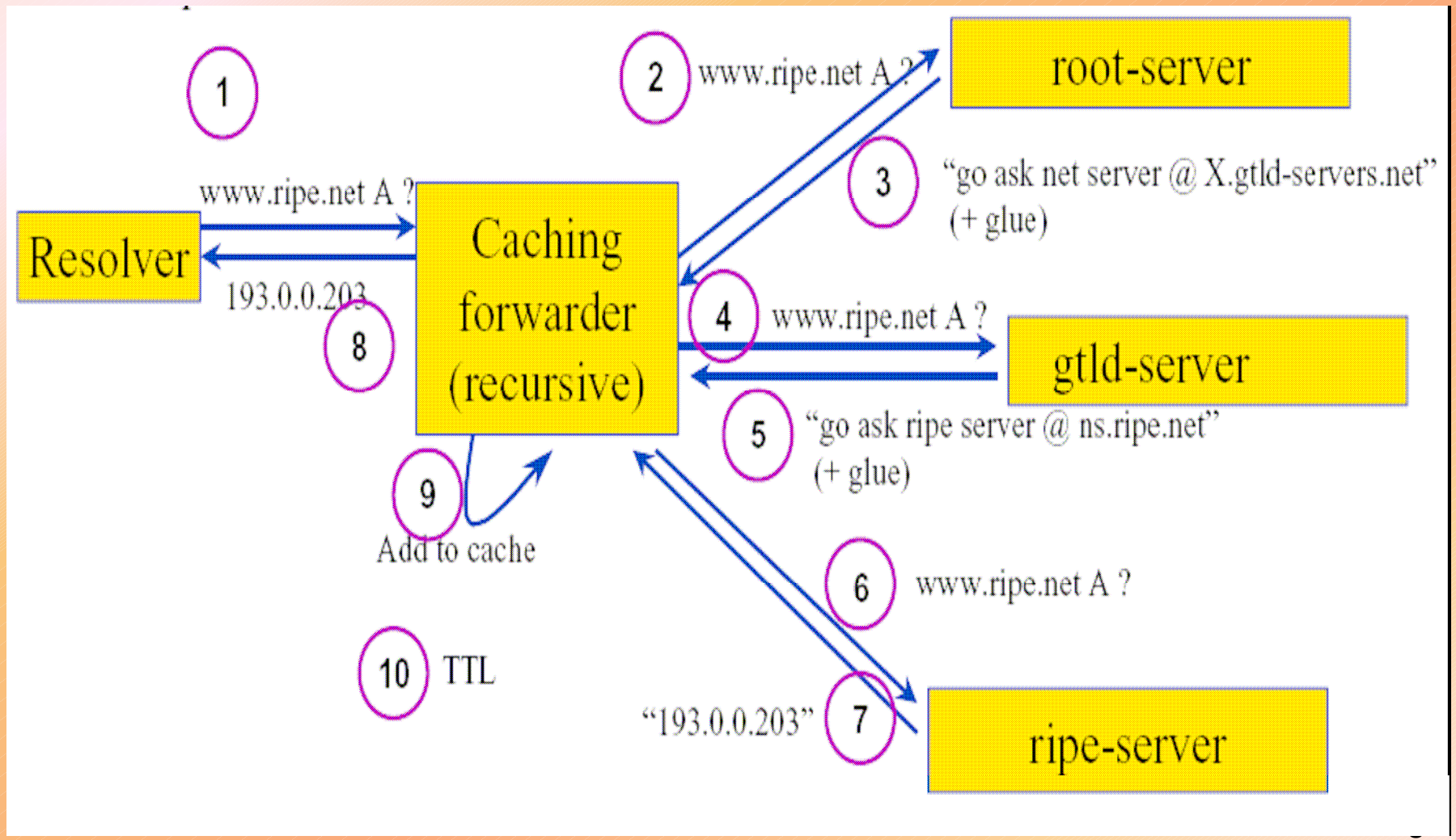
DNSSEC

Perché DNSSEC ?

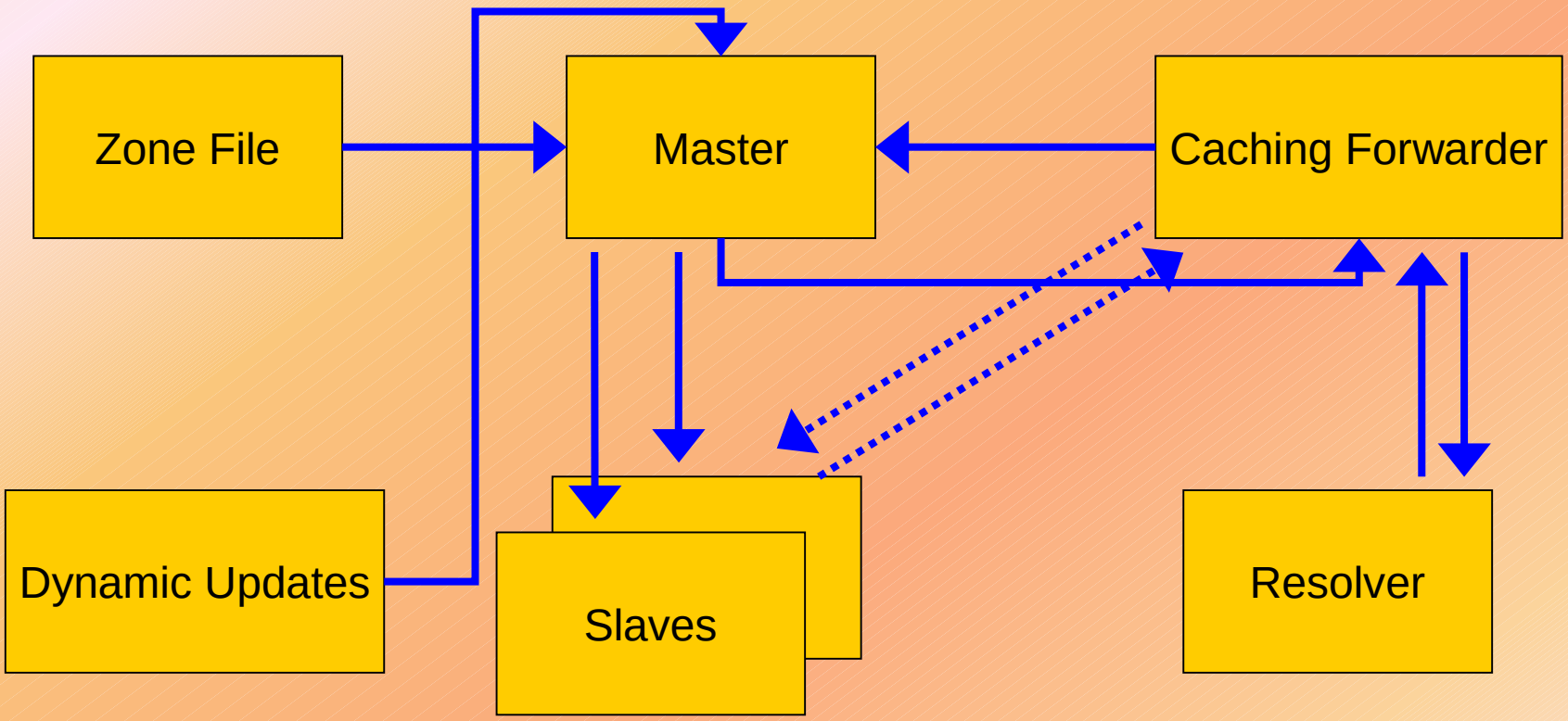
- DNS NON è sicuro
  - Vulnerabilità pubblicamente conosciute
  - DNS è un servizio di grande importanza

