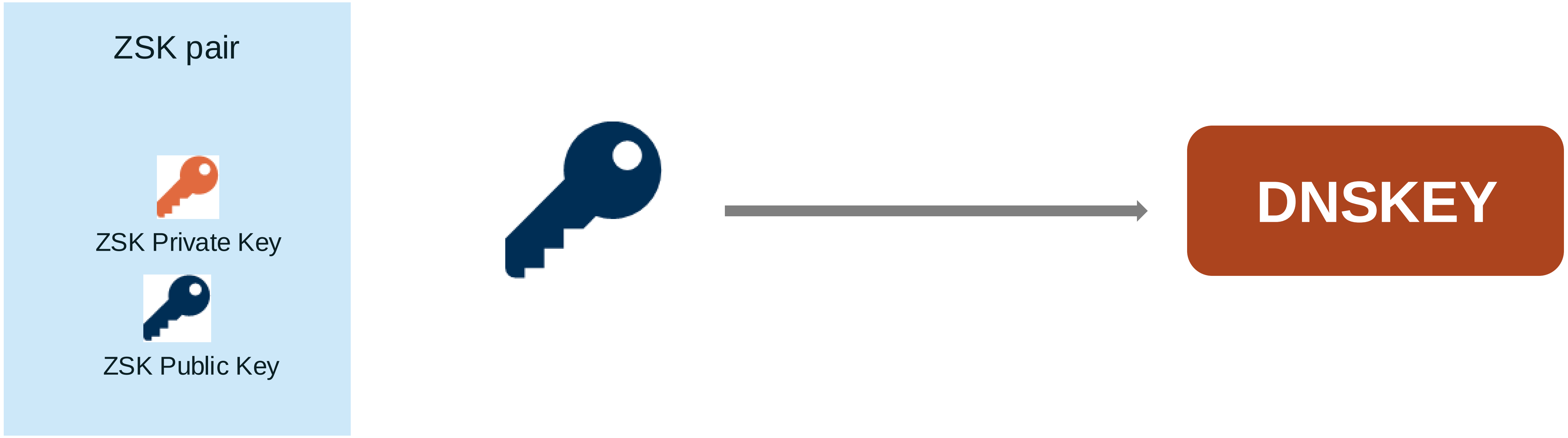
- Recall: An **RRset** is the set of all Resource Records of a given record type for a given name.
- Each zone in DNSSEC has a Zone Signing Key pair (ZSK)
- The zone operator creates digital signatures for each RRset using the private ZSK and then stores them in their name server as RRSIG records.



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.

