

DNS Security

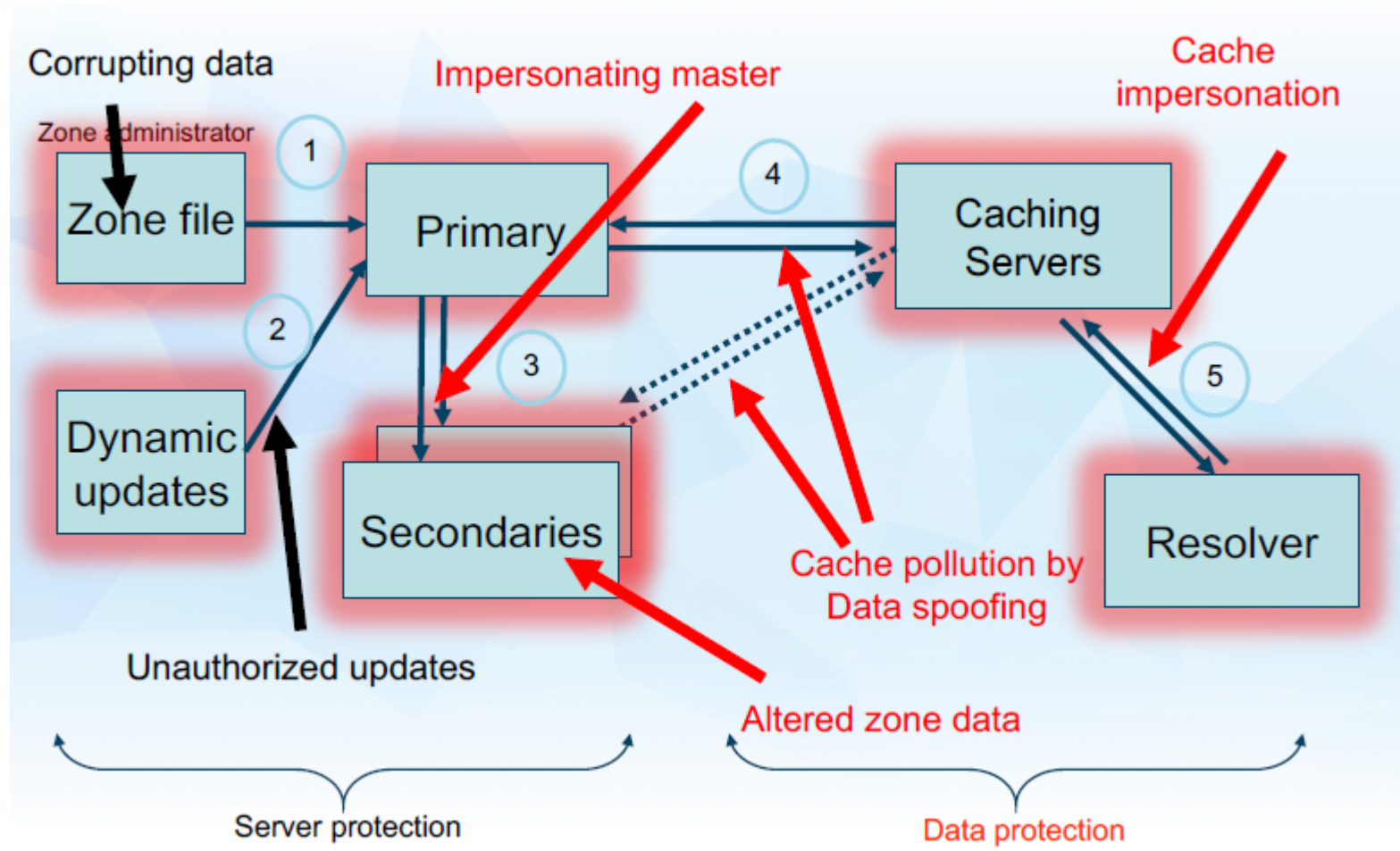
Anoop Kumar Pandey
Principal Technical Officer
Centre for Development of Advanced Computing (C-DAC)
Electronics City, Bangalore 560 100

Centre of Excellence in DNS Security
22nd May 2020

Agenda

- DNSSEC
 - Need
 - Introduction
 - New Record Types
- Working of DNSSEC
- Implementation of DNSSEC

Need for DNSSEC



DNSSEC

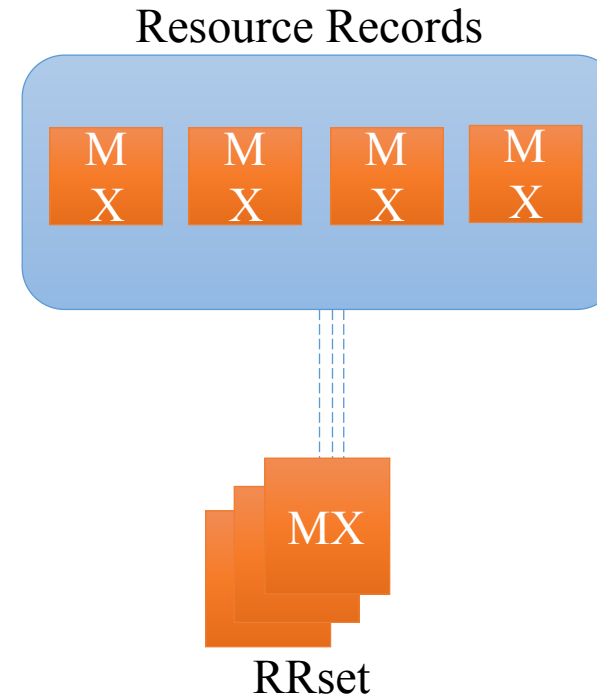
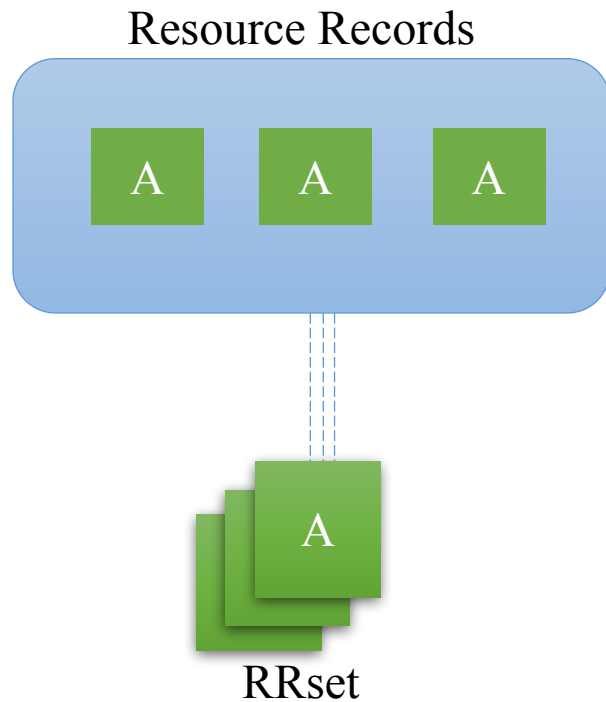
- Adds a layer of Trust through authentication
 - Adds cryptographic signature to existing DNS Records
- Verify Signature
 - Data coming from authoritative server
 - Ensures
 - No modification en-route
 - No fake record injection
- Hierarchical Trust Model