- DNSSEC protegge dallo spoofing e dalla alterazione dei dati del traffico DNS

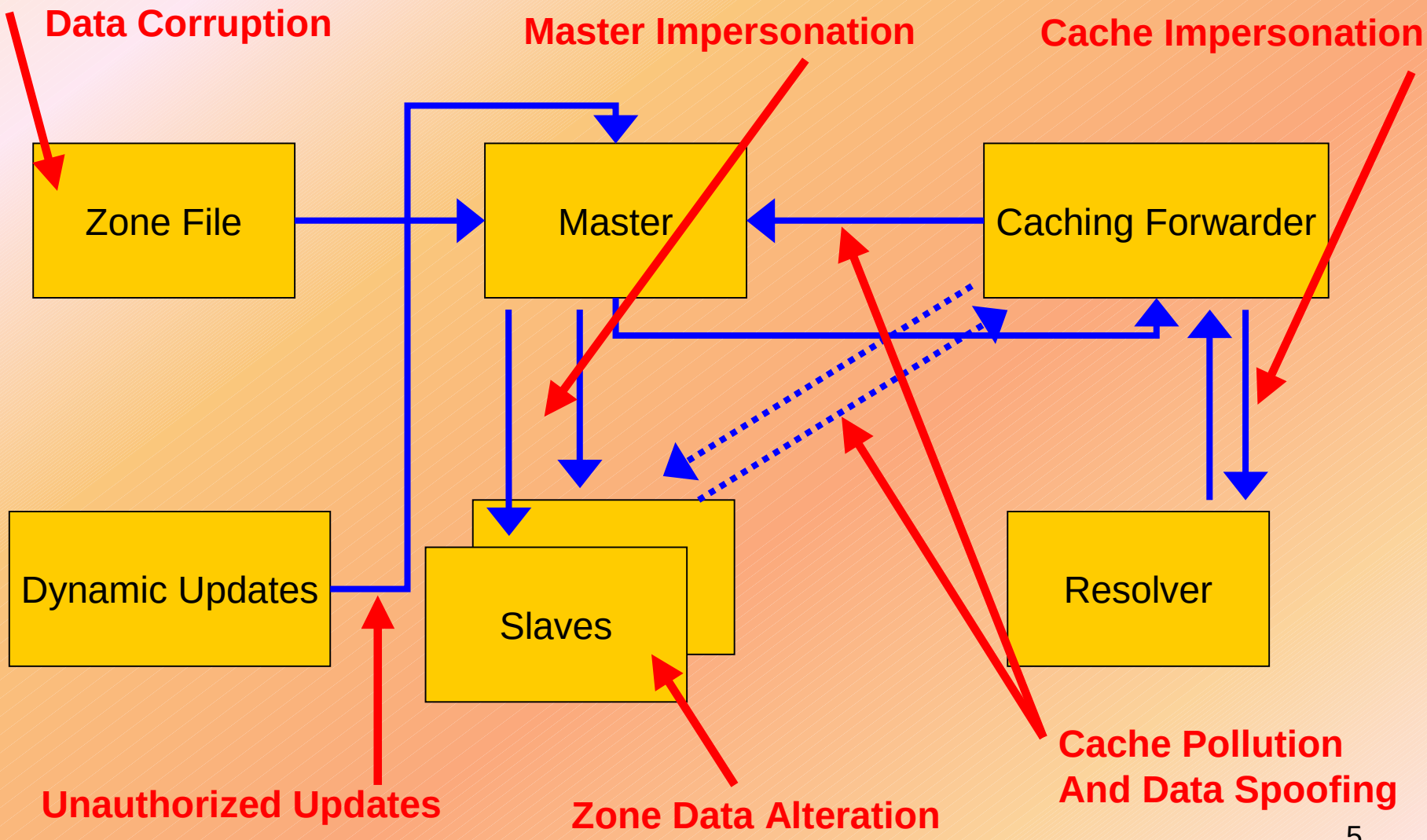
Risoluzione DNS



DNS: Data Flow



DNS: Vulnerabilità



DNS: Vulnerabilità Protocollo

- I dati Dns trasmessi tra il *name-server master*, il *forwarder* ed il *resolver* possono essere contraffatti (*spoofing*) e corrotti.
- Il protocollo Dns non consente di controllare la validità dei dati trasmessi dal protocollo stesso
  - Bugs nell'implementazione del protocollo sul resolver (ID di transazione prevedibile)
  - Cache-Pollution dei forwarders può causare danni per un certo periodo di tempo (TTL)
  - Dati Dns corrotti possono finire nella cachee rimanervi a lungo
- Impossibilità per il name-server slave (secondary) di accertare se ha instaurato la comunicazione con il giusto master (primary) server

