CHALMERS

Deploying DNS Security Extensions

Master's Thesis within the MPNET programme

SAM ROSTAMPOUR

Department of Computer Science
Division of Networks and Distributed Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden November 2012

Deploying DNS Security Extensions

SAM ROSTAMPOUR

Examiner: Magnus Almgren

Department of Computer Science
Division of Networks and Distributed Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Abstract

The number of attacks towards the Domain Name System (DNS) increases exponentially. Due to the essential role of the DNS in the Internet service, fast reaction is needed to secure this system. Domain Name System Security Extensions (DNSSEC) is an Internet scale solution for protecting the DNS. DNSSEC provides security for the DNS by adding integrity and data original authentication to the DNS messages. As this solution is rolling out fast in the top-level domains (TLDs) and some second-level domains (SLDs) the VOLVO Information Technology (VOLVO IT) company became interested in implementing this solution in their infrastructure. Therefore, they initiated a thesis to become aware of the procedure of deploying DNSSEC and its requirements.

In this thesis, we have gone through the design of the DNS system. We demonstrated how this system is unsecure and why it is the target of many attacks. We explained how DNSSEC protects the DNS and highlighted the objectives of this solution. We also discussed about the current implementation of the DNSSEC. Implementing this solution in a large-scale network is not an easy task and adds lots of complexity to the administrators of the DNS infrastructure. In addition, some security concerns still exist regarding this solution. We have analysed the common attacks and availability issue related to the DNSSEC implementation. We used this analysis to avoid using any implementation, which makes the DNS infrastructure vulnerable to the DNSSEC related attacks. We covered the deployment of the DNSSEC within a particular organization (VOLVO IT). We used the best practice solutions to reduce the security issues regarding the deployment of the DNSSEC.

This thesis is mainly divided in two parts. One part is the security analysis of the DNS and DNSSEC design. The other part deals with the best practice solutions that can be used to set up a secure DNSSEC deployment within the infrastructure of the VOLVO IT.

Acknowledgments

This thesis work has been initiated by the Network Application (Net Apps) team from the VOLVO IT company.

First, I would like to thank my family for their love and consistent support. I would like to thank Magnus Almgren my supervisor from Chalmers University of Technology for bringing new ideas and his motivations. During this project, he spent extended periods of his time for my guidance. I would like to thank my supervisor Joacim Ekroth from the VOLVO IT for his assistance and insights.

Table of Contents

List of Figures

List of Abbreviations

AD	Authenticated Data
CD	Checking Disabled
DS	Delegation Signer
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DLV	DNSSEC Look-aside Validation
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
HSM	Hardware Security Module
ICANN	Internet Corporation for Assigned Names and Numbers
IPsec	Internet Protocol Security
ISP	Internet Service Provider
KSK	Key Signing Key
NS	Name Server
NSEC	Next Secure Resource Record
PKI	Public-key Infrastructure
RR	Resource Record
RRSet	Resource Record Set
SEP	Secure Entry Point
SIG(0)	Request and Transaction Signatures (Asymmetric key cryptography)
SLD	Second-level domain
SSL	Secure Sockets Layer
SOA	Start of Authority
TLD	Top-level domain
TSIG	Transaction SIGnature (Shared secret key cryptography)

TTL	DNS Time to live value, defines the amount of time (in seconds) which a resolved Resource Record can stay in the cache of the resolver
US-CERT	United States Computer Emergency Readiness Team
ZSK	Zone Signing Key

Chapter 1 Introduction

In this chapter, we will determine the scope and the objectives of this master thesis. In the final section, we will go through the organization of this document.

1.1 Background

DNS (Domain Name System) [1],[23] is a fundamental part of Internet and private networks. Many applications and services rely on this system for the name resolution. Lack of data origin authenticity and the absence of the integrity checking mechanisms in the DNS protocol helped the attackers to perform different types of attacks to redirect DNS clients to their malicious domains. DNS Security Extensions (DNSSEC) [20], [21], [27] dealt with these vulnerabilities by adding security parameters to the DNS messages. Adding security parameters would help the clients to verify the authenticity and the integrity of the receiving data by using the included security parameters. As this solution is rolling out fast in the Top Level Domains (TLDs) and in some Second Level Domains (SLDs), the VOLVO Information Technology (VOLVO IT) company became interested to implement this solution in their infrastructure. This thesis was set up by the VOLVO IT company to become aware of the requirements and the procedure of deploying DNSSEC.

1.2 Problem description

This thesis was designed to achieve three main goals:

1. Get a complete understanding of the DNSSEC and its related modifications to the DNS protocol.

2. Determining requirements for deploying the DNSSEC within the infrastructure of the VOLVO IT.

3. Investigating for the best practice solution for deploying the DNSSEC.

The DNSSEC protocol faced several iterations to be improved for the DNS security. These iterations were due to the unsuccessful implementation of the DNSSEC in the earlier lifetime of this protocol. However, the current implementation of the DNSSEC also suffers from some security issues and introduces new forms of attacks. In addition, enabling DNSSEC in the large-scale network is a complex task and will bring new challenges in front of the DNS administrators. Beside the defined goals, which are explained above, the security analysis should be conducted through this thesis. We will use this analysis to show that the DNSSEC, in general, and our solution reach the security objectives, which we are aiming to achieve.

1.3 Report organization

This chapter defines the scope of this thesis. The second chapter covers the design and the structure of the DNS. Chapter 3 presents the DNS security vulnerabilities and explains in brief how DNSSEC can protect DNS against some of these security threats. In the fourth chapter, we will go through the design and the implementation of the DNSSEC. In chapter 5, we will discuss the security and availability issues related to the DNSSEC implementation. In chapter 6, we will cover the deployment of the DNSSEC considering the best practice solutions within the infrastructure of the VOLVO IT company. In chapter 7, we will discuss about the delimitations of our solution. Chapter 8 contains the summary of this report.

Chapter 2 DNS

DNS (Domain Name System) [1],[23] plays an important role in the Internet and private networks. Many applications and services rely on this system for name resolution. Name resolution, returning the IP address of a host name, is the main role of the domain name service. In this chapter, we will go through the DNS design, its architecture and define some basic terminologies related to the DNS. We will also explain how this system works.

2.1 DNS topology

The DNS has a structure like a tree. The database of this system is distributed among the nodes of this tree. Each node of this tree is called a zone. DNS zone may contain one or more domains in the DNS tree [2]. For simplicity, in this report each zone will contain only one domain. In other words, we will use the zone and domain words interchangeably.

Each zone in the DNS hierarchy is owned by an authority and managed by one or many organizations. The zone is authoritative over all of its data (but with one exception see section 2.3). Database of each zone is stored in the servers, which are known as the authoritative name servers. In the DNS, the authorities are able to delegate the authority to other organizations.

The beginning node of the DNS tree is the “root” zone. The official root zone is managed by the Internet Corporation for Assigned Names and Numbers (ICANN) [9].

Under the root domain Top Level Domains (TLDs) exist, that root zone delegated the authority to them. These domains are grouped in to two types:

1- Generic TLDs (gTLD): For example, “.com”,”.net”,”.org” and etc.

2- Country Code TLDs (ccTLDs): For example, “.se”,”.us”,”.ca” and etc.

The administrations of the ccTLDs are delegated to each country. The name servers of gTLDs and ccTLDs are under the supervision of different organizations, which are called registries.

In the DNS hierarchy, under each TLD zone other zones exist which are called Second Level Domains (SLDs) and the TLD delegate the authority to them. The SLDs are owned by different organizations that have been registered their domain names to the registrars (see section 2.1.1). For example “volvo.com” is owned by the VOLVO Group company. Other domains can also exist under the SLD and DNS tree can become more extended within the organization.

2.1.1 Registrar

For ease of the administration of the DNS tree, ICANN and ccTLDs assigned the responsibility of the registration of the domain names to the registrars [68]. The registrars are different organization from the registries and provide services to the public. Registrars are responsible for providing the registries with the information of the registered domain name and the IP address of the authoritative name servers for that domain [57].

2.2 Domain name space

Domain names are in the Fully Qualified Domain Name (FQDN) format. In this format, each domain name consists of two or more labels that are separated by dots. The beginning node of the DNS tree is the “root” domain that is represented with the “.” at the end of the domain name. The right most label of the domain name is the TLD name (gTLD or ccTLD) and the second label is the SLD name. The domain name can also be followed by other labels, which are the name of the other subdomains.

For example, in the “subdomain.volvo.com.”:

“.com” is the gTLD name,”.volvo” is the SLD name and the “subdomain” is the name of the third level domain.

2.3 DNS data

DNS data is stored in each domain in a file called “zone file”. A zone file is created from a special data structure that is called Resource Record (RR).

2.3.1 Resource Record (RR)

RR is the core data structure of the zone file, DNS queries and answers. The format of the RR is illustrated in the Figure 2.1.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
/                                               /
/                      NAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.1 Format of the RR[1].

The RR consists of the following fields:

NAME: This field contains the owner name of the RR.

TYPE: This field contains the Type of the RR.

CLASS: This field contains the protocol family (it always contains IN value).

TTL: This field contains a 32 bit signed integer specifying the time interval, which this RR is going to be cached in the resolvers.

RDLENTH: This field contains an unsigned 16 bit integer that specifies the length of the RDATA field.

RDATA: This field contains a variable length data that defines the resource of the RR.

Each RR has three properties, which are Name, Type and Class. For example the IP address of the webserver of the X company is stored in an A RR with the owner name of "www.x.com". It has the "IN" (Internet) class and A (IPv4 address) type. The RDATA in this example will be "203.0.113.35" which is the IP address of the corresponding webserver.

RR has different types. The most common RR types are as follows:

SOA: This RR identifies the start of authority. This RR stores the information of the zone, the default TTL for the names in the zone etc.

A: This RR identifies the host address in the IPV4 format.

AAAA: This RR identifies the host address in the IPv6 format.

NS (Name Server): This RR identifies the name of an authoritative name server.

PTR: This RR identifies the domain name pointer.

CNAME(Canonical Name): This RR identifies the alias name for a host.

MX (Mail Exchange): This RR identifies the name of the mail exchange server.

SRV: This RR identifies the name of the available services in the zone ex. LDAP, HTTP etc.

TXT: This RR is used for storing text string. This RR is used for storing the information and carries the human-readable text in a DNS RR.