New 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.

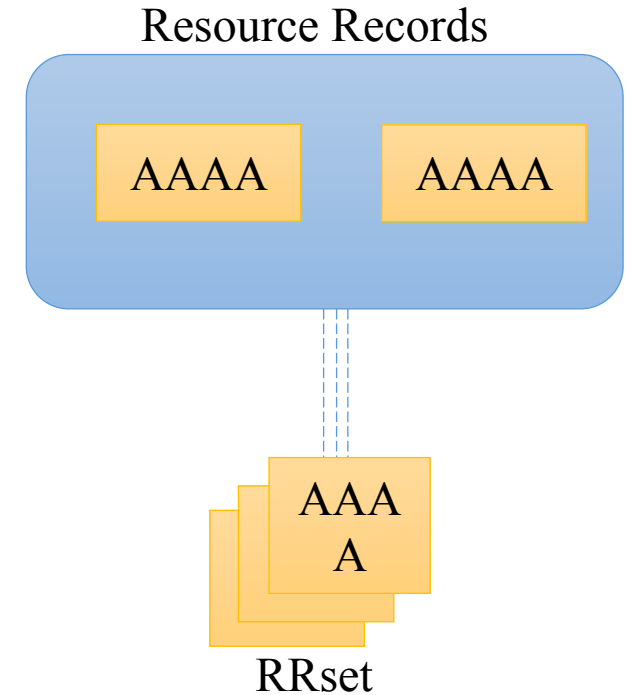
RRset

- All the records with the same type (A, AAAA etc) on the same label (i.e. www.microsoft.com)



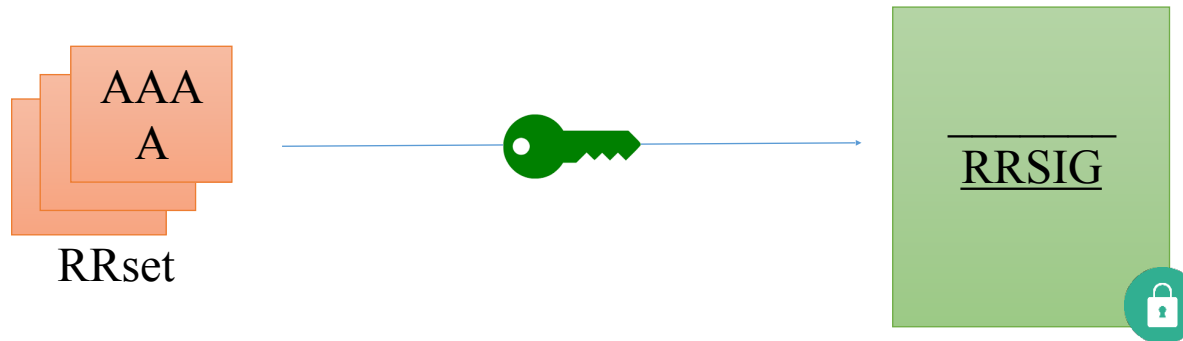
RRset

- Full RRset is signed
 - Not one record
- Request and validate entire RRset for one label
 - Validating one won't suffice