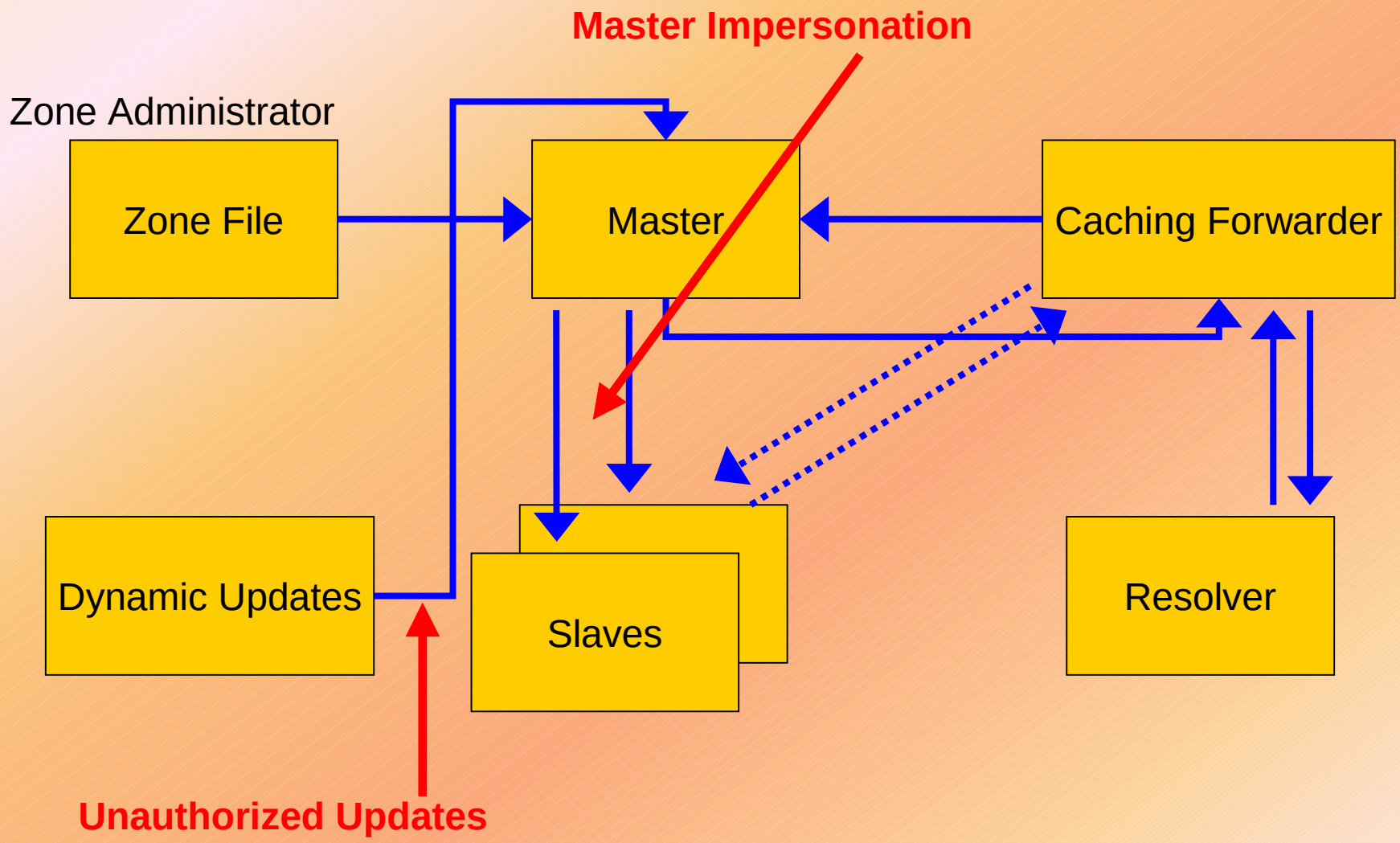
DNSSEC: Soluzioni

- DNSSec protegge contro il *data spoofing* ed il *data corruption*
  - TSIG/SIG0: fornisce un meccanismo per autenticare la comunicazione tra server DNS
  - DNSKEY/RRSIG/NSEC: fornisce un meccanismo per stabilire l'autenticità e l'integrità dei dati Dns
  - DS: fornisce un meccanismo per delegare una relazione di *trust* a chiavi pubbliche di terze parti
- Il DNS sicuro necessita di **un'infrastruttura a chiave pubblica**, tuttavia non stiamo parlando di una vera e propria PKI.

DNSSEC: RFC

- RFC 4033
  - DNS Security Introduction and Requirements
- RFC 4034
  - Resource Records for the DNS Security Extensions
- RFC 4035
  - Protocol Modifications for the DNS Security Extensions

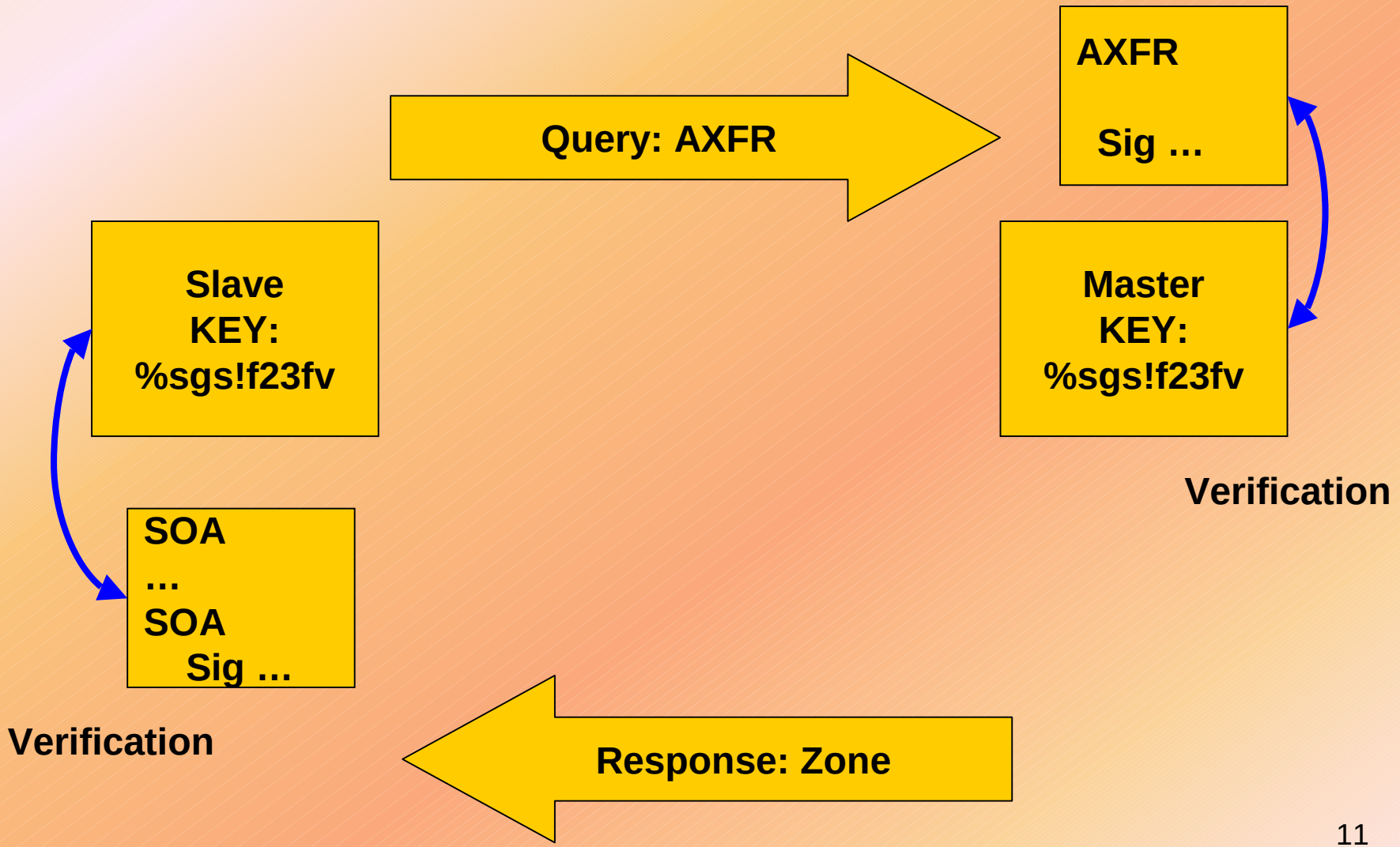
TSIG: Vulnerabilità Affrontate



TSIG: Transaction Signature

- TSIG (RFC 2845)
  - Autorizzazione dei *dynamic updates* e dei *zone tranfers*
  - Autenticazione dei *caching forwarders*
  - Usabile anche indipendentemente dalle altre componenti di DNSSEC
- Funzione di Hashing
  - Interrogazioni Dns, Risposte Dns e Timestamp
- Il traffico viene firmato con una chiave di tipo ***shared secret***
- Specificato nel file di configurazione del servizio e NON nel file di zona

TSIG: Esempio



TSIG: Zone Transfer

- Generazione ***segreto***
- Comunicazione ***segreto***
- Configurazione Servers
- Test

TSIG: Generazione Segreto

- Algoritmo: HMAC-MD5
- Bits: 256 o maggiore
- Nome: identificatore univoco

- Il ***segreto*** può essere anche generato in modo differente, ad esempio con una codifica BASE64
- Il TSIG non dovrebbe mai essere inserito nei files di zona, in quanto potrebbe confondere in virtù del suo apparire come un normale Resource Record
  me-friend. IN KEY 512 3 157 nEfRX9…bbPn7l=

Master Server: named.conf

- Configurazione della **Key**
  - *Key “me-friend.” {*
    algorithm hmac-md5;
    secret “nEfRX9jxOmzsby8VKRgDWE”;
    };
- Opzione ***allow-transfer*** nel blocco di zona per indicare di quale chiave sia consentita la trasmissione
  - Può essere combinato con restrizioni basate sugli indirizzi IP
  - *Zone “example.net” {*
    type master;
    file “zones/examples.net”;
    allow-transfer { key me-friend.; };
    notify yes;
    };

Slave Server: named.conf

- Configurazione della **Key**
  - *key “me-friend.” {*
    algorithm hmac-md5;
    secret “nEfRX9jxOmzsby8VKRgDWE”;
    };
- Blocco **Server** per indicare la chiave usata
  - la configurazione di zona non cambia sullo slave server
  - *Server 192.168.10.1 {*
    keys {me-friend.; };
    };