2.3.2 Resource Record Set (RRSet)

Each RR has its own specific Name, Type and Class. All RRs that have the same Name, Class and Type but have different RDATAs are referred to as a Resource Record Set (RRSet). When a query is sent from a resolver to an authoritative DNS server, this query is sent as a tuple (QNAME, QTYPE, QCLASS). The format of the question section in the DNS query is shown in the Figure 2.2.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
/                                               /
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.2 Format of the question section [1].

The answer of the query will be the RRSet that contains all the RRs with the same tuples and different RDATAs. For example when a resolver queries for the "volvo.com", specifies NS in the name resolution command, it will query for the RRSet and not for an individual RR [5]. Then it will receive all the NS RRsets of the authoritative name servers for the "volvo.com".

2.4 Name resolution

Before we explain the process of the name resolution, it is important to become familiar with the concept of the referral. Each authoritative name server is responsible to send an authoritative answer for the RR that is resided in its zone (not all the RRs). When a resolver sends a name resolution request to an authoritative name server, if that name is authoritative for that zone, name server will send the answer. However, if the name belongs to the lower zones other information should be sent to the resolver. The Name Server RRSet (NS RRSet) and A RRSet of the child zone are stored at the parent zone at the zone cut. These records are non-authoritative RRSets. NS RRSet contains the names of the authoritative name servers of the child zone and the A RRSet contains the IP addresses of these servers. The reason that we call these RRs non-authoritative is that these RRs belong to the child zone but they are resided at the parent zone. In the query process in order to help the resolver to find the location of the authoritative servers of the child zone these records are sent as the referral answer to the resolver.

The process of resolving an IP address to a host name is called "Name resolution" or recursion [2]. This process is illustrated in the Figure 2.3.

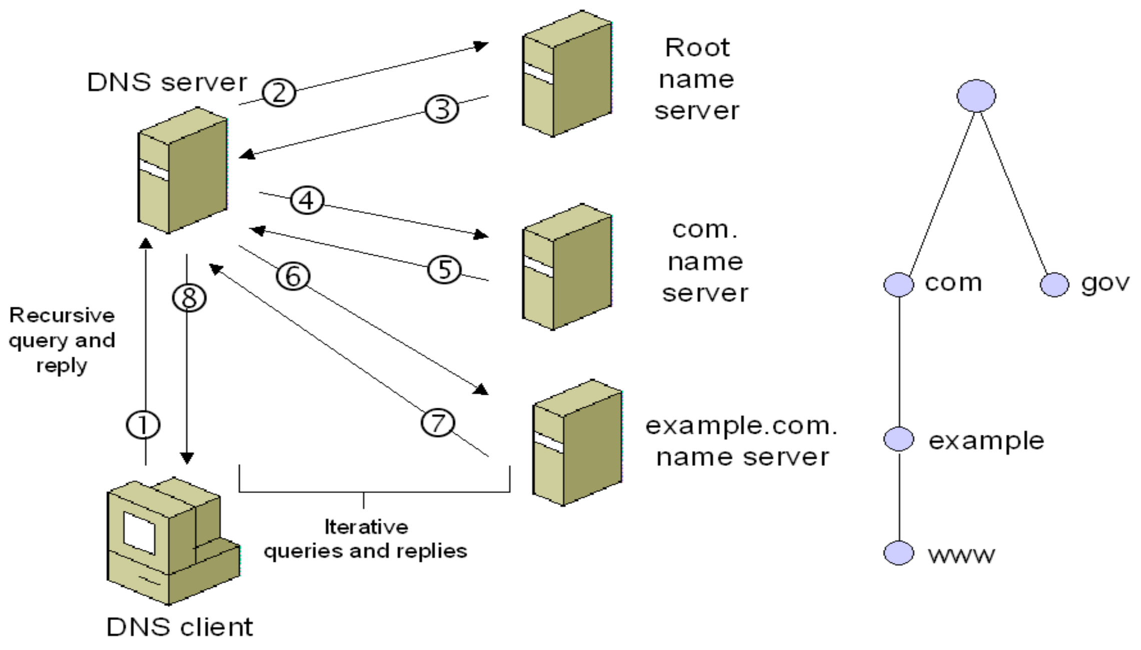


Figure 2.3   Name resolution sequence for the query of the "www.example.com" [69].

Step 1: When a name resolution request is sent from the client, it is first sent to the local resolver as a recursive query. In the recursive query, the client is not referred to another name server and the final result is sent back to it.

If this name, and other domain names in the DNS tree ex. ".com", has not been resolved before, the iterative queries will be started in the resolver from the root servers.

Steps 2 and 3: The query is sent from the resolver to one of the authoritative name servers of the root zone. When a root server receives a query, for "www.example.com", it will answer with the referral of the ".com" zone. The "www.example.com" is not an authoritative A RR for the root zone but ".com" zone is the child of the root.

Steps 4 and 5: When the resolver finds the location of the authoritative name server of the ".com" zone, it will send another iterative query to the authoritative name server of the ".com" zone. The authoritative server of the ".com" zone returns the referral to the resolver, which contains the name and the address of the authoritative name server of the "example.com" zone.

Steps 6 and 7: The resolver performs another iterative query to the authoritative name server of the "example.com" zone. At this step, it receives the A RR that contains the IP address of the "www.example.com" webserver.

Step 8: The results of the recursive query is sent back to the client.

The result of each iterative query is stored in the cache of the local resolver until their TTL has expired. The cached data can then be used by the resolver to answer later queries from the client.

Chapter 3 DNS security analysis

During the implementation of the naming services, there was no consideration for the security. In this chapter, we will try to demonstrate the existing security vulnerabilities in the DNS protocol.

3.1 DNS security issues

For more than a decade, many discussions took place on how to authenticate parties, who are involved in the DNS query process. It was difficult to define a comprehensive model to apply security to the DNS while this service has become enormously extended. In addition, it has become evident that deploying security will add a lot of complexity in the DNS administration [10].

Both Internet and private networks users use DNS service for mapping FQDN names to IP addresses. When a resolver, it can be a cache-only DNS server or a stub residing in the client device, make a request from an authoritative DNS server for name resolution, no authentication takes place in between. During the query, any unauthorized party can compromise the cache of the resolver by injecting the wrong information. Afterwards, that party would be able to direct the clients to its desired locations. Another security hole in the DNS design is the absence of an integrity checking mechanism for receiving data. One example for this vulnerability is when the attacker is eavesdropping on the communication line between requestor and responder and tries to send modified data to requestor on behalf of the responder [11]. DNSSEC has been purposed to countermeasure these vulnerabilities.

Observing an increasing number of attacks to the DNS and considering the vital role of this service in Internet, urged organizations who were involved in development of DNS to come into an agreement to define a standard protocol for securing DNS. This protocol is called Domain Name System Security Extensions (DNSSEC). It is important to mention that DNSSEC is not a complete cure for the DNS security issues. This solution has its own weaknesses [12] but it provides enhancement for the DNS security and decreases the possibility of performing attacks to this service. In the following sections, based on the vulnerabilities mentioned above, we will show existing threats towards the DNS. We will also describe in brief how DNSSEC protects the DNS.

3.1.1 Packet interception

When a resolver starts to ask about the location of the specific domain from a name server, it sends the query packets through the network. This network is unsecure and most of times resolvers use the UDP protocol. One reason for using the UDP packet is that the resolvers in the internal network reside behind the firewall and do not use the TCP connection. During the query process, third party can eavesdrop on the exchanged information from the communication line between requestor and responder. This third party can make copies, delete or modify packets. This is called packet interception. Man-in-the-middle (MITM) attack uses the packet interception.

Man-in-the-middle attack or Process-in-the-middle attack

In this attack, the attacker listens to the requests from the resolver, which is looking for an authoritative name server for a specific domain. The essential requirement for this attack is that the attacker and the resolver must be in the shared network. When the attacker becomes aware of the query process it tries to send an answer to the resolver before the resolver receives any response from the real authoritative name server. The answer packet contains the IP address of the faked name server, which is bound to FQDN name of the requested domain. When resolver receives the answer it will bind the queried domain name to the spoofed IP address in its cache. Now attacker will be able to provide the victim with fake information [13]. The attack is shown in Figures 3.1, 3.2, 3.3:

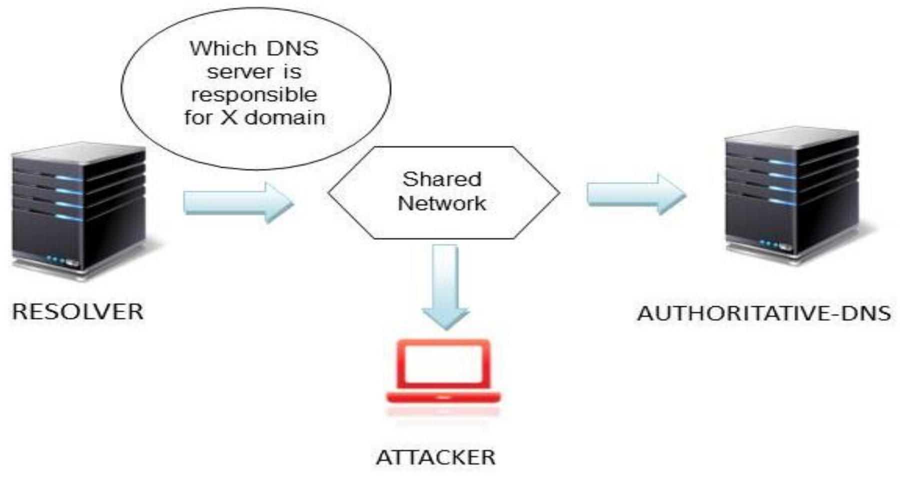


Figure 3.1 MITM attack –Resolver asks for the DNS server's IP which belongs to the specified domain.