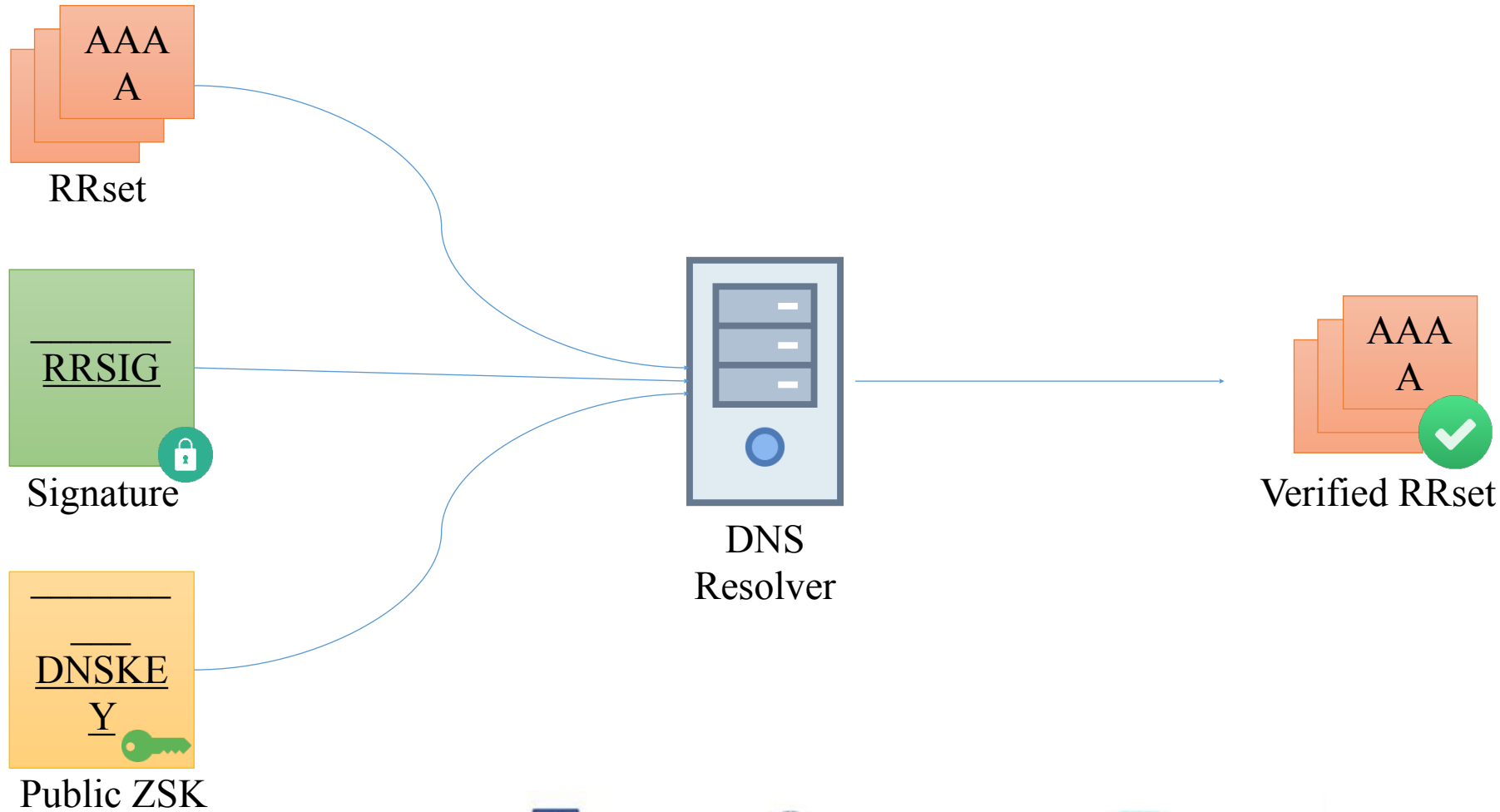


Zone Signing Key (ZSK)

- Key Pair
 - Private key to sign RRset
 - Public key to verify the signature



Zone Signing Key (ZSK)

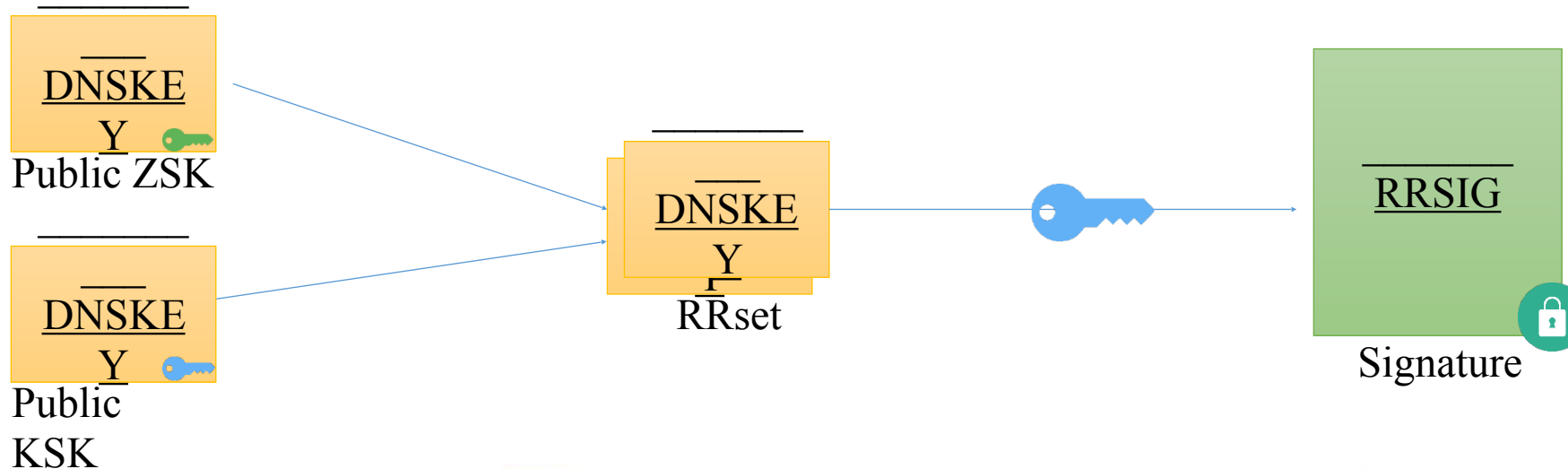


Zone Signing Key (ZSK)