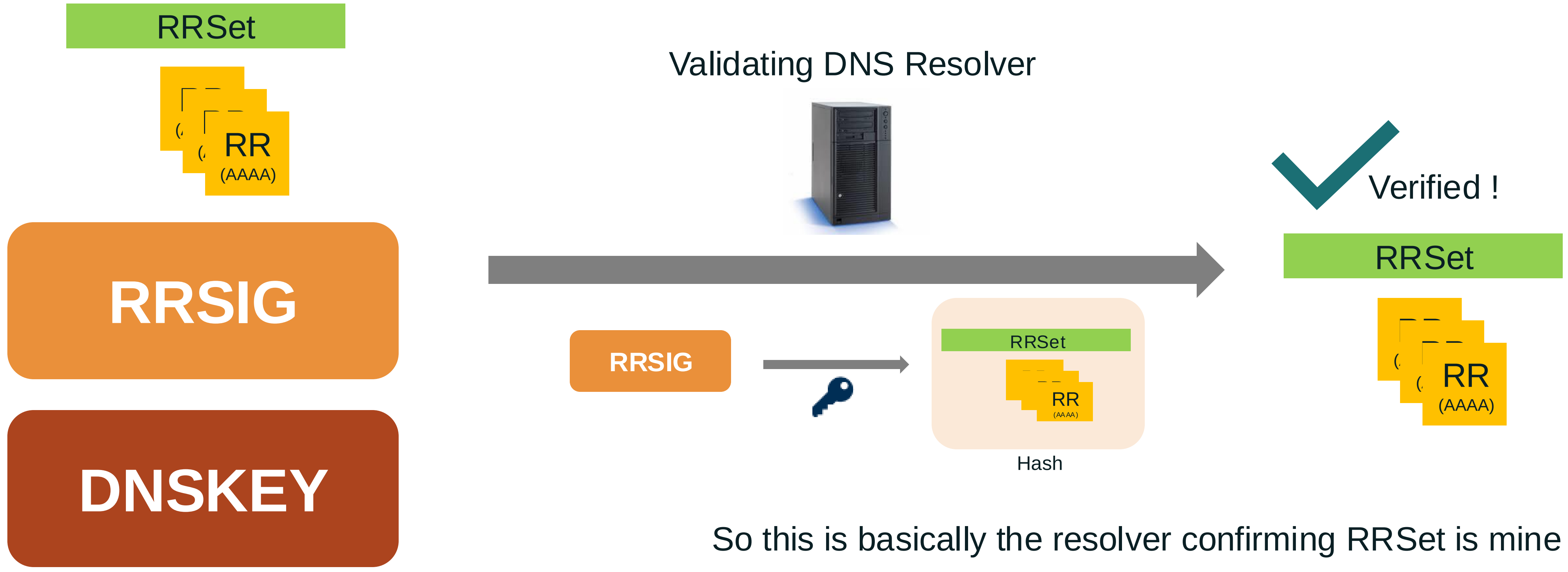


So this is basically me advertising my public key for others to verify

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).



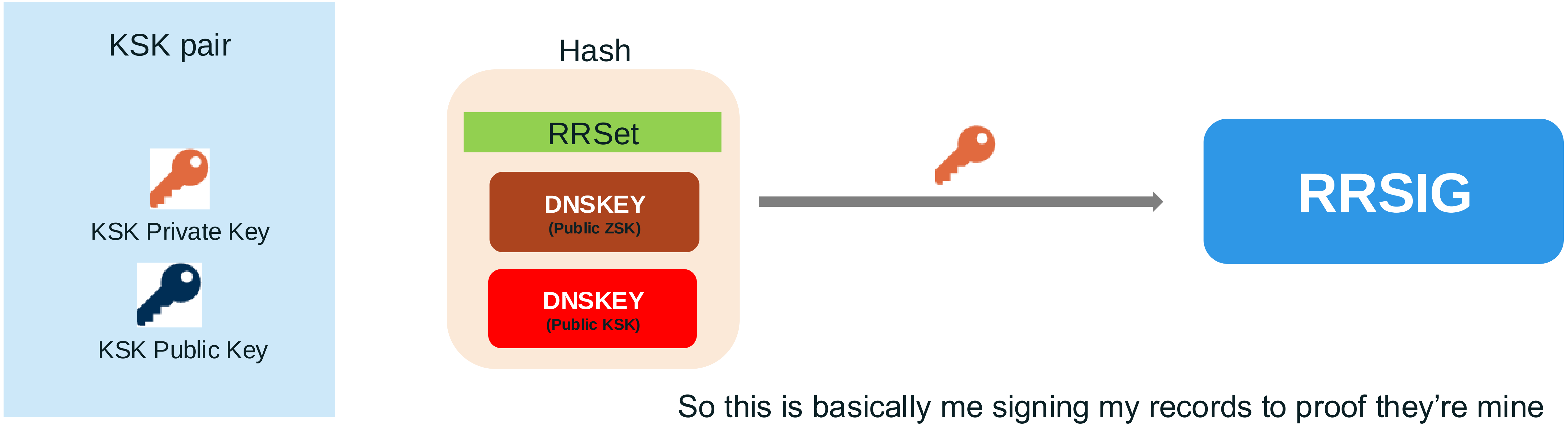
So this is basically the resolver confirming RRSet is mine

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?)