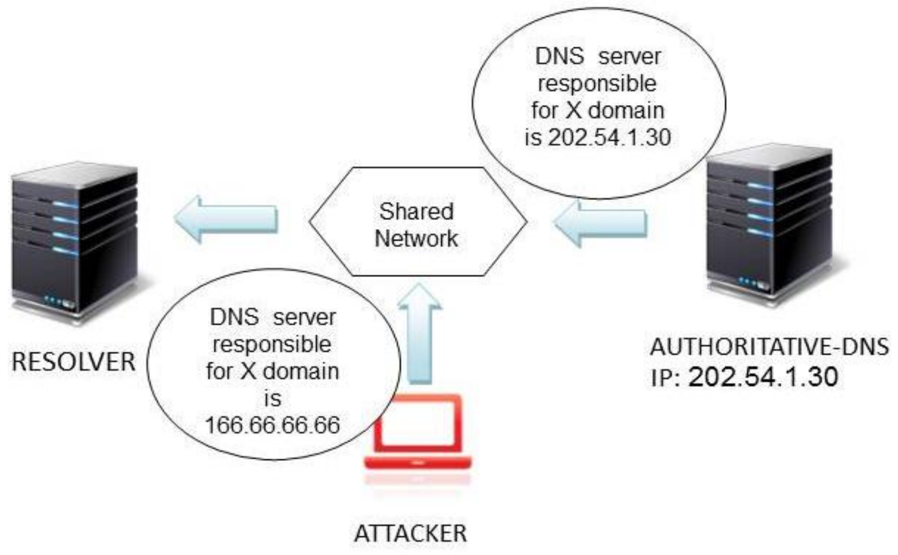

DNS server responsible for X domain is 202.54.1.30
Shared Network
DNS server responsible for X domain is 166.66.66.66
RESOLVER
AUTHORITATIVE-DNS
IP: 202.54.1.30
ATTACKER

Figure 3.2 MITM attack –Two responses but the attacker response is received first.

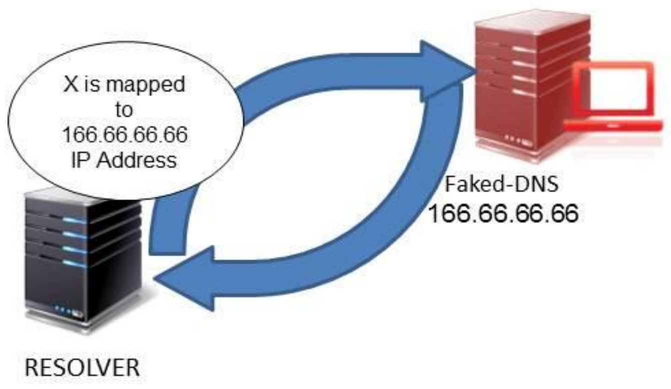


Figure 3.3 MITM attack –Faked DNS become responsible in later queries for the specified domain.

In the MITM attack, the attacker makes sure that its response is received first, by performing a denial of service (DoS) attack to slow down the original authoritative DNS server performance [14].

The reason why the attacker easily can perform this attack is due to the usual behaviour of authoritative DNS servers. An authoritative DNS server sends answers to the resolver without signing or encrypting the packet contents and as a result, the resolver cannot determine whether this information has been forged or if it is from the real DNS server [15].

Countermeasures: Some countermeasures have been proposed for this attack, like using channel security mechanism. Channel security can be implemented by different means namely, TSIG or IPsec. One problem with these solutions is that they require lots of processing resources. In special cases, like DNS service, large amount of requests and responses are generated in a small portion of time so the demand for processing, due to searching through database, creating answer packets and encrypting or signing messages, becomes higher [15]. In root or gTLDs domain servers, this demand is much higher. Another problem with these types of solutions is the cost of establishing and maintaining two-way trust relation between entities of the DNS hierarchy. In the DNS query process there is no need to have the permanent secure channel between entities. Another problem is that these countermeasures use hop-by-hop integrity checking mechanism.

DNSSEC: We need to have some sort of end-to-end data integrity checking mechanism, which is performed at the end devices. DNSSEC provides an end-to-end data integrity checking mechanism. In addition, it has better performance comparing to the other countermeasures [15].

3.1.2 ID guessing and query prediction

Sometimes attackers establish the DNS attack by guessing the transaction ID of the DNS query of the resolver, which is trying to connect to an authoritative name server. In this attack, the attacker comes and tries to impersonate itself as the original DNS server by guessing the transaction ID and UDP port-address used in the communication. The attacker also needs to predict the behaviour of the resolver. This action is referred to as "Query Prediction" [15]. This attack is different from the previous one because in the MITM, the attacker had a complete control on the communication channel but in here, the attacker tries to steal the communication by guessing the parameters in the name resolution request. This attack is harder for the attacker to perform than MITM but not impossible because the attacker can guess the required parameters in the UDP and DNS headers by doing brute force searching. The UDP port-address is a 16-bit field and the DNS ID field in the DNS header is also 16 bits, these parameters are generated for every pair of request and response so that the resolver and the authoritative DNS server can match them together [1]. To make guessing harder for the attacker these two parameters are generated randomly but these parameters have a short length. These two fields together are 32 bits and it takes a small time for the attacker to try 2^{32} different combinations of 1s and 0s. If the victims are using fixed port-number attacker only needs to try 2^{16} combinations. In addition, this process becomes easier by analysing the previous traffic [15].

However, as we mentioned, in addition to guessing these two parameters the attacker needs to predict the behaviour of the resolver, by guessing the QNAME and QTYPE of the RR that resolver might be querying. The only thing that the attacker needs to do is to find the state of the resolver ex. attacker can follow the state of the resolver from the time the resolver is rebooted [15].

Countermeasures: We can use the same solution, like in the Packet Interception attack, using TSIG or IPsec to make sure that the integrity of the answers are not compromised. However, these solutions do not provide an end-to-end integrity checking mechanism.

It is important to note that we can use IPsec and TSIG in DNS infrastructure beside DNSSEC. In the last hop communications, client to resolver, we can use IPsec and in the Zone Transfer TSIG can be used, which is a security mechanism based on a pre-shared key, to enhance the security.

DNSSEC: If the DNS server signs the answers by its private key then the resolver will be able to detect the forged answer by verifying the signature using the related public key.

3.1.3 Cache-poisoning

As it was mentioned before, RRs are the core data structures of the DNS and they map the host names to the IP addresses. When the resolver is referred to another name server, by the referrals, the IP address and the host name that belongs to a specific name server can be forged. This forged name or IP address can be used as a hook by the attacker. This enables the attacker to inject malicious data into the cache of the victim [15]. Name chaining attack is one type of "Cache-poisoning" attack [18] where the attacker tries to redirect DNS users to his desired name servers. All the name chaining attacks have the same property, which is manipulating the referrals.

In the "Cache-poisoning", in the majority of the scenarios the attacker manipulates information related to NS, CNAMEs, DNAMEs and in some cases the MXs (Mail servers) and SRVs (defines the location of a host for a specific service) [15]. In this way, the attacker can redirect users to its desired servers and prepare them for other attacks.

The following steps describe a query process, real world scenario, where the Cache-poisoning attack is established [19].

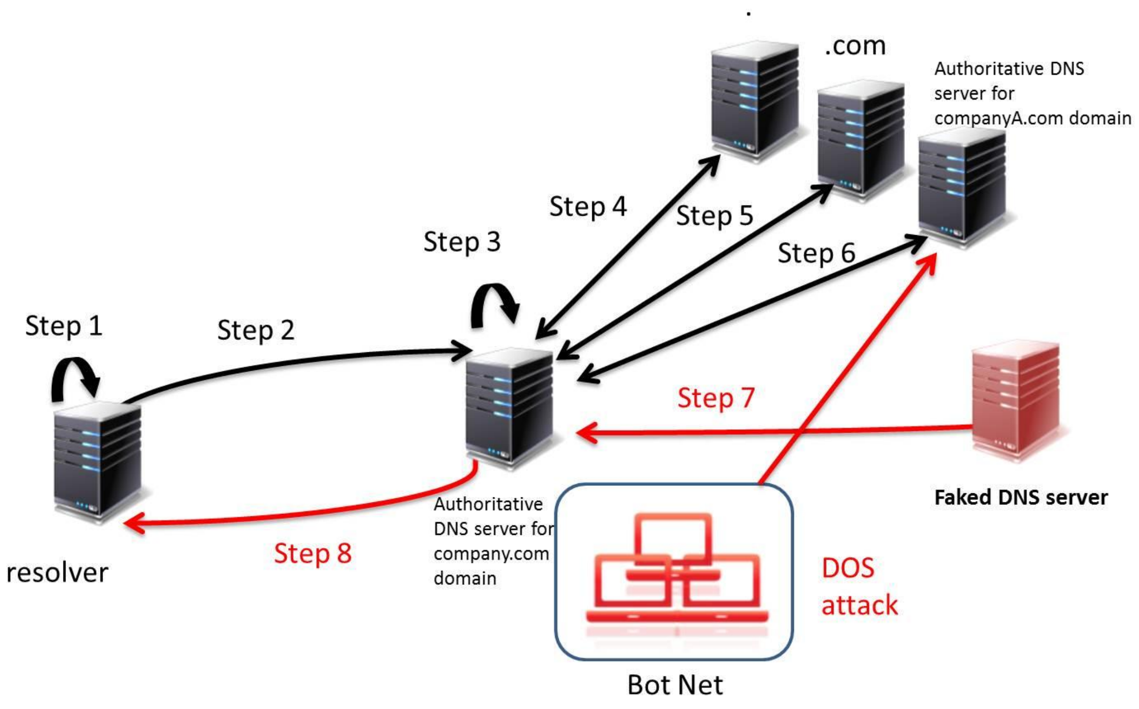

.
.com
Authoritative DNS server for companyA.com domain
Step 4
Step 5
Step 3
Step 6
Step 1
Step 2
Step 7
Faked DNS server
Authoritative DNS server for company.com domain
Step 8
resolver
DOS attack
Bot Net

Figure 3.4 Cache-poisoning attack [19].

We follow the attack steps from Figure 3.4:

1- When a resolver faces a name resolution request, for example a request for "webpage.companyA.com", it first checks the cache to find an entry related to that domain name to return the related IP address.

2- If no such entry exists, it will send a name resolution request to the authoritative DNS server of the internal network, here "company.com" name server.

3- In this step, the authoritative DNS server searches through its database to find a match for the requested domain name. In this case, requested domain name does not belong to "company.com" so no match is found, as a result, it checks in its cache to see if this name has been resolved before, from previous iterative queries, but there is no such entry in its cache.

4- The internal authoritative DNS server starts the iterative query process, asking the IP address of the ".com" gTLD server from the servers of the root zone, the root server returns the answer, which contains the IP address of ".com" name server.

5- Then the DNS server asks ".com" server about the location of "companyA.com". The IP address of companyA.com server is returned as an answer.