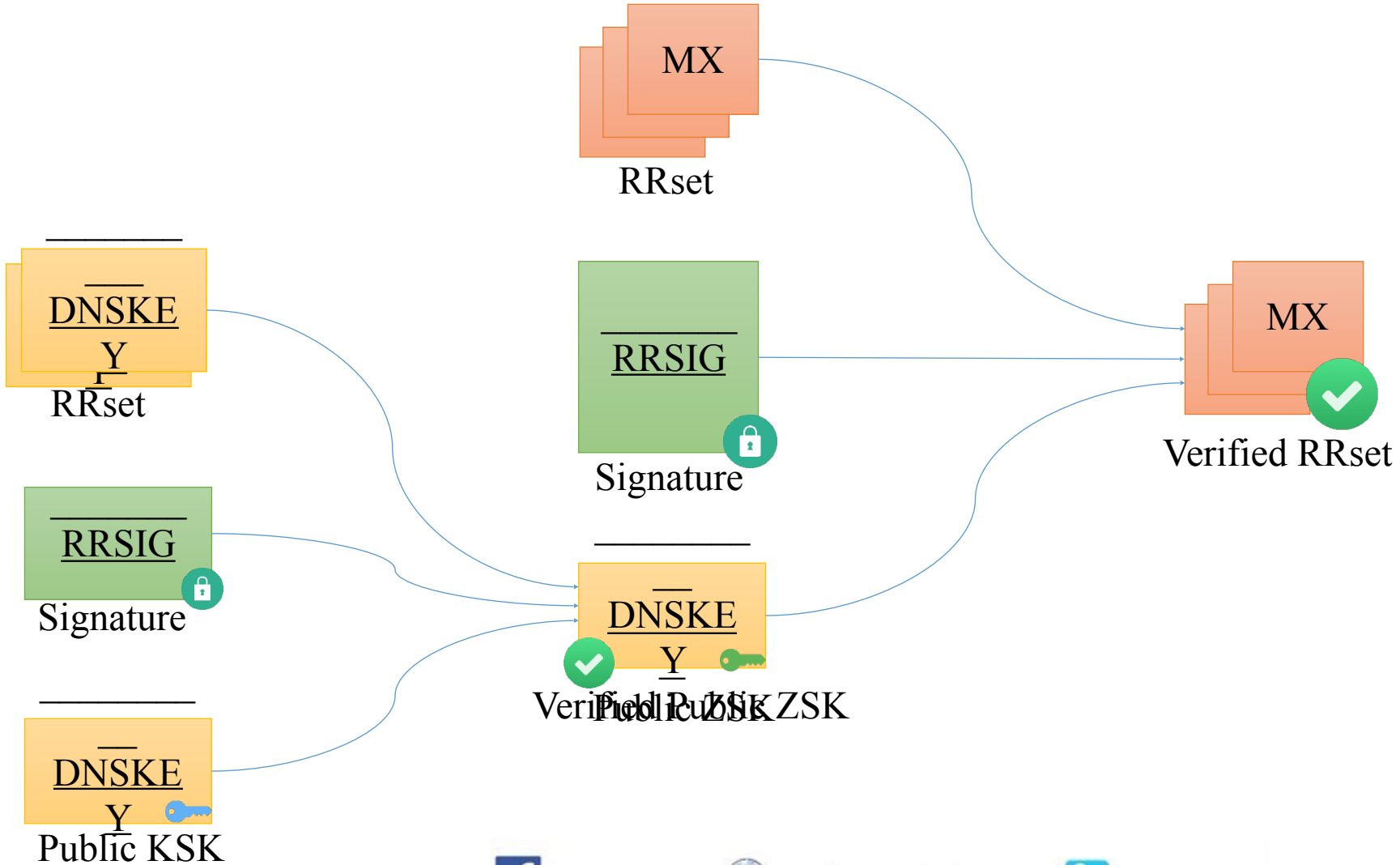
- ZSK can validate records
- But what if ZSK is compromised?

Key Signing Key (KSK)

- Signs public ZSK (stored in DNSKEY)
 - Creates RRSIG for DNSKEY
- Public KSK is stored in another DNSKEY record
- Both Public ZSK and KSK are signed using private KSK
- Resolvers use public KSK to validate public ZSK



Record Verification

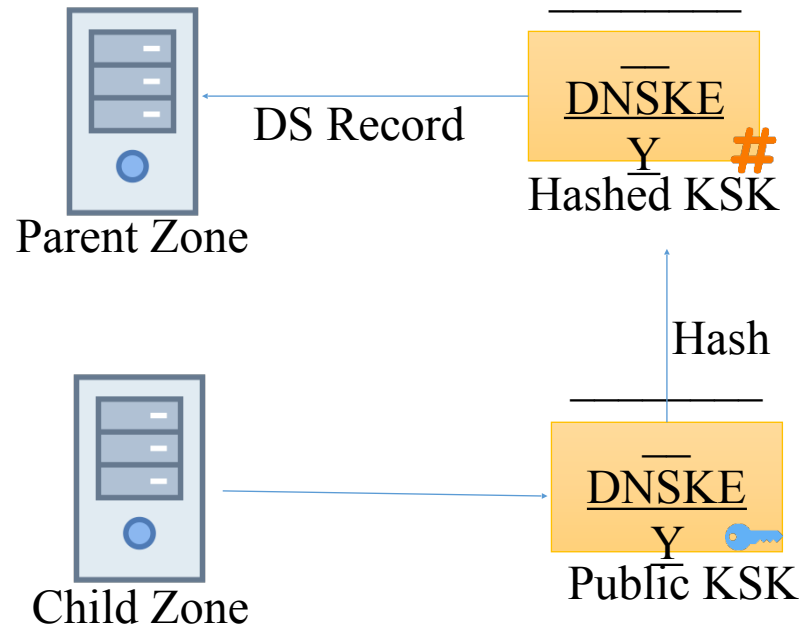


DNS Hierarchy

- Trust established in self zone
- But KSK is self-signed (Trust ends there itself)
- Trust should be propagated to parent zone for full trust