Test: dig

- E’ possibile usare l’utilità dig per testare la configurazione TSIG
  - Dig @<server> <zone> AXFR –k <TSIG keyfile>
  - Dig @10.0.53.204 example.net AXFR –k me-friend.+157+51197.key
- La chiave sbagliata darà un errore del tipo ‘**Transfer Failed**’ e sul server sarà registrato nei log della sicurezza:
  - **Security: error: client 193.0.0.182#1228: zone transfer ‘example.net/IN’ denied**

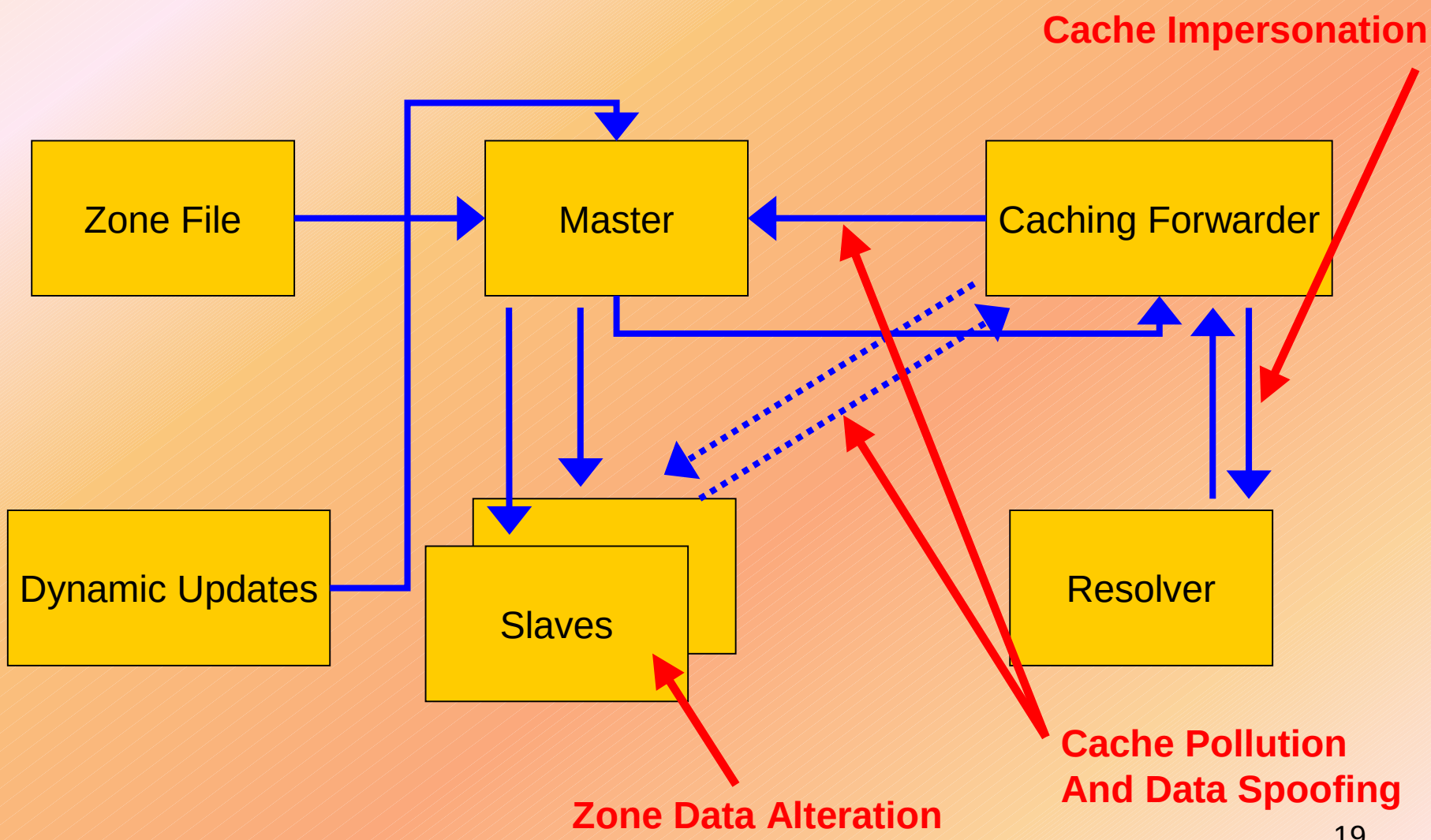
Importanza del Timestamp

- TSIG/SIG0 firma una transazione (richiesta/risposta) DNS con un *timestamp*
  - Per prevenire attacchi di tipo ***replay***
- Eventuali problemi legati alle temporizzazioni
  - Necessità di una corretta definizione del local-time-zone
  - Usare la sincronizzazione NTP

SIG0: Autenticazione Servers

- In alternativa a TSIG, è possibile usare SIG0
  - Non ancora universalmente utilizzato
  - Funziona bene in un ambiente che usi “dynamic update”
- Algoritmo a Chiave Asimmetrica
- Specificato in RFC 2931

DNSKEY/RRSIG/NSEC: Vulnerabilità Affrontate



DNSSEC: Sommario

- L’integrità e l’autenticità dei dati ottenuta firmando i RRs con la chiave privata
- La chiave pubblica DNSKEY usata per verificare i RRSIGs
- I nodi ‘*figlio*’ firmano le proprie zone con la propria chiave privata
  - L’autenticità della chiave è stabilita dal *parent* attraverso la coppia ***signature/checksum***
- Caso ideale: una chiave pubblica DNSKEY distribuita

DNS: NON una PKI

- Tutte le procedure riguardanti le chiavi sono stabilite a livello di ***local policy***
- Una PKI è tanto robusta quanto lo è il più debole dei suoi componenti
- Il DNS non usa le Certificate Revocation Lists
- Se il dominio è sotto un solo controllo amministrativo, potresti imporre una determinata *policy*

Crittografia a Chiave Asimmetrica

- Key Pair: una chiave privata (*secret*) e la sua corrispondente chiave pubblica
- In breve …
  - Se si conosce la chiave pubblica, è possibile verificare una signature creata con la chiave privata
  - Se si conosce la chiave pubblica, è possibile cifrare dati decifrabili solamente usando la relativa chiave privata
- DNSSEC utilizza solamente le *signatures*
  - PGP usa entrambi i metodi

Autenticità ed Integrità dei Dati

- Autenticità: I dati sono pubblicati dall'entità che si ritiene ne abbia l'autorità???
- Integrità: I dati ricevuti sono i medesimi che sono stati pubblicati???
- La crittografia a chiave asimmetrica risponde alle seguenti esigenze:
  - Verificare l'integrità e l'autenticità dei dati attraverso le signatures
  - Verificare l'autenticità delle signatures

Zone Status

- **Sicuro** in modo verificabile
  - RRSet e il suo RRSIG possono essere verificati con una chiave DNSKEY ed il *parent* ha un Record DS
- **Insicuro** in modo verificabile
  - RRSet risiede in una zona che non è firmata e per la quale il *parent* non ha un Record DS
- **BAD**
  - RRSet ed il suo RRSIG non può essere verificato (RRSIG scaduto)
  - Una zona e le sue sottozone sono BAD quando la signature del *parent* sulla chiave del *child* è a sua volta BAD

Resource Records

- 3 RRs correlati alla cifratura a chiave asimmetrica
  - RRSIG→ signature su RRSet ottenuta usando la chiave privata
  - DNSKEY→ Chiave pubblica, necessaria per verificare un RRSIG
  - DS→ Delegation Signer: puntatore per la costruzione di catene di autenticazione
- 1 RR per la consistenza interna
  - NSEC →Indica quale nome sia il successivo nella zona e quali *typecodes* sono disponibili per il nome corrente