6- In this step, when the authoritative DNS ask the IP address of the "webpage.companyA.com" from "companyA.com" domain server, the attacker involves itself in this process.

7- Attacker tries to establish its attack using query prediction and ID guessing in order to inject forged information to the cache of the DNS server of the "company.com". In here as you can see from the picture the attacker uses its Bot-Net resided in any place of the Internet, to perform Distributed Denial of Service (DDoS) attack to delay "companyA.com" server from responding to the issued query and wins some time to guess the next DNS transaction ID and the port number of the request of the resolver. In the response from the attacker wrong referrals (NS RRSet and A RRSet of the malicious server) are included which poisons the cache of the internal authoritative name server.

8- This information is returned back to the local resolver. These fake records will redirect clients to the attacker's name server.

   From now, it all depends on the attacker's intent whether he wants to prepare clients for another attacks like Phishing attack to get important information from the users or poison the name server with another malicious RRs.

DNSSEC: This attack is a complex attack and uses different security vulnerabilities from the DNS design. By deploying DNSSEC, the resolver would be able to check signatures for each RRSet and find out if the information associated with each name in the answer packets were inserted by the corresponding authoritative DNS server or by someone else. Chain of trust is another valuable feature of DNSSEC [12]. This feature establishes trust relations between root and top-level nodes, between top-level nodes and second level nodes and continues this relation to the end nodes in the DNSSEC hierarchy. Only servers, which are in the DNSSEC trust hierarchy, will be able to send legitimate answers to the requestor, consequently the attacker cannot get involved in the query process.

3.1.4 Betrayal by the trusted server

How can we make sure the DNS server, which we are getting information from, is trustworthy? In the current DNS architecture when a client requests a name resolution, the name servers perform the query on behalf of the client and the client trusts them. In the DNS architecture, there is no list of the trustworthy servers and non-trustworthy ones. By default the client sends its query to any servers, which have been configured for it ex. DHCP assigned name servers. When the client gets connected to the ISP some parameters like IP address and DNS servers are assigned automatically to the client and it uses these DNS servers without knowing that these servers are trusted or not. If the client is not lucky, the malicious DNS server is assigned to it and it will provide the client with malicious information [15]. This is a well-known security issue when clients are portable and try to connect to the Internet service providers during their travels.

Countermeasure: Solutions for this attack is somehow similar to the solution for the packet interception. Both sides should authenticate each other to be able to start to exchange DNS packets. This authentication can be in form of using a pre-shared key like in TSIG or using other authentication methods. However, TSIG does not guaranty that the name server is trustworthy. TSIG can only secure the communication channel between the client and the recursive DNS server, the client made the assumption that the server is trusted [15].

DNSSEC: In the DNSSEC design, a trust relation is established using the "Chain of Trust" [30] (It is explained in detail in chapter 4). To establish the trust relation between two nodes, there should be a fully trusted point in the network that all nodes can rely on. In the DNS hierarchy, resolvers choose root as a fully trusted point. Resolvers need to keep root's public key and they are required to authenticate the responses from the root and follow the chain of trust to their addressed name server [12].

The resolver uses the public key to authenticate the answers by checking the signatures appended to the RRSets in behalf of the client, if the client does not trust the resolver server for name resolution it should do the entire signature checking. We need to consider if the client wants to check the authenticity and integrity, DNSSEC should also be deployed to the client's device. Clients also need to keep public keys individually [15].

3.1.5 Denial of service attack (DoS attack)

By the definition from US-CERT, a denial of service attack is a type of attack, where the attacker tries to prevent users to access different type of services [67].

Here the service is DNS and any attempt that inhibits users from accessing information or using services through the DNS hierarchy is called a DNS DoS attack. Because DNS is an extended service, DoS attacks against this service are also extended.
DoS attacks have different forms and everyday new forms of these attacks appear. Distributed DoS (DDoS) attack is one type of the DoS attacks. In this attack, attacker uses Bot-Net to magnify the result of the attack. One example of the DDoS is "Amplification" attack, which will be discussed later (see section 5.4).
In the paper from Steven Cheung [22], he proposed a tree model for this type of attack, as you can see from Figure 3.5 that root of this tree is the main goal of the attacker and each node in this tree shows the way the attacker accomplishes his goals.

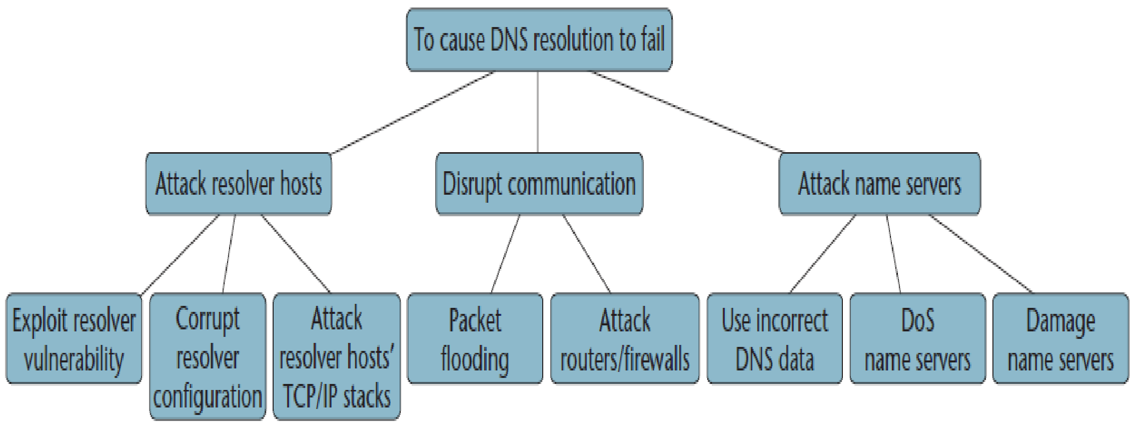


Figure 3.5 DNS DoS attack tree model [22]

Based on the [15], DNSSEC would not provide any protection against these attacks and may make new types of DNS DoS attacks possible. DNSSEC messages are larger than ordinary DNS messages, which require more processing resources. In addition, the resolver is required to do additional processing for signatures checking. As a result, the attacker can make use of this situation and issue many DNSSEC responses to make the resolver busy all the time checking signatures of these messages, so it would not be able to respond to legitimate users.

3.1.6 Other threats

Security vulnerabilities of the DNS protocol are more than the vulnerabilities, which were mentioned above. We will explain these vulnerabilities in brief in this section while DNSSEC does not cover them. DNS vulnerabilities are as follows.

Unsecure zone transfer: In the DNS specification [23], [1], Zone transfer is a mechanism, which is used between primary and secondary DNS servers to allow secondary servers to have a redundant copy of the master’s database. They synchronize any change that may occur in records with the primary server. During the zone transfer from one DNS server to another, there is always a risk that this information, which is not encrypted, is eavesdropped by unprivileged users. Zone transfer is one of the known DNS breaches, countermeasures are: deploying physical security, using security features in the DNS servers like TSIG that encrypts the contents with its pre-shared key [15], using VPN connections etc.

Unsecure dynamic updates: Dynamic update is one of the functionalities of the DNS server [24], which is the process of updating RRs like adding, modifying or deleting RRs in the primary DNS server database based on updates received from secondary DNS servers. To maintain consistency the primary server is first updated then updates are sent to the secondary DNS servers. It is also used to update zone’s IP address information based on updates received from DHCP server. This process also suffers from the same kind of vulnerabilities that we saw in the zone transfer. To secure Dynamic updates two methods are suggested by IETF: Using TSIG or SIG(0) [25] for authentication and data integrity. TSIG has key distribution problems in large-scale networks, due to the use of a pre-shared key, and cannot scale very well [15] so it is suitable for small-scale environments. In SIG(0) Public Key infrastructure (PKI) is used which has more advantages over pre-shared key. In PKI, we have no problem for distribution of the public keys because everyone can read them.

DNS server security issues: All the DNS servers namely BIND, Windows DNS servers suffer from some security issues. Attackers always try to find the security holes in these devices to perform their attacks. The best countermeasure for this vulnerability is that administrators educate themselves about weaknesses of servers, which they are working with, to avoid any implementation that leads to a security hole in the network. Updating servers with the trusted patches is another good practice.

Chapter 4 DNSSEC

Lack of the data origin authenticity and absence of the integrity checking mechanisms in the DNS protocol helped the attackers to perform their attacks. DNS security extensions (DNSSEC) [20], [21], [27] dealt with these vulnerabilities by adding security parameters to the DNS's transaction messages. Adding security parameters helps the receiver to verify the authenticity and the integrity of the receiving data by verifying the included security parameters. In this chapter, we will describe how DNSSEC modifies the DNS dataset and protocol to provide security enhancements.

4.1 DNSSEC history

The DNS protocol was designed in the 1980s. At that time, there was no security consideration in mind. In 1995 a paper was published by Steven M. Bellovin, "Using the Domain Name System for System Break-Ins" [16], which described a serious vulnerability in the DNS design that helped the attackers to perform the Cache-poisoning. From that time, applying the security to the DNS became an important concept. IETF DNSEXT (DNS Extensions) working group became responsible for updating the design of the DNS and updating the RFCs related to the new design of the DNS. In 1997, the first attempt for standardizing DNS security took place. This standard was described in 2065 RFC, which was using digital signatures to add security to DNS. In 1999, this RFC was replaced with RFC 2535 and become the standard for implementing DNSSEC. However when this standard was implemented it had many operational problems so it was decided to modify this standard. In 2001, new drafts were defined. These drafts described the DNSSEC protocol design and new RRs. During 2002 to 2003, these drafts were refined. The NLnet labs start to do an experiment to implement 2535bis [3]. In 2004, it was decided that the drafts are ready to be published. In 2005 new RFCs (4033, 4034, 4035) [20], [21], [27] were published and became the standard for DNSSEC.

In October 2005 SEWEDEN domain (.SE) signed their zone and become the first ccTLD which deployed DNSSEC [3]. From 2005 until now many drafts, standard and informational RFCs are defined to remove the security issues and operational complexity regarding the deployment of the DNSSEC standard (ex. standardizing the NSEC3 in 2008 due to zone enumeration in the previous DNSSEC design and key timing consideration and etc.). In 2010, the root zone was signed [31]. Now deploying DNSSEC is rolling out fast and many TLD domains have signed their zones.

4.2 New Terms

When entities of the DNS infrastructure become aware of DNSSEC, new terms are used, which are as follows.