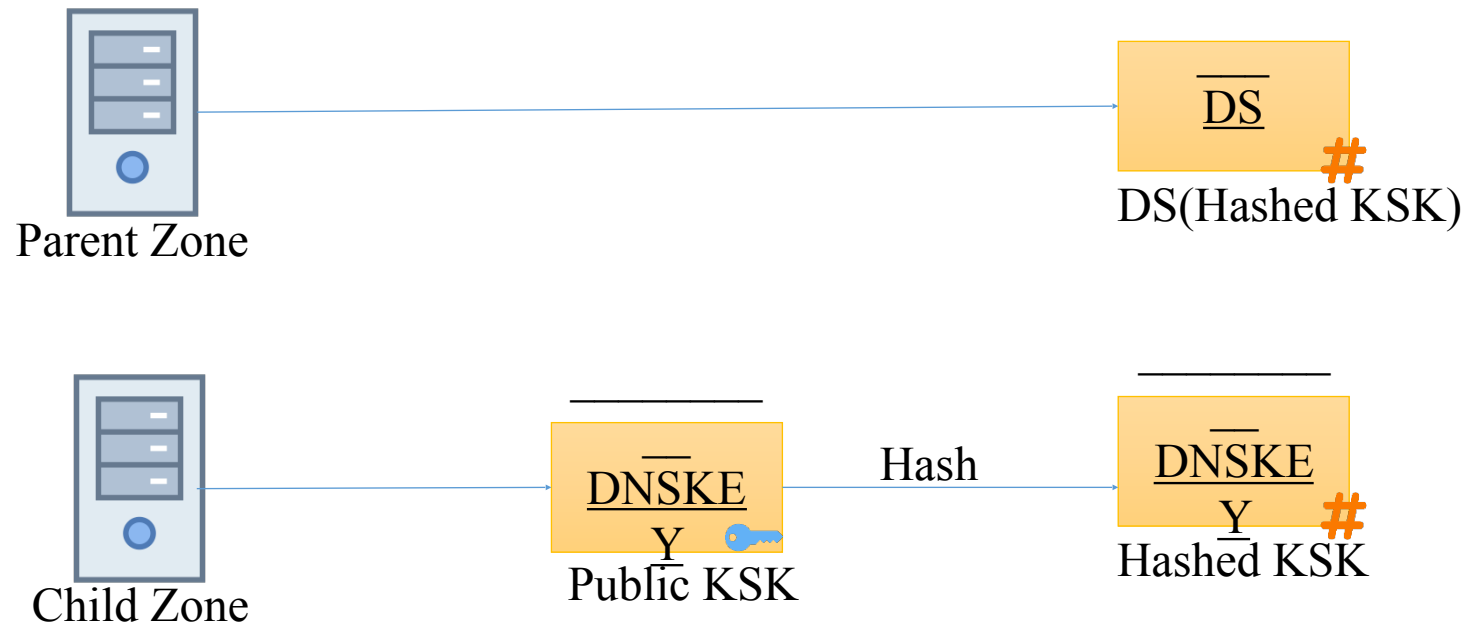
Delegation Signer Records

- Hash of DNSKEY record containing Public KSK
- Published as a DS record in Parent Zone
- DS Record establishes that child zone is DNSSEC enabled.

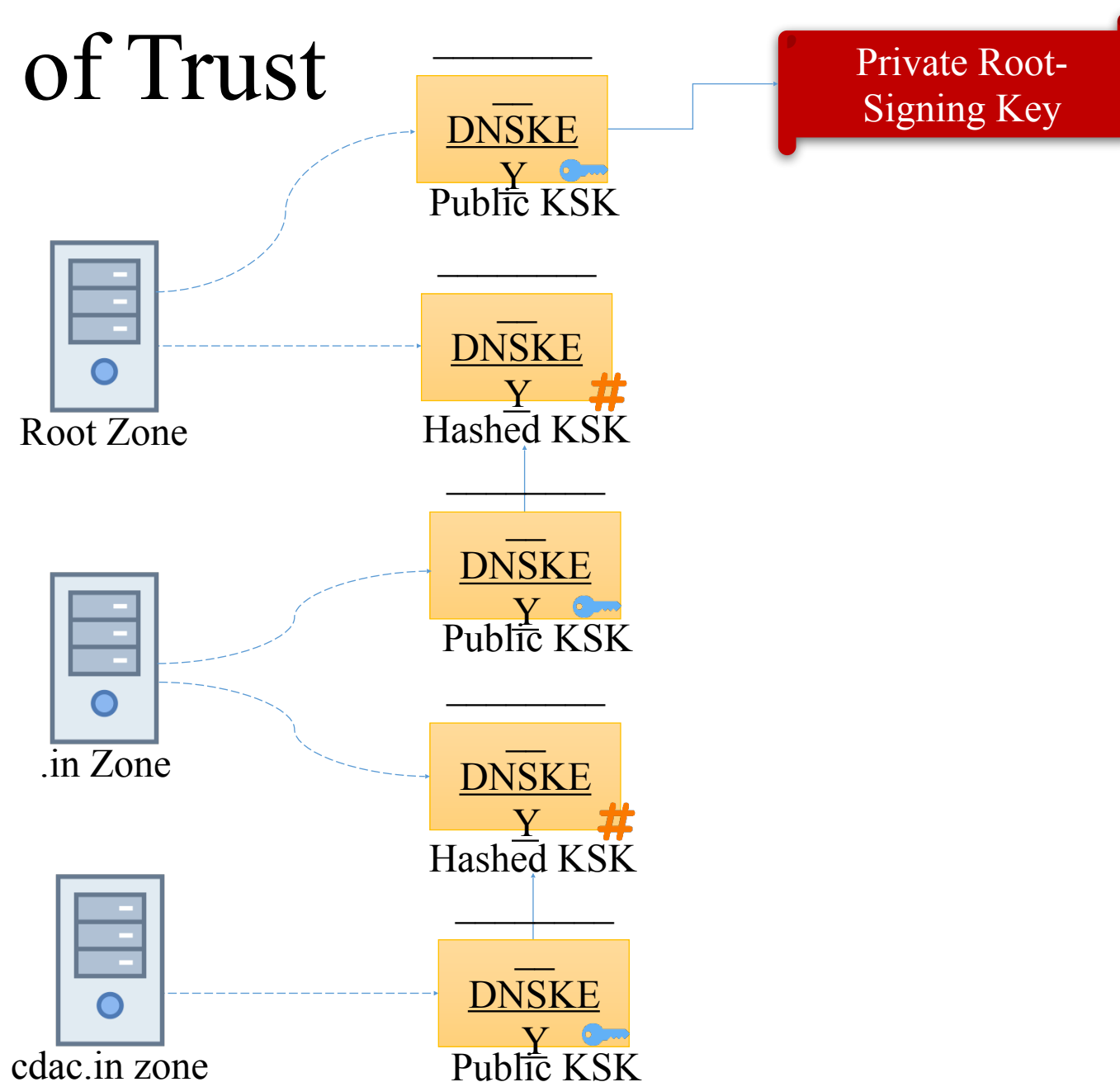


Delegation Signer Records

- Verification of Child Zone KSK




Chain of Trust





NSEC/NSEC3 Record

- For non-existent Domain
 - Empty answer
 - What to sign??
- Explicit authenticated denial of existence of a record
- NSEC (Next Secure) Record
 - Returns the “next secure” record for a non-existent domain (alphabetical order)
 - ‘A’ record for *auth*, *drive*, *mail*, *www*
 - Request for ‘A’ record for *smtp*  NSEC Record containing *www* will be returned
 - Record can be verified
 - Prone to zone walking
 - Security threat if some sub-domains need to be kept private
- NSEC3 Record
 - Uses cryptographically hashed record names to avoid the enumeration of the record names in a zone.