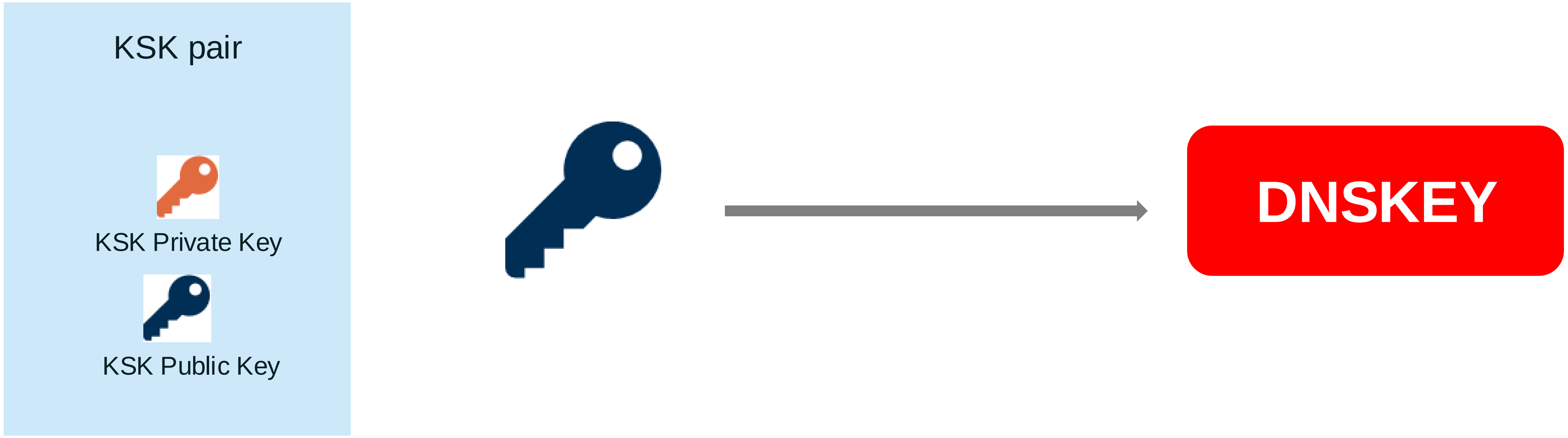
- To validate the public ZSK, DNSSEC name servers have another pair called **Key Signing Key** (KSK). This KSK works the same we explaining for ZSK by signing the public ZSK with the private KSK (private KSK encrypts DNSKEY containing both public ZSK and public KSK) and storing that signature in another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



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

KSK

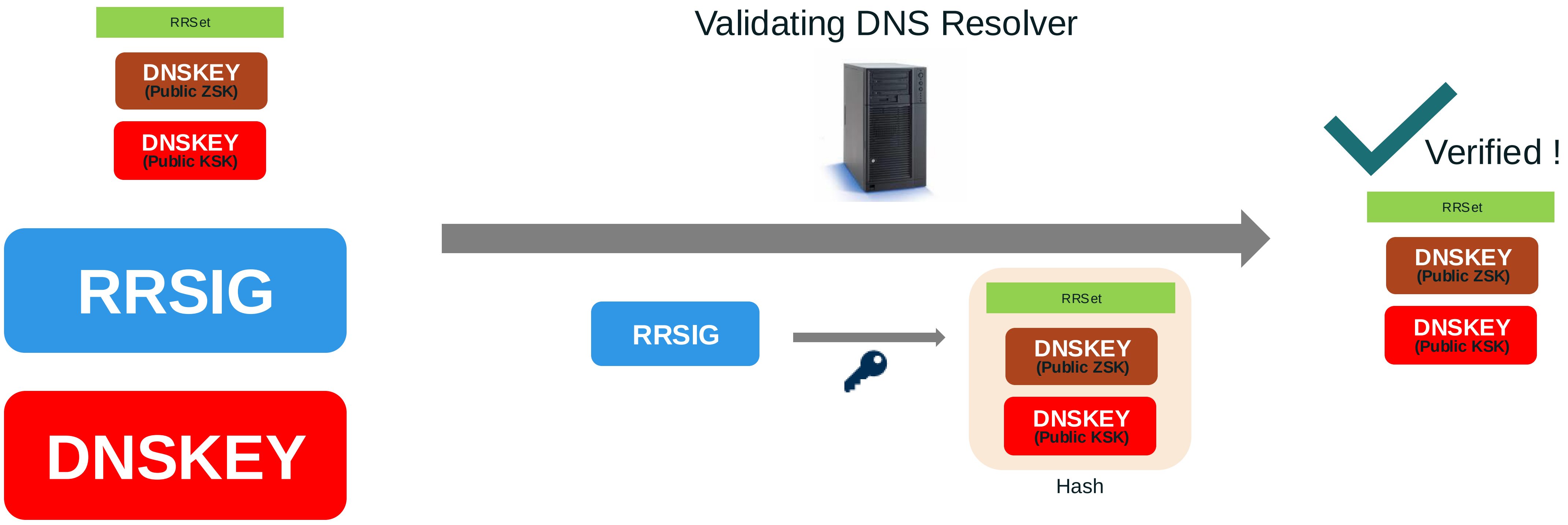
- Also, zone operators must give their public KSK for others to verify the signature. So they publish the public KSK in another DNSKEY record on their name servers.