Altre Chiavi nel DNS

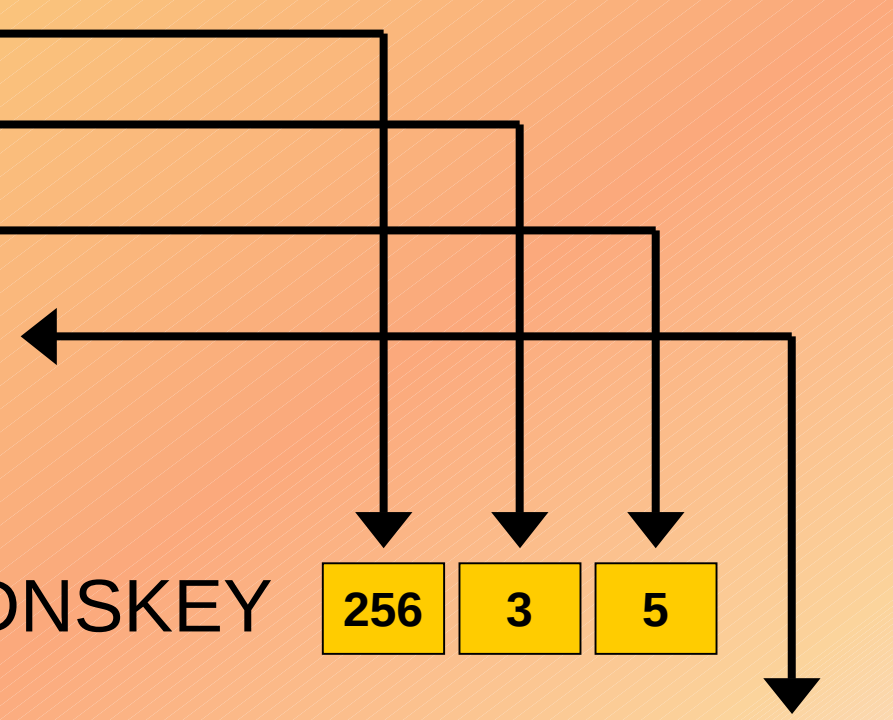
- Il RR DNSKEY dovrebbe essere usato solamente per DNSSEC
  - Chiavi per altre applicazioni dovrebbero usare altri tipi di RRs
- CERT
  - Per i certificati X.509
- Chiavi ancora da definire ufficialmente
  - IPSECKEY
  - SSHFP

RRs e RRSets

- Resource Record
  - Name TTL class type rdata
  - *www.example.net. 7200 IN A 192.168.10.3*
- RRSet: RRs con gli stessi *name*, *class* e *type*
  - *www.example.net. 7200 IN A 192.168.10.3*
    A 10.0.0.3
    A 172.25.215.2
- *Gli RRSets sono firmati, ma non lo sono i singoli RRs che lo compongono*

DNSKEY RDATA

– 16 Bits: FLAGS
– 8 Bits: Protocol
– 8 Bits: Algorithm
– N*32 Bits: Public Key



Esempio:

Example.net. 3600 IN DNSKEY **256** **3** **5**

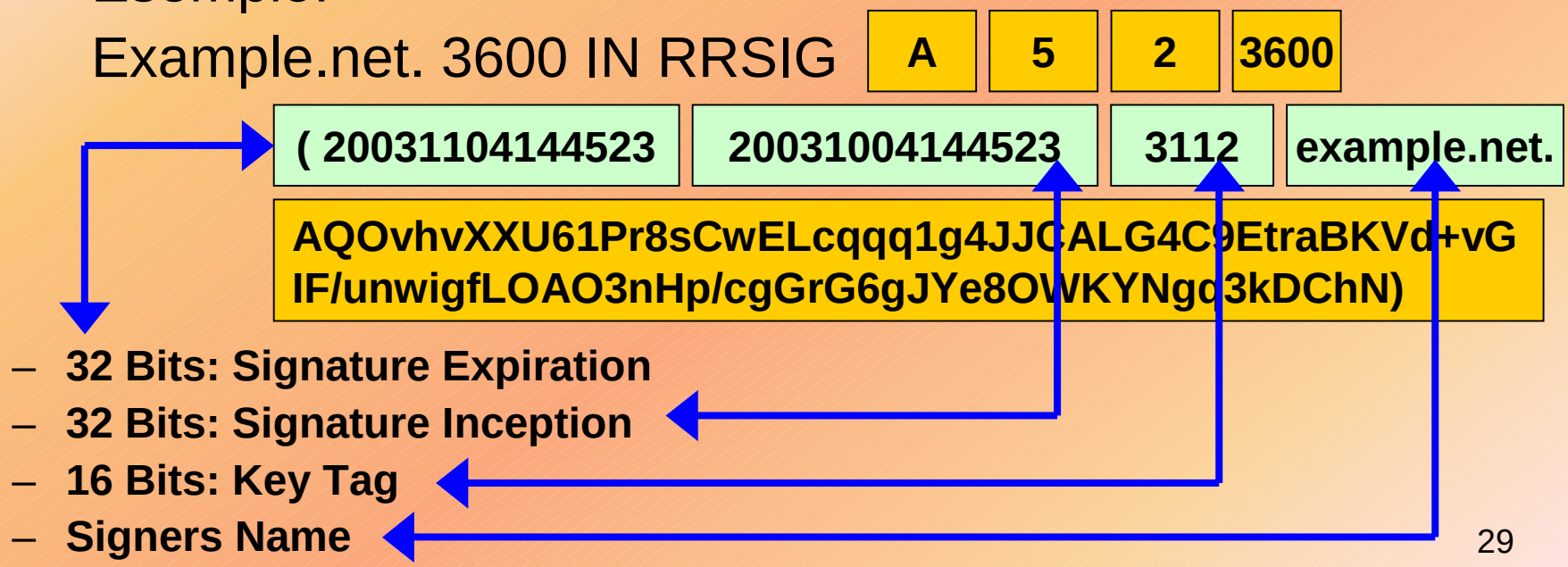
**(AQOvhvXXU61Pr8sCwELcqqq1g4JJCAL
G4C9EtraBKVd+vGIF/unwigfLOAO3nHp/c
gGrG6gJYe8OWKYNgq3kDChN)**

RRSIG RDATA

- **16 Bits: Type covered**
- **8 Bits: Algorithm**
- **8 Bits: nr. Labels covered**
- **32 Bits: Original TTL**

Esempio:

Example.net. 3600 IN RRSIG



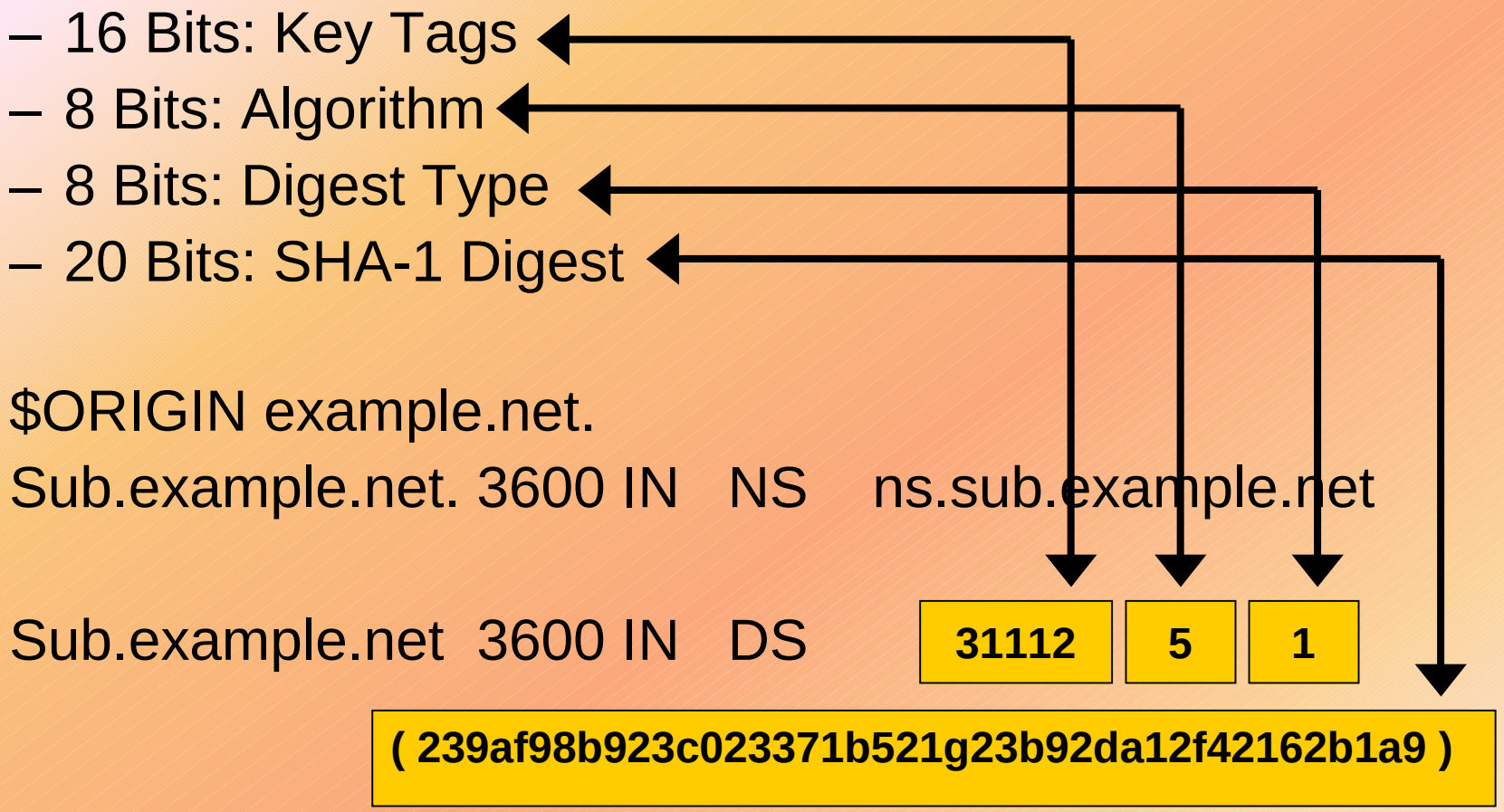
- **32 Bits: Signature Expiration**
- **32 Bits: Signature Inception**
- **16 Bits: Key Tag**
- **Signers Name**

Delegation Signer (DS)

- Il RR Delegation Signer indica che:
  - La zona delegata è firmata digitalmente
  - La chiave specificata è usata per la zona delegata
- Il *parent* è autoritativo per il DS delle zone *child*
  - NON per il record NS che delega la zona *child*
  - Il DS NON dovrebbe essere nelle zone *child*

DS RDATA

- 16 Bits: Key Tags
- 8 Bits: Algorithm
- 8 Bits: Digest Type
- 20 Bits: SHA-1 Digest



NSEC RDATA

- Punta al nome del dominio successivo nella zona
  - Elenca anche quali sono tutti i RRs esistenti per il nome di dominio in questione
  - Il record NSEC dell'ultimo nome si riaggancia al primo nome nella zona (*wrapping around*)
- N*32 bits
- Esempio:
  - *www.example.net. 3600 IN NSEC example.net. A RRSIG NSEC*

Records NSEC

- Se la query Dns richiede dati che non esistono nella zona, il RR NSEC fornisce la prova di *non-esistenza*
- Se dopo una query, la risposta è:
  - NXDOMAIN: uno o più RRs NSEC indicano che il nome non esiste
  - NOERROR e sezione della Risposta vuota: l'array NSEC TYPE prova che il QTYPE non esiste
- Più di un NSEC può essere richiesto nella risposta
  - Wildcards

Sicurezza di una Zona (1)

1. Generazione di una keypair
   - Includere la chiave pubblica (DNSKEY) nel file di zona
2. Firma della zona
   - Inserimento di:
     - Records NSEC
     - Records RRSIG (signature su ogni RRSet)
     - Records DS (opzionali)
   - Generazione del file di key-set

Sicurezza di una Zona (2)

3. Pubblicazione della Zona firmata
4. Configurazione del Forwarding Resolver
5. Test
6. Distribuzione della chiave pubblica (DNSKEY)

Dnssec-keygen

- Dnssec-keygen: strumento per generare le chiavi
  - Dnssec-keygen –a alg –b bits –n type [options] name
- Algorithm: RSASHA-1 (o RSA o DSA)
- Bitsize: dipende dall'algoritmo e dal livello di sicurezza desiderato
- Type: zone
- Name: la zona che si vuole firmare

Creazione delle chiavi

- *$dnssec-keygen – a RSASHA1 –b 1024 –n zone example.net. Kexample.net.+005+20704*
- Vengono creati 2 files
  - Kexample.net.+005+20704.key
    - Contiene la chiave pubblica
    - Dovrebbe andare nel file di zona
  - Kexample.net.+005+20704.key
    - Contiene la chiave privata
    - Dovrebbe rimanere **SEGRETA**!!!

Firma & Pubblicazione

- I Records NS per la zona stessa vengono firmati
- I Records NS per le zone delegate non vengono firmati
  - I Records DS sono firmati
- I Glue Records non vengono firmati

Preparare il file di Zona

- Includere le chiavi pubbliche nel file di zona:
  - Cat Kexample.net.+005+20704.key >> example.net
- Usare il *named-checkzone*
- Incrementare il numero seriale del SOA
  - **Incrementare sempre il seriale SOA prima di firmare!!!**

Firmare la Zona

- *Dnssec-signzone [options] zonefile [ZSK's]*
- Se il nome del file di zona NON è il nome della zona stessa, usare:
  - Usare: -o <origin> option
- Il file di zona firmato viene chiamato "zonefilename.signed"
- Viene creato un Keyset
  - Pronto per andare al *parent*
- Per creare records DS a partire dal keyset:
  - Usare: -g option

Pubblicare la Zona Firmata

- Editare named.conf
  - *Zone “example.net” {*
    type master;
    file “zones/example.net.signed”;
    allow-transfer {10.1.2.3 ;
    key mstr-slave.example.net.; };
    notify yes;
    };
- Usare named-checkconf
- Ricaricare la zona
- Testare

Configurazione del Resolver

- Per verificare il contenuto di una zona
  - Prendere la chiave pubblica (key signing) e controllare che la chiave appartenga al proprietario della zona
- Configurare le chiavi attendibili come sicure nel file named.conf
  - trusted-keys {
    “example.net.” 256 3 1 “AQ…QQ==”;
    };

Testare la Zona sicura (1)

- Dig +dnssec [@server] record [TYPE]
- I Flags di risposta sono importanti
- Query di esempio ad un nameserver autoritativo:

```
; <<>> DiG 9.1.1 <<>> +dnssec @193.0.0.202 www.example.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1947
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 3,
  ADDITIONAL: 4
```

authoritative answer
Not authenticated!