Summary

- DNSSEC Guarantees:
 - Authenticity of DNS answer origin
 - Integrity of reply
 - Authenticity of denial of existence
- Accomplishes this by signing DNS replies at each step in the hierarchy
- Uses public-key cryptography to sign responses
- Typically use trust anchors, entries in the OS to bootstrap the process

Summary

- DNSSEC does not
 - Provide confidentiality for DNS data
 - Protect against Denial of Data

Implementation [Ubuntu]

Create keys folder in /var/cache/bind/ and generate keys in that folder

- `dnssec-keygen -a ECDSAP256SHA256 -3 anoop.in`
- `dnssec-keygen -f KSK -a ECDSAP256SHA256 -3 anoop.in`
- `chown bind:bind -R keys`
- `nano /etc/bind/named.conf.local`

```
zone "anoop.in" IN {  
    type master;  
    file "/var/cache/bind/zones/anoop";  
    key-directory "/var/cache/bind/keys/";  
    auto-dnssec maintain;  
    inline-signing yes;  
};
```

Implementation

- `named-checkconf [optional]`
- `named-checkzone anoop.in anoop [optional]`
- `rndc reload`
- `rndc reconfig`
- `systemctl status named [optional]`
- `dig email.anoop.in @localhost +dnssec +multiline`
- `dig @localhost dnskey anoop.in | dnssec-dsfromkey -f - anoop.in`

Thank You

Digital Signatures

What is a Digital Signature ?

- A *Digital signature* of a message is a **number (fingerprint)** dependent on
 - a secret known only to the signer **and**
 - the content of the message being signed
- Digital Signatures can be
 - Verified for Authenticity
 - Verified for Integrity
 - Verified for Non-Repudiation

```
00000000230000000d000000726573705f69646556e746966790000000000000000
6170695f696e666f2300000000000000000000000000000000000000000000000000
000000002300000009000000726573705f696e666f000000000000000000000000
6170695f737461747323000000000000000000000000000000000000000000000000
00000000230000000a000000726573705f7374617473000000000000000000000000
6170695f61757468656e746966792378616a505579506d0000000000000000000000
00000000230000000f000000726573705f61757468656e7469667900000000000000
6170695f656e637279707423626c4343797966780000000000000000000000000000
000000002300000008000000202e01013b3b243a0000000000000000000000000000
6170695f646563727970742372494d586c794f4a0000000000000000000000000000
00000000238b040808000000300b0f1a2e3b0d080000000000000000000000000000
6170695f627965230000000000000000000000000000000000000000000000000000
000000002300000008000000726573705f6279650000000000000000000000000000
6170695f69646556e74696679234e7a77754a715143000000000000000000000000
00000000234300000d000000726573705f69646556e746966790000000000000000
```

Creating Digital Signature

- Every individual is given a pair of **keys**
 - *Public key* : known to everyone
 - *Private key* : known only to the owner
- To *digitally sign* an electronic document the signer uses his/her *Private key*
- To *verify* a digital signature the verifier uses the signer's *Public key*

What is a key pair?



Private Key

```
3082 010a 0282 0101 00b1 d311 e079 5543 0708 4ccb 0542 00e2
0d83 463d e493 bab6 06d3 0d59 bd3e c1ce 4367 018a 21a8 efbc
ccd0 a2cc b055 9653 8466 0500 da44 4980 d854 0aa5 2586 94ed
6356 ff70 6ca3 a119 d278 be68 2a44 5e2f cfcc 185e 47bc 3ab1
463d 1ef0 b92c 345f 8c7c 4c08 299d 4055 eb3c 7d83 deb5 f0f7
8a83 0ea1 4cb4 3aa5 b35f 5a22 97ec 199b c105 68fd e6b7 a991
942c e478 4824 1a25 193a eb95 9c39 0a8a cf42 b2f0 1cd5 5ffb
6bed 6856 7b39 2c72 38b0 ee93 a9d3 7b77 3ceb 7103 a938 4a16
6c89 2aca da33 1379 c255 8ced 9cbb f2cb 5b10 f82e 6135 c629
4c2a d02a 63d1 6559 b4f8 cdf9 f400 84b6 5742 859d 32a8 f92a
54fb ff78 41bc bd71 28f4 bb90 bcff 9634 04e3 459e a146 2840
8102 0301 0001
```

Public Key

```
3082 01e4 f267 0142 0f61 dd12 e089 5547 0f08 4ccb 0542 00e2 0d83 463d
e493 bab6 0673 0d59 bf3e c1ce 4367 012a 11a8 efbc ccd0 a2cc b055 9653
8466 0500 da44 4980 d8b4 0aa5 2586 94ed 6356 ff70 6ca3 a119 d278 be68
2a44 5e2f cfcc 185e 47bc 3ab1 463d 1df0 b92c 345f 8c7c 4c08 299d 4055
eb3c 7d83 deb5 f0f7 8a83 0ea1 4cb4 3aa5 b35f 5a22 97ec 199b c105 68fd
e6b7 a991 942c e478 4824 1a25 193a eb95 9c39 0a8a cf42 b250 1cd5 5ffb
6bed 6856 7b39 2c72 38b0 ee93 a9d3 7b77 3ceb 7103 a938 4a16 6c89 2aca
da33 1379 c255 8ced 9cbb f2cb 5b10 f82e 6135 c629 4c2a d02a 63d1 6559
b4f8 cdf9 f400 84b6 5742 859d 32a8 f92a 54fb ff78 41bc bd71 28f4 bb90
bcff 9634 04de 45de af46 2240 8410 02f1 0001
```