Security-aware name server: This is an authoritative name server in the DNS infrastructure, where the DNSSEC is deployed. This server understands new RRs definitions and can perform the zone signing. It also knows how to provide security-aware resolvers with authenticated RRs in the query responses [27].

Security-aware resolver: This is a server in the DNS infrastructure, which sends queries to the security-aware name server and asks for authorized RRSets. It can be "validating", when it validates RRSets by verifying signatures that are included in the response message, or it can be “non-validating”, when it trusts the security-aware recursive name server. The non-validating security-aware resolver requests the recursive name server to do this verification on its behalf [27].

Security-aware recursive name server: This is a server in the DNS infrastructure, which can act as a security-aware name server and a security-aware resolver at the same time [27].

Security-aware stub resolver: The stub resolver is a module that performs DNS name resolution functions using recursive queries and it is located at the client machine [1]. The security-aware stub resolver understands the DNSSEC protocol and its dataset modifications. The security-aware stub resolver can be “validating” or “non-validating” [27].

Authentication Key: The security-aware name server signs RRSets with its private key, then security-aware resolver uses the related public key as an “Authentication Key” to authenticate RRSets in the response message [27].

4.3 What does DNSSEC provide?

DNSSEC provides three features for the DNS protocol:

1. Data origin authentication
2. Data integrity
3. Authenticated denial of existence

These features are provided by performing some modifications to the existing DNS protocol and dataset [27].

4.3.1 Data origin authentication and integrity

In a DNSSEC aware environment, the requestor validates the authenticity and integrity of the receiving data by checking the digital signatures. These signatures are generated using the asymmetric cryptographic keys at the responder side [20].

Before sending the response to the requestor, the responder uses the private key [28], to create a signature for each RRSet [27] in the zone. On the other side, the requestor requests the name resolution and asks for the secured RRSet [27]. Then the requestor validates the signatures over the RRSet in the response message [21] and caches the receiving information. For this purpose, DNSSEC needs to add some changes to the existing DNS protocol. This process is illustrated in the Figure 4.1.

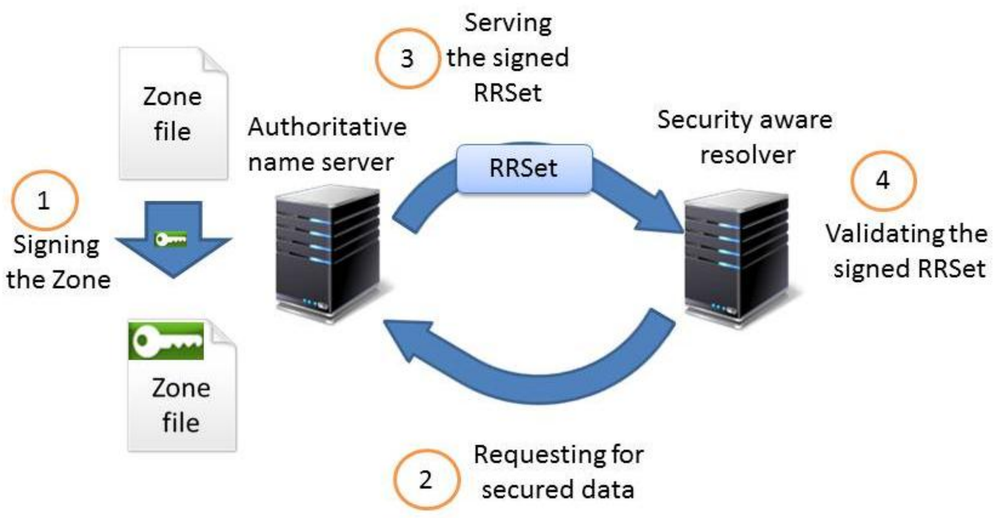


Figure 4.1 Signing, serving and validating the RRSet.

As it is demonstrated in Figure 4.1, the authoritative name server signs the zone and then it serves the security aware resolver with signed RRSets. The security aware resolver then validates the signed RRSet by checking its signature [29].

4.3.2 Trust model

Before adding security parameters to the DNS messages and checking their validity, first a trust relation should be established. This trust relation is from the root servers to the security aware authoritative name server, who wants to serve the resolver [30]. In order to have such a trust relation, it is required to perform some modifications to the current DNS tree.

This trust relation model is known as the “Chain of trust” [30]. Each parent node in the DNSSEC hierarchy is responsible to sign its zone and establish trust relations to its parent [30]. In 2010, the root domain was signed and it was considered as a secure point for all nodes in the DNS hierarchy. In the DNSSEC hierarchy, root gives delegation to gTLDs and ccTLDs. TLD domains are also responsible for signing their zones and sending the signing key to the root zone. In addition, they need to accept and authenticate keys from their child-zones, SLDs, and establish trust relations with them. This chain of trust continues until it reaches the leaf-nodes of the DNS tree. In the section of “Chain of Trust” (in section 4.4), we will explain this model in detail.

This model is somehow similar to the trust model, which exists in the Gnu Privacy Guard (GPG) [32]. This trust model is called “Web of Trust”. GPG is using the public key infrastructure (PKI) [33].

Within the following example, we will try to demonstrate the similarity and differences between these two models.

In this example, we have three key owners: Ann, Brook and Cole. Each person has one pair of asymmetric cryptographic keys, one private key and one public key. Ann and Cole want to establish a trust relation between each other by considering the following facts:

1. Ann signed Brook’s public key, with her private key.
2. Brook signed Cole’s public key, with his private key.
3. Ann trusts Brook.

Using the “Web of Trust” model, Ann can delegate or trust the responsibility of authenticating Cole’s public key to Brook. In other words, Ann trusts the public key from Brook and uses it for authenticating Cole’s key [32].

DNSSEC is similar to GPG as each parent in the DNSSEC hierarchy signs, not directly as we will describe later, the public key of its own child [30]. This means, if a resolver trusts the parent this trust can be inherited to the secured child. However, for implementing the “Web of Trust” some severs are dedicated to store public keys [32]. It is not practical to have dedicated public key servers to store public keys from all zones in the Internet [34]. Therefore, a different approach is needed, which is proper for the extended structure of the DNS. Chain of trust is a model that is designed for the DNS infrastructure and uses the DNS tree for establishing this trust. The chain of trust starts from the signed root and propagates until it reaches the end nodes of the DNS tree.

In the DNSSEC, the zone owners are responsible for generating keys and signing their zones. They are also responsible for accepting keys from their child zones, for establishing a trust relation. In other words, during the key generation and distribution in the DNSSEC hierarchy no third party is involved [29]. This feature differentiates DNSSEC from GPG and other applications, which are using PKI.

4.4 New Resource Records

As explained before, DNSSEC defines a new dataset for the DNS data-structure to perform the data origin authentication and integrity checking. The following sections will describe these new RRs.

4.4.1 DNSKEY Resource Record

DNSSEC provides source authentication by means of digital signatures. These signatures are generated and validated using asymmetric key cryptography [27]. Key generating tools of a security-aware name server generate private and public keys. The first key is used for generating signatures and the latter is used for validating signatures [28]. The security-aware name server uses the private key to add a signature for each RRSet then it puts the public key in a special RR called DNSKEY RR [29] in the zone file. The wire format of DNSKEY RR’s RDATA is illustrated in the Figure 4.2.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Flags            |    Protocol   |   Algorithm   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
/                            Public Key                         /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.2 DNSKEY RR’s RDATA wire format [21].

The Flags Field: Bit 7 in the Flags field is the “Zone Key Flag” and bit 15 is the “Secure Entry Point Flag” (SEP bit). Other bits are reserved and contain value 0.

The Protocol Field: This field always contains the value 3. If during the signature verification this value is different from 3, the message is considered as invalid.

The Algorithm Field: This field determines the cryptographic algorithm of the public key.

The Public Key Field: The public key is stored in this field.

It is recommended that two asymmetric key pairs are generated [35]. One pair is called the “Zone Signing Key” (ZSK) and the other is the called “Key Signing Key” (KSK). The ZSK’s private key is used for signing RRSets of a zone and its public key is stored in a DNSKEY RR. The KSK’s private key is used for signing DNSKEY RRSet [27].

Figure 4.3 shows an DNSKEY RR for a zone:

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w==  )
```

Figure 4.3 An example of an DNSKEY RR generated for the “example.com” zone [21].

The first text field is the owner name and the second text is the TTL value. Value 256 shows that this key is the ZSK, because the “Zone Key bit” in the Flags field of the RDATA is set to one. Value 5 is the cryptographic algorithm ID and indicates RSA-SHA-1 algorithm.

The different keys, which are used in the other DNS’s operations, i.e. the KSK public key, keys for SIG(0) and TSIG, are stored in other DNSKEY RRs [21]. However, the security-aware resolver only uses a DNSKEY RR with its “Zone key bit” set, for verifying the signatures of the RRSet [21]. Regularly, one ZSK’s private key is used to sign the data of a zone. It is also possible to generate multiple ZSKs and sign the same zone with them. Several ZSKs can be used for creating signatures for one zone using different cryptographic algorithms [37]. This can be helpful for special cases when a resolver does not support some of the cryptographic algorithms.

The KSK’s public key is sent to the security-aware resolver, using a secure channel. The resolver then adds this key to its configuration file [38]. This key is referred to as a “Trust Anchor” [27].

DNSSEC in general does not distinguish between ZSK and KSK [27], [21], [20]. One key pair can be used for the zone signing and signing the DNSKEY RRSet. From now, we will distinguish between ZSK and KSK key pairs [39].

Signing DNSKEY RRSet is not the only role of the KSK. The “Secure Entry Point” [40] is another name for the KSK public key. When a zone is signed and the administrator of that zone desires to establish a trust relation between that zone and its parent, one DNSKEY must be used as a reference to the signed child zone [27]. For this reason, during the generation of the DNSKEY, the “SEP” bit in the flag portion of the DNSKEY’s RDATA is set. This will help resolvers to distinguish this key from the other keys. During the establishment of the trust relation, administrator of the child zone sends the digest of KSK DNSKEY to the administrator of the parent zone in a secure manner, using IPSEC or secure e-mail. After receiving the digest of the public key, the parent’s administrator creates a new RR type from this digest, which is called “Delegation Signer” RR (DS RR) [41], and uses it as a secure reference to the child zone [29].

4.4.2 RRSIG Resource Record

Signing a zone is the major task of the security-aware name server. The security-aware name server uses one or more ZSKs to sign the RRSets of a zone. The zone, where the RRSets are located, is called the signer [21]. In the signing process, only authoritative RRs in the zone are signed with ZSK’s private key [27]. During the zone signing, a digital signature is assigned to each RRSet and is stored in a new RR, which is called RRSIG. The wire format of RRSIG RR’s RDATA is illustrated in Figure 4.4.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type Covered           |  Algorithm    |     Labels    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Original TTL                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Signature Expiration                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Signature Inception                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Key Tag            |                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+         Signer's Name         /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
/                            Signature                          /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.4 RRSIG RR’s RDATA wire format [21].

The Type Covered Field: This field is used for identifying the type of RRSet, which this RRSIG RR is covering.

The Algorithm Number Field: This field identifies the cryptographic algorithm. It is used for validating the signature.

The Key Tag Field: The DNSKEY RR’s key tag value is stored in this field. It is used for finding the right key in a short time for validating the signature.

The Signers' name: This field contains the owner name of the DNSKEY RR. It is used for validating the signature.

The Signature Field: This field contains the signature.

RRSIG is used in the authentication process. When a resolver in the DNSSEC architecture receives a response, it can verify the data origin of the RRSet by checking the signature that is entailed to it. Four new RR types are generated during the signing of a zone. These new RRs are RRSIG RR and DNSKEY RR, which already have been mentioned. NEXT Secure (NSEC) [42] and Delegation Signer (DS) [41] are the other new RRs, which are generated in the signing process and have special roles in the DNSSEC. During signing of a zone, for each DS RR, NSEC RR and DNSKEY RRSet a RRSIG RR is generated. These RRs are authoritative for the zone [20]. It is good to mention that during signing, RRSIG RRs are not signed, because the standard does not want to create infinite loops in the signing process [20].

RRSIG RR contains two special timing values. One is the "Inception" time, which indicates the time when a signature starts to be valid, and the other is the "Expiration" time, that indicates the time when the signature is going to be expired. These two times in the RRSIG RR define the "Validity Period" of the signature [27]. The validity period adds absolute timing to DNS. The inception and expiration times are taken from the current time of the signer. The lifetime of the signature is estimated by the validity period. This will limit the possibility of replaying signatures.

For creating RRSIG RRs different algorithms can be used, namely: DSA-SHA-1 [36], RSA-SHA-1 [37], RSA-SHA-256 [43], RSA-SHA-512 [43], Elliptic Curve DSA [58] and RSA-MD5 [44], where the latter is not recommended anymore for signing the zone [37]. For each algorithm a number is assigned, which is called the "Algorithm Number". The algorithm number, the owner name and the key id/tag field guide the resolver to use the correct public key for the validation in an efficient way [27]. Figure 4.5 shows RRSIG RR for an owner name in a signed zone.

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                              20030220173103 2642 example.com.
                              oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                              PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                              B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rzn18O3t
                              GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                              J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Figure 4.5 An example of an RRSIG RR, which is created for an owner name in a zone file [21].

The first value is the owner's name, which is signed. The following values are TTL, Class, and Type of the RRSIG RR. A is the type of the RR which is covered by the RRSIG RR. 5 is the cryptographic algorithm id and it indicates RSA-SHA-1 algorithm. 3 is the number of the labels in the owner name. The following number is the original TTL value of A RR. The last number is the expiration time of the signature in seconds. In the following parenthesis, we have four parameters: Inception time of the RRSIG, the key id/tag, the signer's name and the signature.

When a zone is signed, the NS RRs [42] residing in the zone are also signed except those NS RRs and A RRs, which are appearing at the delegation point. A RR is called the glue records, which contains the address of the delegated name server [20]. These records are non-authoritative RR and belong to the child domain. That is why they are not signed.

4.4.3 Next Secure Resource Record (NSEC RR)

Previous RRs provide data origin authentication and integrity for the DNS transactions. To provide negative authenticated response, NSEC RRs are added to the zone file.

The wire format of NSEC RR's RDATA is illustrated in Figure 4.6.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                      Next Domain Name                         /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                       Type Bit Maps                           /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

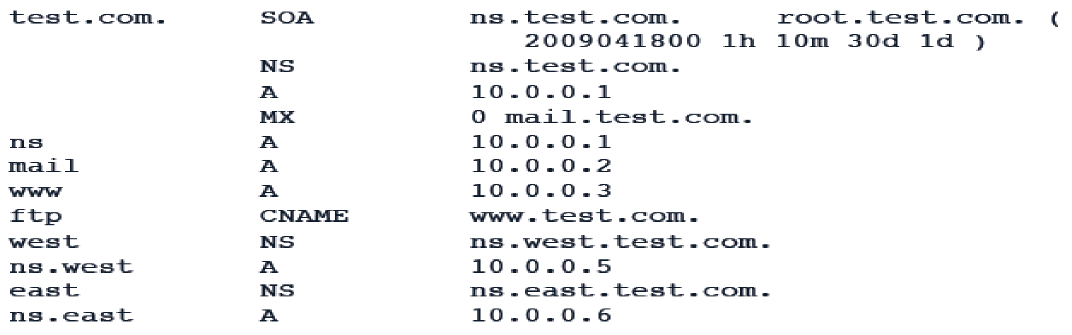
Figure 4.6 NSEC RR's RDATA wire format [21].

Next domain name: This field Contains next owner name in the canonical [20] ordering of the signed zone.

Type Bit Maps: This field identifies the Type of the RR of the owner names.

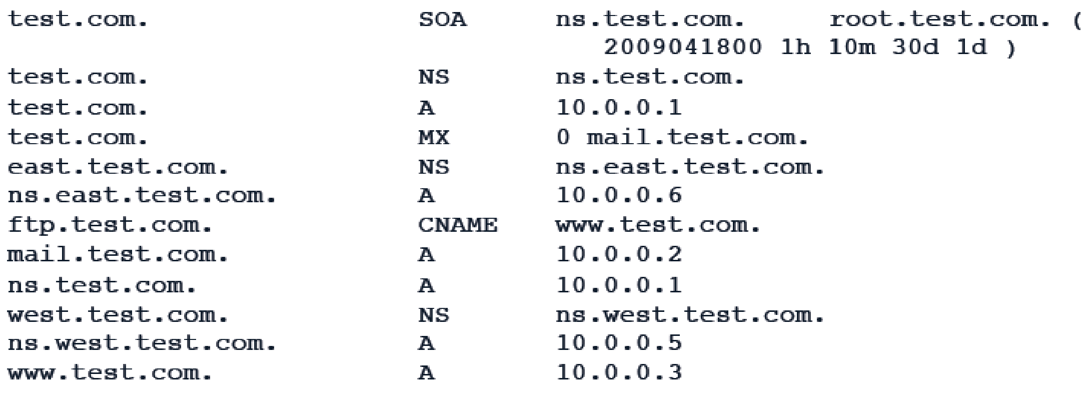
During the signing of each zone, all names are ordered in the canonical order [21]. After ordering the owner names, they are expanded. It means that zone apex, the apex is the SOA owner name that appears in the root of the zone, is entailed to all owner names [21]. In the last step, after RRSets with the same owner name but different types an NSEC RR is included.

Figure 4.8, 4.9 and 4.10 illustrate one zone before and after including NSEC RRs to its RRs.



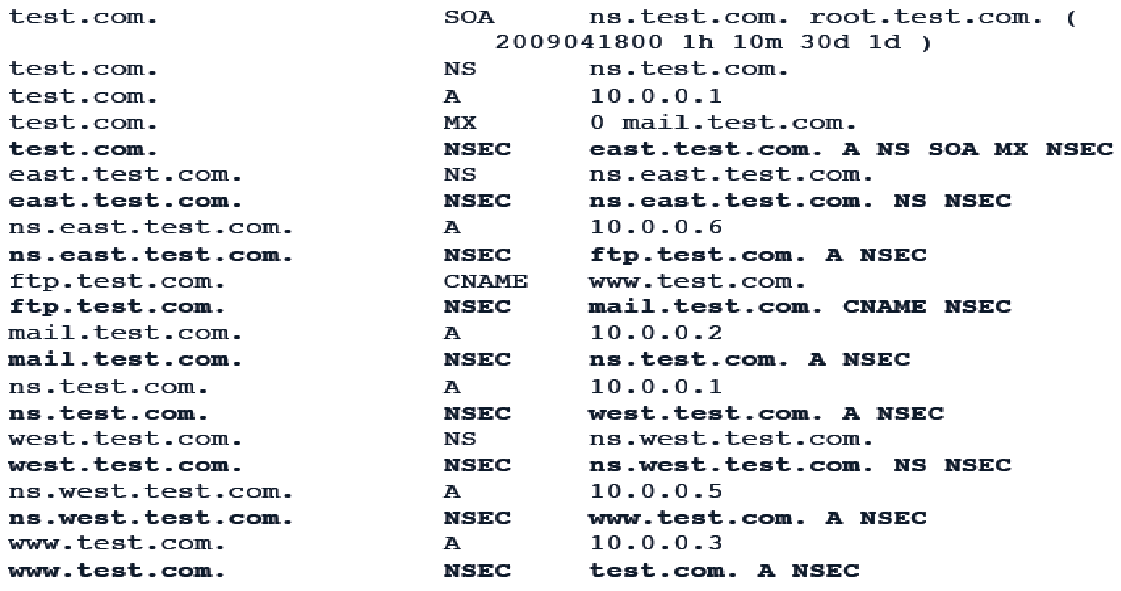
```
test.com.       SOA        ns.test.com.      root.test.com. (
                              2009041800 1h 10m 30d 1d )
                NS         ns.test.com.
                A          10.0.0.1
                MX         0 mail.test.com.
ns              A          10.0.0.1
mail            A          10.0.0.2
www             A          10.0.0.3
ftp             CNAME      www.test.com.
west            NS         ns.west.test.com.
ns.west         A          10.0.0.5
east            NS         ns.east.test.com.
ns.east         A          10.0.0.6
```

Figure 4.8 A zone before the inclusion of NSEC RRs [46].