Recursion desired (but not available, RA is not set)

Testare la Zona sicura (2)

```
; <<>> DiG 9.3.0s20020122 <<>> +dnssec @127.0.0.1 example.net NS
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31630
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0,
   ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;example.net.  IN NS

;; ANSWER SECTION:
example.net.          600 IN NS ns1.example.net.
example.net.          600 IN NS ns2.example.net.
example.net.          600 IN SIG NS 1 2 600 20020314134313
                        20020212134313 47783 example.net.
      DVC/ACejHtZylifpS6VSSqLa15xPH6p33HHmr3hC7eE6/QodM6fBi5z3
      fsLhbQuuJ3pCEdi2bu+A0duuQlQMiHPvrkYia4bKmoyyvWHwB3jcyFhW
      lV4YOzX/fgkLUmu8ysGOiD9C0CkSvNSE6rBCzUa3hfkksHt4FBsuA1oQ
      yoc=
```

"use DNSSEC, if you can"

Authenticated Data

Risoluzione Problemi del Client Dns

- Dig riporta uno stato: SERVFAIL
- Provare dapprima senza +dnssec
- Provare anche con +dnssec +cdflag
  - Il controllo viene disabilitato. I dati vengono inoltrati direttamente.

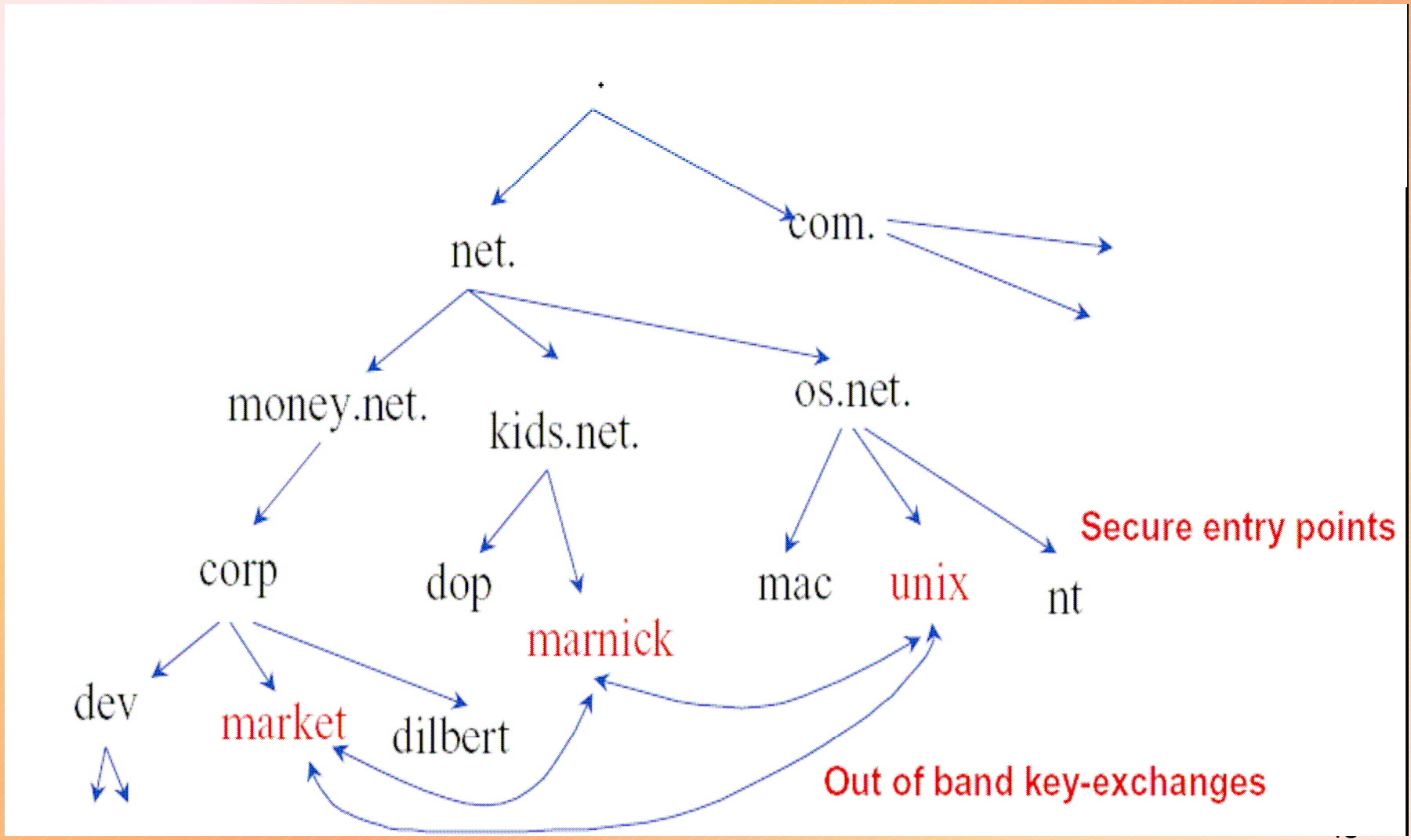
Risoluzione Problemi del Server Dns

- Abilitare il logging. La categoria “dnssec” con livello di debug uguale a 3, fornirà maggiori dettagli sul problema
- L’output del debug è poco dettagliato

Output di Debug: Esempio

```
validating sub.tld KEY: in dsvalidated
validating sub.tld KEY: dsset with trust 7
 validating sub.tld KEY: verify rdataset: success
 validating sub.tld KEY: marking as secure
 validator @0x81b53d0: dns_validator_destroy
 validating b1.sub.tld A: in fetch_callback_validator
 validating b1.sub.tld A: keyset with trust 7
 validating b1.sub.tld A: resuming validate
 validating b1.sub.tld A: verify rdataset: success
 validating b1.sub.tld A: marking as secure
 validator @0x81b9e70: dns_validator_destroy
```

Zone Sicure in Locale



Usare il DNS per distribuire le chiavi

- Le “Isole sicure” rendono la distribuzione delle chiavi problematica
- Distribuzione delle chiavi attraverso il dns:
  - Usare una chiave affidabile per stabilire l’autenticità di tutte le altre chiavi
  - Costruire “catene di fiducia” a partire dal nodo radice
  - I *parents* devono firmare le chiavi dei propri nodi *figlio*
- Mondo ideale:
  - Solamente la *root key* necessita di essere configurata
  - I *parents* delegano sempre la sicurezza ai nodi *figlio*

Records DS per la Delega

- Il *parent* è autoritativo per il record DS
  - NON dovrebbe apparire nel file di zona del nodo *figlio*
- I RRs DS vengono utilizzati per la delega della sicurezza
- DS NON è retrocompatibile con RFC2535
- Facilita la *resigning*
  - Il parent può firmare spesso → vita breve della signature → impatto minore nel caso le chiavi vengano compromesse

Problemi della Chiave

- L’interazione con il *parent* può essere costosa dal punto di vista amministrativo
  - Dovrebbe essere fatto solo quando necessario
  - Le chiavi grandi sono migliori
- Firmare le Zone dovrebbe essere veloce
  - Problemi riguardanti la memoria
  - Problemi riguardanti i fattori spazio e tempo
  - Chiavi più piccole con minore *lifetime* sono migliori

Chiavi: Funzioni

- Chiavi grandi sono più sicure
  - Può essere usata a lungo √
  - Signatures grandi → zonefiles grandi ✗
  - Costoso dal punto di vista computazionale ✗
- Chiavi piccole sono più veloci
  - Signatures piccole √
  - Poco costoso dal punto di vista computazionale √
  - Breve vita (*lifetime*) ✗