Digital Signing – Step 1

This is an example of how to create a message digest and how to digitally sign a document using Public Key cryptography

Hash

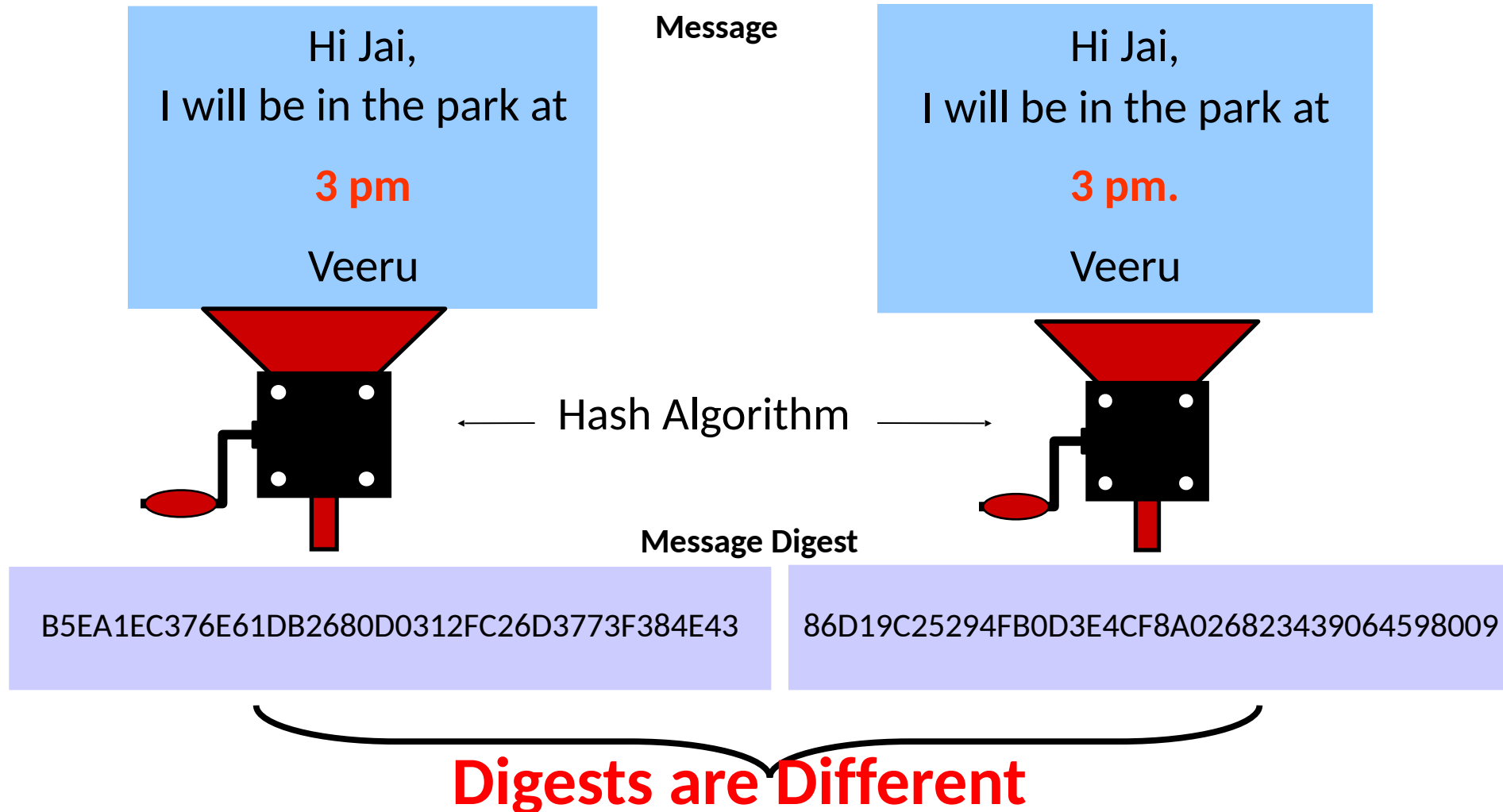
Message
Digest

Hash Function

- A hash function is a cryptographic mechanism that operates as **one-way** function
 - Creates a digital representation or "fingerprint" (Message Digest)
 - **Fixed size output**
 - Change to a message produces different digest

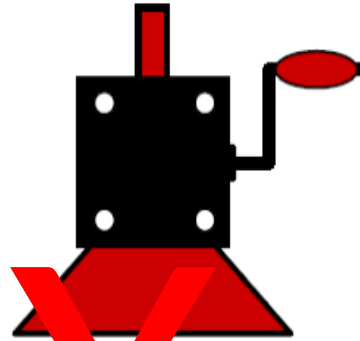
Examples : MD5 , Secure Hashing Algorithm (SHA)