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

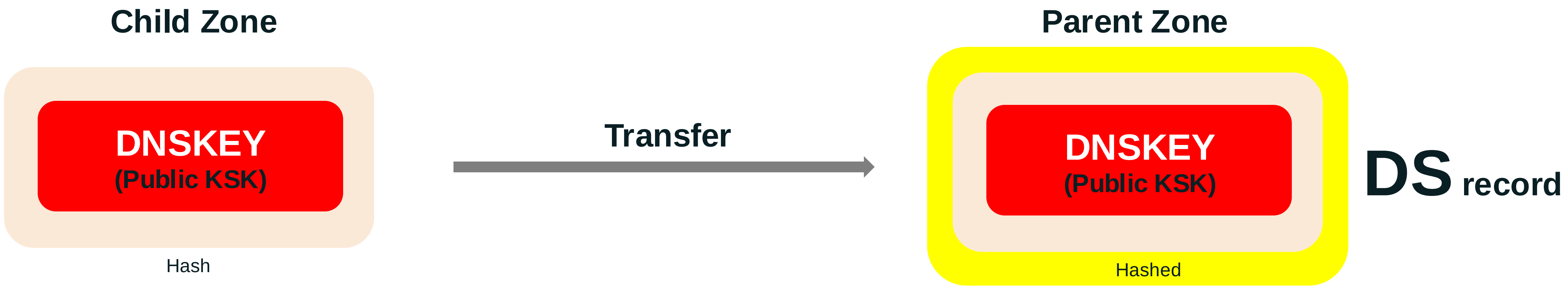
- So far, we have established trust within our zone.

... 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).



Chain of Trust

- Now when a resolver is referred to a child zone (during DNS resolution process), the referral (parent zone) also provides the DS record for that child zone.

- This way, resolver knows that the child is DNSEC-enabled and has a means to validate child's zone public KSK (hashing child's public KSK and comparing with DS from parent).

… and following this though all resolution process is how chain of trust is established down from the DNS root.

- Note that this requires to change parent's zone DS record each time a child zone changes it's KSK. So we must take special care of doing KSKs changes in a consistent way to avoid breaking the zone:

    1. First parent add the new DS record.
    2. Wait for original DS record TTL to expire.
    3. Remove original DS record.

And this is the reason changing ZSK is easier than changing KSK.