Soluzione: Più di una chiave

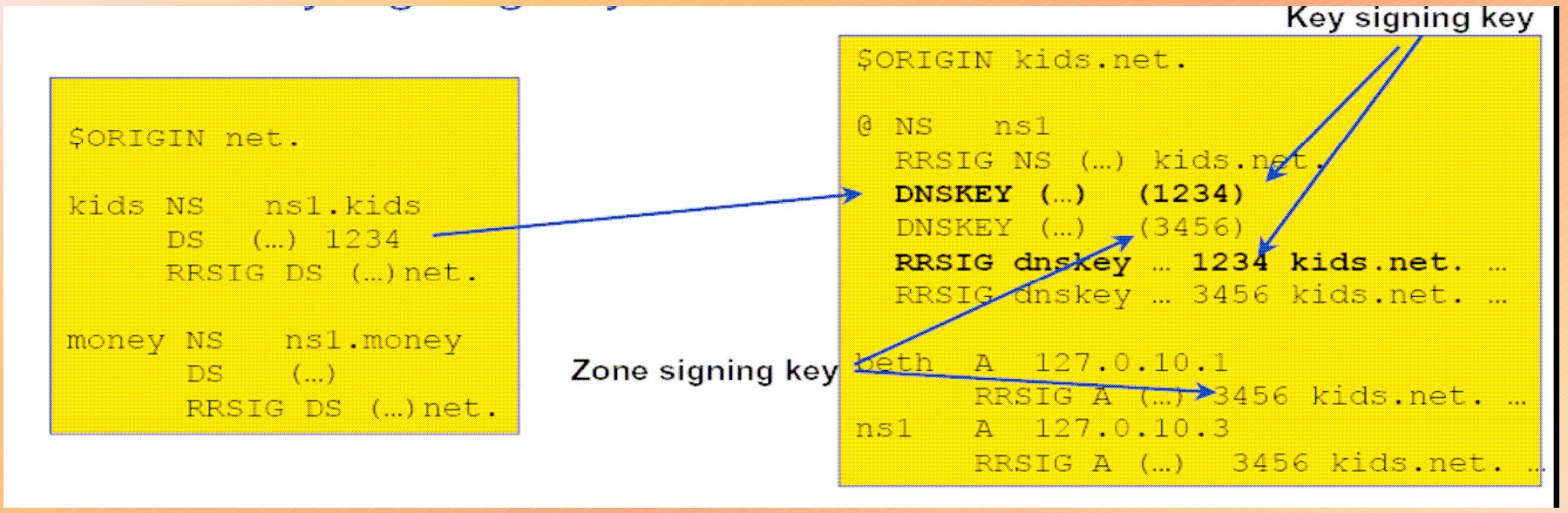
- RRSets sono firmati, ma non lo sono i singoli RRs che li compongono
- DS punta ad una specifica chiave
  - La signature da quella chiave sul DNSKEY del RRSet trasferisce la relazione di trust a tutte le chiavi nel DNSKEY del RRSet
- La chiave a cui punta il DS firma solamente il DNSKEY del RRSet
  - Key Signing Key (KSK)
- Le altre chiavi nel DNSKEY del RRSet firmano l'intera zona
  - Zone Signing Key (ZSK)

Scambio Iniziale della Chiave

- Il nodo *figlio* deve:
  - Inviare al *parent* la chave per firmare il keyset
- Il *parent* deve:
  - Controllare le zone *figlio*
    - Per DNSKEY e RRSIGs
  - Verificare se la chiave può essere affidabile
  - Generare il RR DS

Delega dell'Autorità di Firma

- Il *parent* firma il record DS che punta alla "*key signing key*"



- Il *parent* è autoritativo per il RR DS dei suoi nodi *figlio*

Verifica Catene di Trust: Sommario

- I dati della zona sono considerati affidabili se firmati da una *Zone-Signing-Key*
- Le *Zone-Signing-Keys* sono considerate attendibili se firmate da una *Key-Signing-Key*
- La *Key-Signing-Key* è considerata attendibile se ad essa punta un record DS
- Il record DS è considerato attendibile
  - Se firmato dalla *Zone-Signing-Key* dei *parents*

    oppure
  - I records DS o DNSKEY sono considerati attendibili se scambiati in modalità *out-of-band* e conservati localmente (Secure Entry Point)

Firmare il Zonefile con KSK

- Firmare la zona:
  - Dnssec-signzone –k Kexample.net.+001+20704 → KSK
  - Example.net Kexample.net.+001+16748 → ZSK
- Scegliere ZSK e KSK

Ksk = Key Signing Key

ZSK = Zone Signing Key

Considerazioni sulla Chiave

- Mantenere **segreta** la chiave privata !!!
- La chiave privata può essere rubata
  - Porre le chiavi private su macchine isolate o su *bastion hosts* dietro i firewalls e con robuste politiche di accesso

Cambi di Chiave (Key Rollover)

- Cercare di minimizzare l'impatto
  - Breve validità delle *signatures*
  - Cambi di chiave eseguiti con regolarità
- DNSKEY NON hanno un *timestamp*
  - L'RRSIG sul DNSKEY ha il *timestamp*

Key Rollover (1)

- Rollover della KSK del nodo *figlio*
- Il nodo *figlio* sostituisce la chiave-1 con la chiave-2 e richiede al *parent* di firmarla

```
$ORIGIN net.
Kids NS    ns1.kids
       DS   (…)   1
       RRSIG  DS  (…)  net.
```

Parent Zone

```
$ORIGIN kids.net.
@NS     ns1
     DNSKEY   (…)   (1)
     DNSKEY   (…)   (2)
     DNSKEY   (…)   (5)
     RRSIG      (…) kids.net.     1
     RRSIG      (…) kids.net.     2
     RRSIG      (…) kids.net.     5
Ns1       A      127.0.10.3
      RRSIG  A   (…)   kids.net.    5
```

- A) Creare la chiave-2
- Firmare il KeySet con la chiave-1 e la chiave-2 ed inviare la chiave-2 al *parent*

Key Rollover (2)

- Il *parent* genera e firma il record DS
- Il nodo *figlio* firma la propria zona con la sola chiave-2, una volta che il *parent* abbia aggiornato la propria zona

```
$ORIGIN net.
Kids NS    ns1.kids
       DS    (…)   2
       RRSIG  DS  (…)  net.
```

```
$ORIGIN kids.net.
@NS     ns1
     DNSKEY   (…)   (2)
     DNSKEY   (…)   (5)
     RRSIG      (…) kids.net.     2
     RRSIG      (…) kids.net.     5
Ns1        A      127.0.10.3
      RRSIG  A   (…)   kids.net.    5
```

Key Rollover: Tempi

- Il nodo *figlio* non dovrebbe rimuovere la vecchia chiave mentre ci sono ancora servers che stanno usando il vecchio record DS
- Il nuovo DS dovrà essere distribuito ai servers Slave
  - Tempo massimo impostato nel SOA come *expiration time*
- Il vecchio DS dovrà scadere (expiration) nella cache dei *caching servers*
  - Impostato dal TTL del DS
- Necessario verificare che il master server e lo slave server abbiano ricevuto il cambiamento

TSIG – Dynamic Updates

- E’ possibile usare TSIG o SIG0 per proteggere i *dynamic updates*
- Passi richesti:
  - Configurare la chiave TSIG nel file /etc/dhclient.conf e specificare il FQDN
  - Configurare il file named.conf per consentire agli updates di usare la chiave

TSIG – Dynamic Updates (Client)

- /etc/dhclient.conf

```
send fqdn.fqdn “laptop.example.net.";
send fqdn.encoded on;       # send in dns wire format
send fqdn.server-update off;
                    # tell dhcp server not to update A

key me-friend. {
    algorithm HMAC-MD5;
    secret  "ic…==";
    }

zone example.net. {
    primary 193.0.0.4;
    key me-friend.;
    }
```

TSIG – Dynamic Updates (Server)

- /etc/named.conf:

```
key me-friend. {
     algorithm HMAC-MD5;
     secret "ic…==";
     };

zone “example.net" {
       type master;
       file "zones/example.net.signed";
       notify yes;
       allow-transfer { key tsig.example.net.; };
       update-policy {
       grant  me-friend. name laptop.example.net ;
       };
};
```

Contatti
www.dnssec.net