```
test.com.                  SOA      ns.test.com.     root.test.com. (
                                       2009041800 1h 10m 30d 1d )
test.com.                  NS       ns.test.com.
test.com.                  A        10.0.0.1
test.com.                  MX       0 mail.test.com.
east.test.com.             NS       ns.east.test.com.
ns.east.test.com.          A        10.0.0.6
ftp.test.com.              CNAME    www.test.com.
mail.test.com.             A        10.0.0.2
ns.test.com.               A        10.0.0.1
west.test.com.             NS       ns.west.test.com.
ns.west.test.com.          A        10.0.0.5
www.test.com.              A        10.0.0.3
```

Figure 4.9 A zone, with the expanded owner names. They are also sorted in the DNS name canonical order [46].



```
test.com.                  SOA      ns.test.com. root.test.com. (
                              2009041800 1h 10m 30d 1d )
test.com.                  NS       ns.test.com.
test.com.                  A        10.0.0.1
test.com.                  MX       0 mail.test.com.
test.com.                  NSEC     east.test.com. A NS SOA MX NSEC
east.test.com.             NS       ns.east.test.com.
east.test.com.             NSEC     ns.east.test.com. NS NSEC
ns.east.test.com.          A        10.0.0.6
ns.east.test.com.          NSEC     ftp.test.com. A NSEC
ftp.test.com.              CNAME    www.test.com.
ftp.test.com.              NSEC     mail.test.com. CNAME NSEC
mail.test.com.             A        10.0.0.2
mail.test.com.             NSEC     ns.test.com. A NSEC
ns.test.com.               A        10.0.0.1
ns.test.com.               NSEC     west.test.com. A NSEC
west.test.com.             NS       ns.west.test.com.
west.test.com.             NSEC     ns.west.test.com. NS NSEC
ns.west.test.com.          A        10.0.0.5
ns.west.test.com.          NSEC     www.test.com. A NSEC
www.test.com.              A        10.0.0.3
www.test.com.              NSEC     test.com. A NSEC
```

Figure 4.10 A sample zone after the inclusion of NSEC RRs [46].

This procedure is performed to keep track of the existing and non-existing owner names in the signed zone. The NSEC RRs create a chain of host names in the canonical order of the zone. NSEC RRs help the security-aware name server to provide authenticated denial of existence [27]. The complete set of NSEC RRs forms a chain of owner names [21]. DNSSEC uses an algorithm to find the proof of non-existence for authoritative names. This algorithm will be described in the “DNSSEC protocol modifications” section (in section 4.4).

```
alfa.example.com. 86400 IN NSEC host.example.com. (
                                A MX RRSIG NSEC TYPE1234 )
```

Figure 4.7 An example of an NSEC RR, which is created for an owner name in a zone file [21].

Figure 4.7 shows an NSEC RR for the owner names in a signed zone. The first value is the name of the owner names that NSEC RR is created for them. The following values are TTL, Class and the Type of the NSEC RR. The last value is the next owner name in the canonical order. The values in the parenthesis are the Type of the owner names that this NSEC RR covers.

4.4.4 Delegation Signer Resource Record (DS RR)

Security-aware resolvers, in a connected DNSSEC tree, will follow their queries from one secured parent zone to its secured child zone. Before traversing between two secured zones, a trust relation should first be established between these zones [30]. DS RR is a RR designed for creating a trust relation between two signed zones. The DS RR is placed in the parent zone and refers to a DNSKEY RR in a child zone. This DNSKEY RR is the KSK’s public key. In the chain of trust, the resolver first authenticates the DS RR and then uses the digest of the KSK in the DS RR’s RDATA field to authenticate the KSK of the child zone [21].

The wire format of DS RR’s RDATA is illustrated in Figure 4.11.

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Key Tag             |  Algorithm    |  Digest Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
/                            Digest                             /
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.11 DS RR’s RDATA wire format [21].

The Key tag, the algorithm and the digest of the DNSKEY RR are stored in DS RR’s RDATA. The KSK’s public key is used as the DNSKEY RR.

Key TAG: This field contains KSK's public key id/tag.

Algorithm: This field contains the KSK's cryptographic algorithm id.

Digest Type: This field contains the algorithm, which is used for creating the digest from the KSK's public key.

Digest: This field contains the digest of the KSK's public key.

When a zone is signed the digest of its DNSKEY, the KSK's public key, is sent to the parent zone. In the parent zone, the DS RR is created from this digest [40]. This process is illustrated in Figure 4.12.

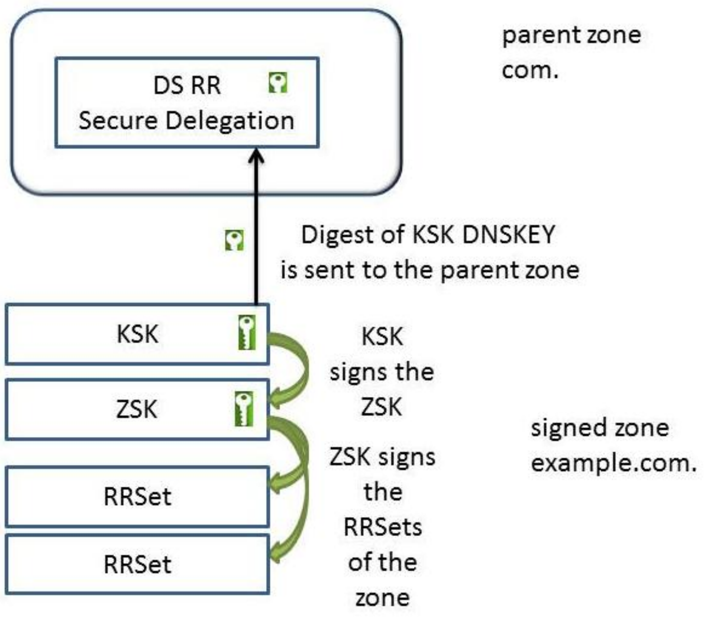


Figure 4.12 This figure demonstrates the procedure to establish a trust relation between the signed child zone and its parent.

The DS RR and the DNSKEY RR have the same owner name, child zone apex, but they are stored in different places. This placement simplifies the administrative tasks due to signing delegations within the organizational boundaries [27]. The authentication procedure will be discussed in more details in the "Security Verification" section. Figure 4.13 shows an DS RR in the ".com." zone. It is referring to a DNSKEY RR, which is used for signing the child zone.

```
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                           98631FAD1A292118 )
```

Figure 4.13 An example of an DS RR, which is referring to the DNSKEY RR of a secured child zone [21].

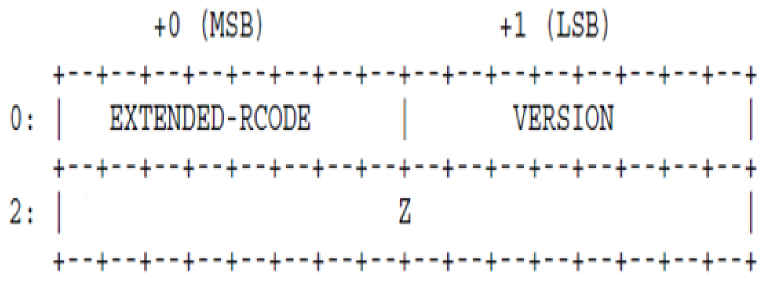
Like previous RRs, the first four fields are the owner name, TTL, Class and Type of DS RR. The value 60485 is the key id/tag for the DNSKEY RR. Value 5 is the algorithm id, which is used for signing the child zone. Value 1 is the algorithm id, which is used for creating the digest of DNSKEY RR. The value in parenthesis is the digest.

4.5 Protocol modifications

As mentioned, DNSSEC modifies the DNS protocol. We will explain these modifications below in different sections for each entity in the DNSSEC infrastructure.

4.5.1 Support for EDNS0 (Extension Mechanisms for DNS v.0)

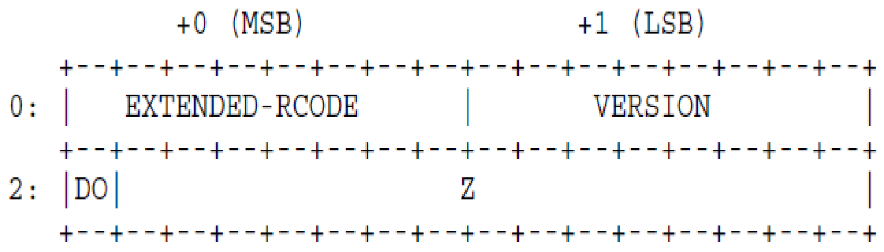
When the DNS entities use TCP connections for exchanging DNS messages, there is no problem for sending large packets. One problem with DNS's TCP connections is that they add a large overhead. These overheads are related to the connection setup and teardown [47]. DNS uses UDP connections for exchanging transactions more than TCP connections to reduce these overheads [48]. In the case of using UDP, based on the DNS's specifications, some restrictions limit the UDP packet size to 512 octets [1]. After deploying the DNSSEC, the size of the DNS messages are increased [21], therefore some changes are applied to the DNS protocol. These modifications help the servers to send the UDP packets with sizes larger than 512 octets. EDNS0 is a standard for these modifications [49]. EDNS0 applies these modifications by changing the header of the DNS message. As we can see from Figure 4.14, EDNS0 adds extension to ROCDE and the Z field in the header of the DNS message.



```
            +0 (MSB)                  +1 (LSB)
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  0: |   EXTENDED-RCODE      |       VERSION         |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
  2: |                       Z                       |
     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4.14   Extended-RCODE and Z field [49]

EDNS0 helps the requestor to specify its desired UDP packet size for the sender. It is suggested that all the entities in the DNSSEC aware environment support the EDNS0 or its later versions [20]. In a DNSSEC aware environment when a resolver asks for a name resolution, which contains DNSSEC RRs, it shows its intention by setting the DO (DNSSEC OK) bit. DO is the most significant bit in the Z field of EDNS0 OPT header [47], as is illustrated in Figure 4.15.