Chain of Trust

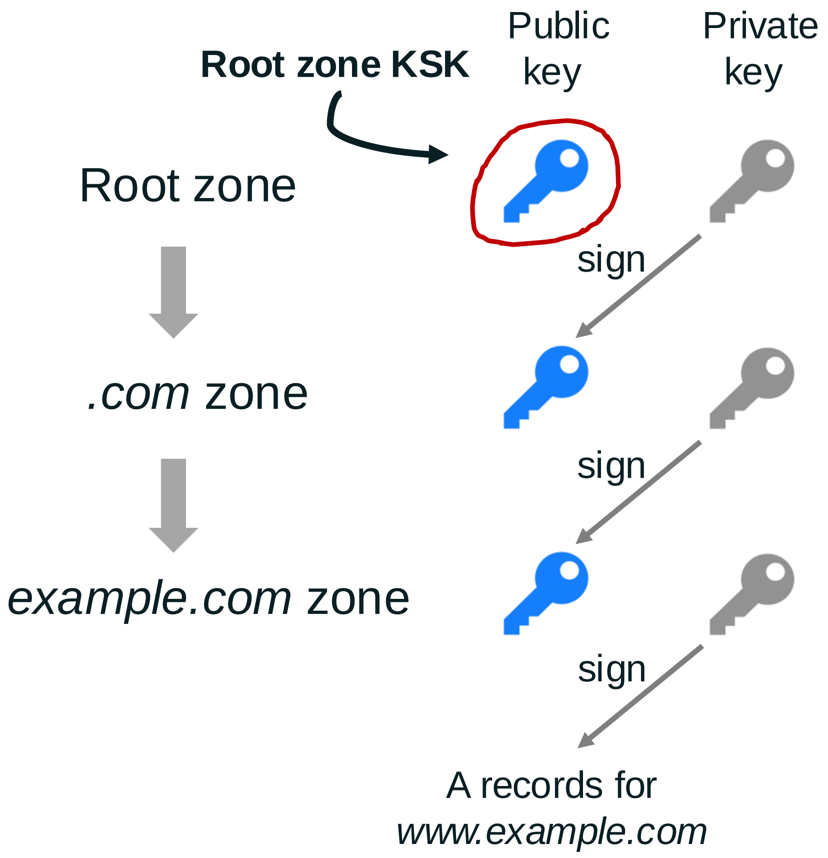
- Finally, how do we trust DS record?

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

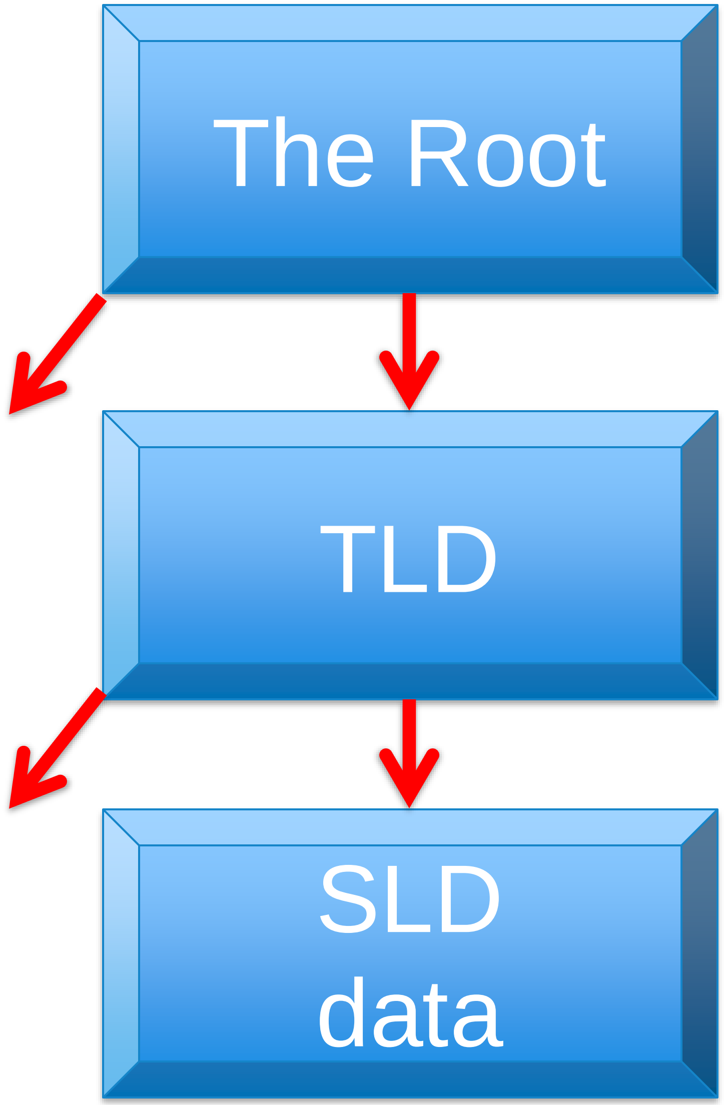
We repeat the validation process and get to the parents public KSK... And again must go to that parent's DS record to verify… on and on up to the DNS root.