Hash - Example



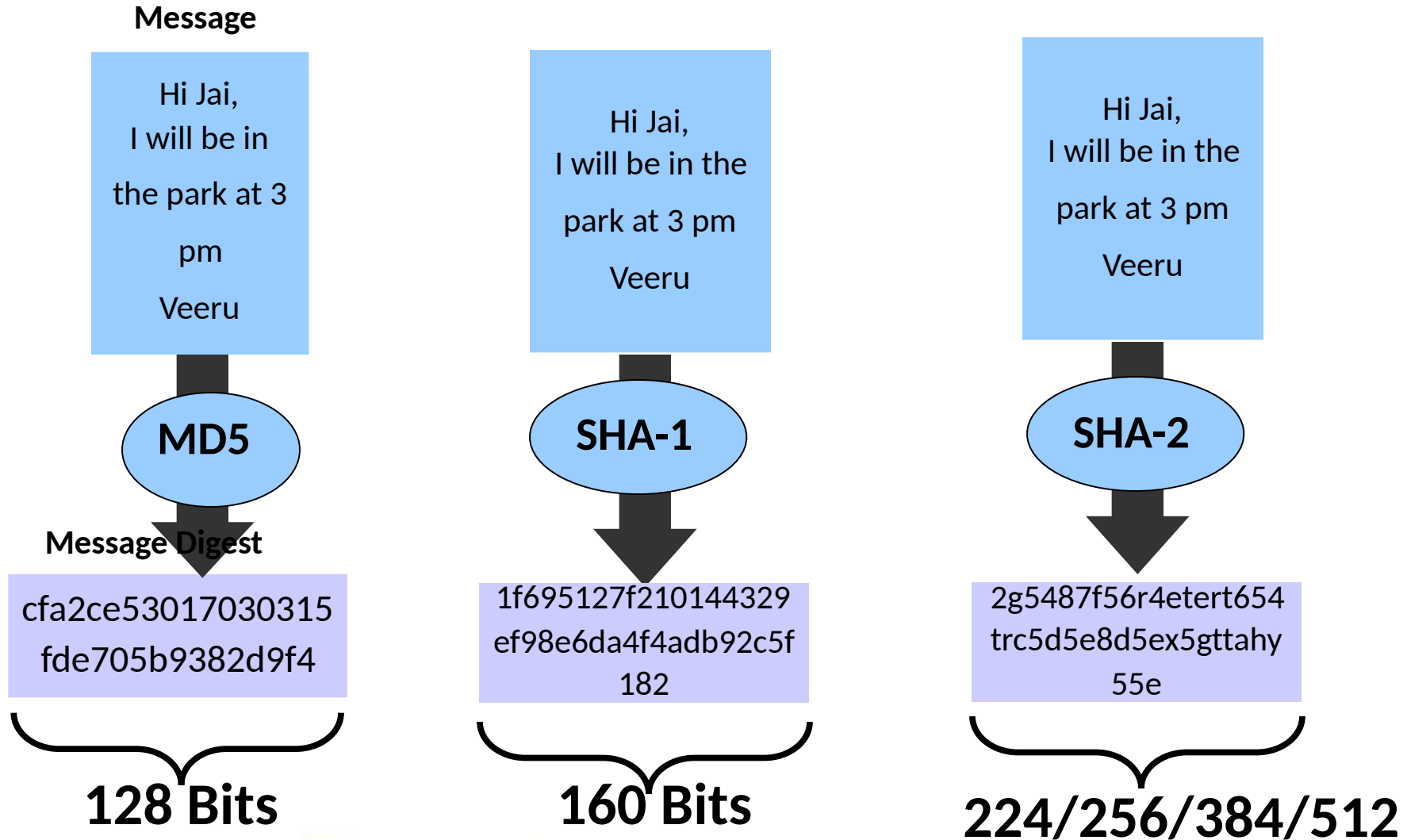
Hash – One-way

B5EA1EC376E61DB2680D0312FC26D3773F384E43



li Jai,
I will be in the park at
3 pm
Veeru

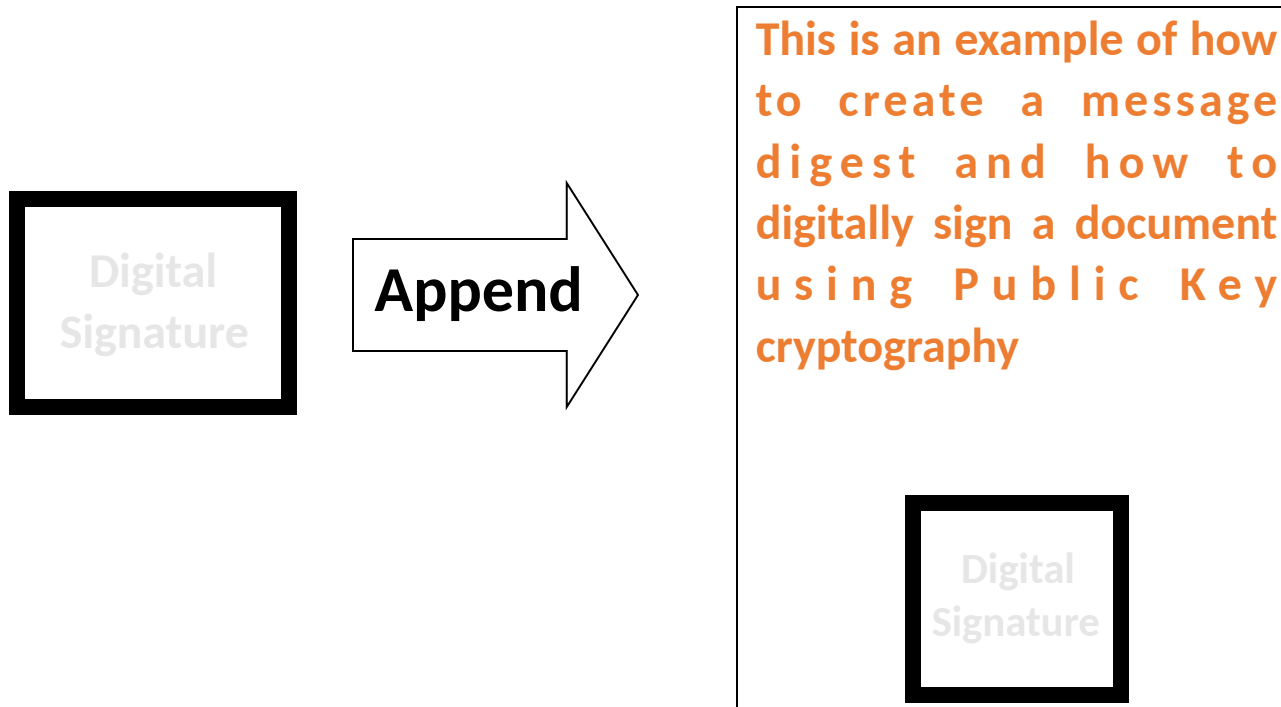
MD5 and SHA



Digital Signing – Step 2



Digital Signing – Step 3



Digital Signature Verification

This is an example of how to create a message digest and how to digitally sign a document using Public Key cryptography

Digital
Signature

Hash

Message
Digest

Decrypt with
public key

Message
Digest