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

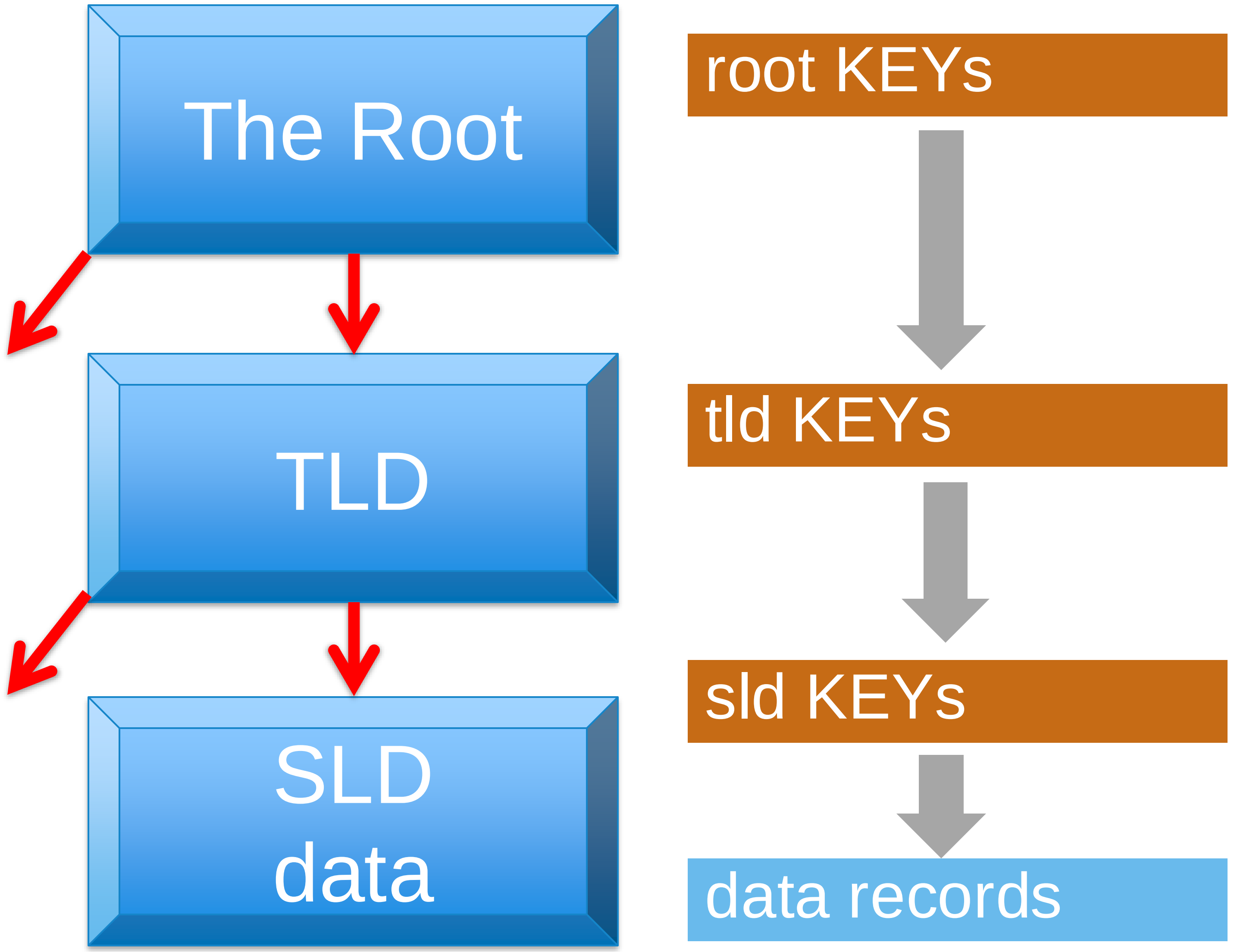


DNS's Data Organization



- The DNS is a federation of owners (registrants) of information
  - Each owner signs their own data with their own key
  - That's a lot of keys
- Hierarchy to the rescue!

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

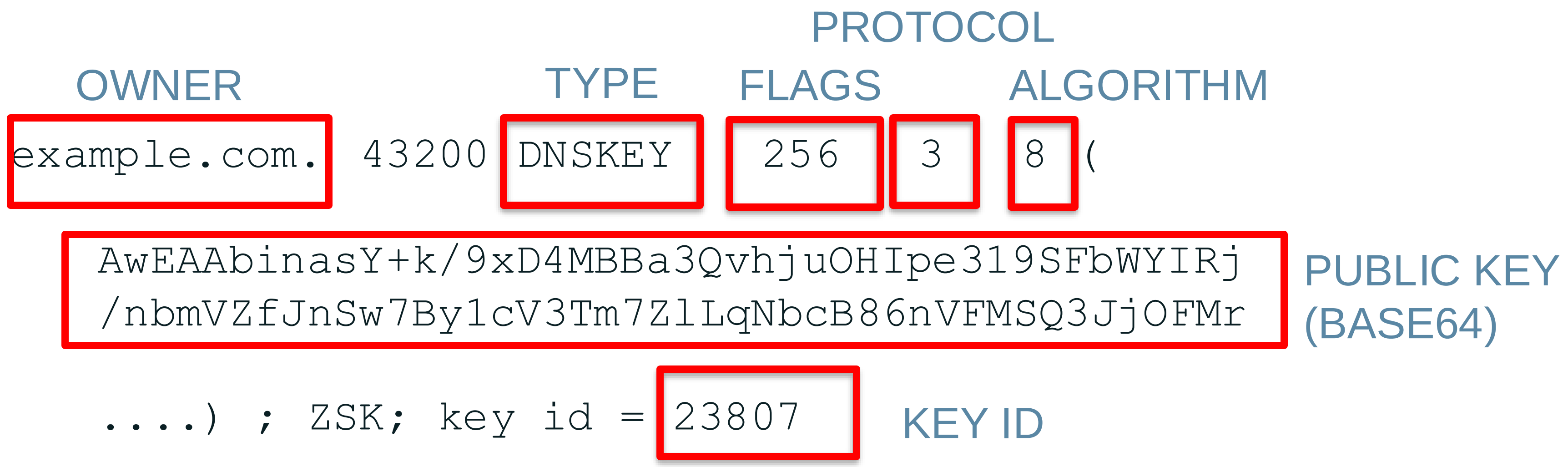
This creates a "chain" used in validation

Resource Records

- Adds following DNS Resource Records:

1. DNSKEY: Public key used in zone signing operations.
2. RRSIG: RRset signature
3. NSEC &
4. NSEC3: Returned as verifiable evidence that the name and/or RR type does not exist
5. DS: 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).

RR: DNSKEY



```
example.com. 43200 DNSKEY 256 3 8 (
  AwEAAbinasY+k/9xD4MBBa3QvhjuOHIpe319SFbWYIRj
  /nbmVZfJnSw7By1cV3Tm7ZlLqNbcB86nVFMSQ3JjOFMr
  ....) ; ZSK; key id = 23807
```

- 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
```

```
                            TYPE COVERED #LABELS
OWNER               TYPE                ALG           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
  . . .
  )
```

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

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
```

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

Questions?

ICANN

Engage with ICANN – Thank You and Questions

ICANN

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