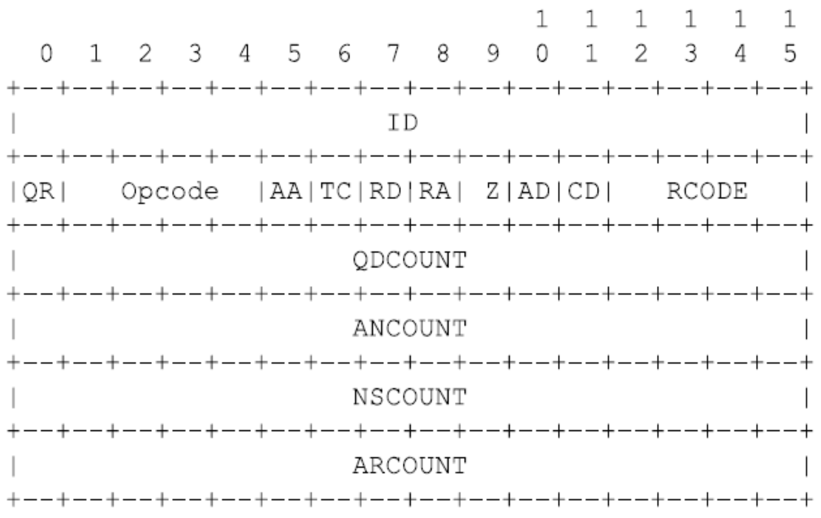
```
             +0 (MSB)                            +1 (LSB)
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 0: |   EXTENDED-RCODE      |         VERSION       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 2: |DO|                    Z                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4.15 DO bit in the EDNS0 OPT header. If this bit is set to one it indicates that the requestor is DNSSEC enabled [47]

EDNS0 is not only used for DNSSEC. Generally, it is used in cases that requires an increase in the size of DNS UDP packets i.e. such as when deploying IPV6 for name servers [50], [51].

4.5.2 Authenticated Data (AD)

DNSSEC introduces two new mechanisms: Authenticated Data (AD) and Checking Disabled (CD) [20]. DNSSEC chose two unused bits from the DNS header [49] and used them for communications between the security-aware name servers and security-aware resolvers [52]. These bits are shown in Figure 4.16.



```
                                 1  1  1  1  1  1
   0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                      ID                       |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |QR|   Opcode  |AA|TC|RD|RA| Z|AD|CD|   RCODE   |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    QDCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    ANCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    NSCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 |                    ARCOUNT                    |
 +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4.16 Location of the AD (Authenticated Data) and CD (Checking Disabled) bits in the DNS header [4]

If the security-aware recursive name server sets the AD bit it means the name server specifies to the security-aware resolver that all the information, which it has received are authentic [20].

The security-aware resolver uses the CD bit. We will discuss the CD bit and its feature in the protocol modifications related to the security-aware resolver (in section 4.5.2).

4.5.3 Protocol modifications related to the security-aware name server

DNSSEC applies some modifications in the DNS protocol to enable the security-aware name server to serve security-aware resolvers with signed RRSets. These modifications are as follows.

Including RRSIG RRs in the response

When a zone is signed, the security-aware name server includes RRSIG RR after each RRSet in the zone. When the security-aware name server receives a name resolution request from a security-aware resolver, it will include the RRSIG RR in the response for RRSet. By this parameter, the name server will add integrity and data origin authentication to the message.

If there is no space for including the signature, the security-aware name server must set the TC (truncated header) bit. When the resolver receive a response with the TC bit set it should ignore the response and perform another query using the TCP connection [45].

Including DNSKEY RR in the response

A security-aware name server may include the DNSKEY RRSet in the response messages. These keys are used for validating the signed RRSets. When the DNSKEY RR is included in the “Answer” section, it has the lowest priority to any other information that is included in this section [20]. The resolver can also ask for the DNSKEY in another query when it requires a key for authentication.

Including NSEC RR in the response

A security-aware name server should indicate in the response message if the queried name did not match the existing owner names in its zone. As it was described before, NSEC RR provides two things for a security-aware name server: the next owner name and a set of RR Types for the owner names which the NSEC RR covers [42]. The next owner name can be the next authoritative RR or a NS RR (referral), in the delegation point of the zone. The next owner name indicates if a specific queried name exists in that signed zone or not [21]. When a security-aware name server receives a query where the DO bit is set, with a request for an RR, which does not exist in that particular signed zone, it responds to the requestor by including the NSEC RRs of the previous name in the chain as a proof of non-existence [20].

The Security-aware name server can locate the specific RRSet or its non-existence by performing the following algorithm.

The algorithm tries to prove that there is no owner name that matches the queried name ex. N. As it was described before, during the signing of the zone a chain of NSEC RR is created from all owner names. The RRs are ordered in the canonical ordering of the zone. The algorithm should find a name M as a proof of non-existence, which in the “Next Domain Name“ field of the covering NSEC RR for M another name exists that is higher than N. When the M is found, then the name server includes M‘s covering NSEC RR and RRSIG RR in the response message as an authenticated denial of existence for the queried name N.

Including DS RR in the response

In some cases, a security-aware name server may receive a query for a name that does not exist in its zone but it knows that the name might be stored in its child zone. If the queried name server has the referral information about the next zone, it should return NS RR plus A RR that contains the name server address of the child zone. The security-aware name server should also include DS RR and its associated RRSIG RR in the "Authority" section as DNSSEC security parameters [20].

Right now, in some part of the DNS tree, there is no full connection from the root to the end nodes. In some part of this hierarchy, some zones are not signed or trust is not established between them and their parents.

If the child is not signed and the DO bit of the query message is set, the security-aware name server should return a "No Data" message. In this case the security-aware name server should return both NSEC RR and its related RRSIG, to prove that DS RR does not exist in the parent zone. The security-aware name server should also send the NS RR plus A RR to redirect the resolver to the new name server [20]. It is important to know that the DNSSEC does not protect the child zones, which are not signed. The resolver will continue its iterative query from the child zones in an unsecure way.

Including information regarding referrals and security parameters in the response message will increase its size. If there is not enough space for referral information, unnecessary data like A RRs will be omitted. If there is still not enough space, the security-aware name server must set the TC bit [20].

4.5.4 Protocol modifications related to the security-aware resolver

Some protocol modification is needed to enable the security-aware resolver to request for DNSSEC messages and validating the authenticity of the receiving RRSets, which are as follows.

Checking Disabled

The CD bit is set by the security-aware and validating resolver to disable the signature validation in the security-aware recursive name server. After receiving a query, with the CD bit set in the header, the recursive name server passes the queried information and covering signature to the resolver. The name server copies the CD bit state in the DNS header without modifying it to inform the security-aware resolver that it should do the signature verification itself [20]. We discussed in the previous chapter about resolvers concern of non-trustworthy name servers. This feature allows resolvers to check the validity of the receiving data with their own configured public keys. This feature allows resolvers to request for the raw data, RRSet with its related security parameters, from their security-aware recursive name server. The reason for such a request can also be due to the existence of an "island of security" [27] in the DNS architecture or misconfiguration in the clock of the recursive name server [20].

Signature Validation

If the security-aware resolver has no key for a signed RRSet, it should request it from the sender. After the request, it will receive a DNSKEY RRSet. It can validate signatures using the correct DNSKEY RR from the DNSKEY RRSet. For validation, the resolver must check whether the RRSIG covers the receiving RRSets and then check the validity period of the RRSIG [20].

Signature Verification

The resolver uses the DNSKEY RR or the DNSKEY RRs for deciphering signatures. The resolver needs to find out the correct cryptographic algorithms, for deciphering RRSIG RRs, from the algorithm id field in the DNSKEY RR. After deciphering, the resolver compares the result with the RRSet. If the result of comparison is true then RRSet is the original data and is not spoofed.

If several keys are used for signing the zone, the security-aware resolver should continue this deciphering, using different DNSKEY RRs and comparison operation [20].

Caching

After accepting the RRSet, the security-aware resolver caches the RRSet and its covering RRSIG RR. Before caching, the TTL value should be assigned to RRSIG RR. The TTL must not be greater than the minimum of the TTL value of the RRSet, the TTL value of the RRSIG RR, the Original TTL value of the RRSIG RR and the difference of the server's current time and the signature expiration time of the RRSIG RR in the response message [20].

Authenticated Denial of Existence

When a security-aware resolver receives a proof of non-existence for an RRSet, it first should determine whether this information is valid and authentic by checking the RRSIG RR covering the NSEC RR. After determining the authenticity of the receiving data, it should update its cache information.

Denial of existence is determined when the requested name appeared after the authenticated NSEC's owner name, in canonical order, and before the name in the NSEC's Next Domain Name field. If the name exists, the resolver should check the requested name's type with all RR types that they are presented in the NSEC RR. If no match is found, it proves that the queried name did not exist [20].

4.6 Chain of Trust

When a zone in a security-aware name server is signed, the name server is ready to serve the security-aware resolvers with signed RRSets. At this level, the signed zone is referred to as an "Island of security" [27]. The reason for this naming is that no trust relation is established between this zone and other secured zones in the DNS hierarchy. To help the security-aware resolver to find this secured zone, the signing key, is added to the configuration file of the security-aware resolver [34].

The resolver needs to store all DNSKEY RR from the signed zone for validating signed RRSets. In the real world scenario, it is not practical for the resolver to store these keys from all zones in the Internet. In addition, handling of these signing keys will become complicated for zone's administrators [34]. The earlier idea, used for one version of the DNSSEC's implementation proved these complications [4], [3].

The IETF DNSEXT group decided to change the key management problem by investigating other methods. The idea, that made the DNSSEC practical, was to establish a "Chain of Trust" [27] within the DNS hierarchy. As it was described before, the new resource record called the DS RR and it is the essential part of this chain [41]. The DS RR makes a secure reference to the trusted child zone. It also removes the complication of handling the DNSKEY from different zones [20]. This chain of trust is established from the root zone, ".", and extends to end nodes of the DNS tree.

In theory, all zones from the root to the leaf of the DNS tree are signed and delegate the trust, in a secure way, to their child zones. In this model, all zones are responsible for signing their zones and sending RRSIG RRs and DNSKEY RRSet to the security-aware resolvers. In this model, the resolvers only need to anchor the KSK DNSKEY of the root zone and start their iterative query process from the root zone.

4.6.1 Security verification

Following the previous discussion, the security-aware resolver only needs to anchor the root's KSK to its configuration file and then request secured RRs, i.e. asking the IP address of "www.example.com", from the root zone. Root has no information about the queried name so it refers the resolver in a secure way to the TLD zone. The resolver continues following the chain of trust and authenticating keys until it reaches the desired zone and receives its signed RRSet. Finally, the resolver can validate the signature of the RRSet by requesting the signing key from the last node.

Now it is good to demonstrate this process in detail, using the definitions described in the previous sections. In our example, there is a security-aware resolver, who anchored the KSK of the root in its configuration file. Now it is looking for the IP address of the "www.example.com" webserver through the iterative queries. This process is shown in Figure 4.17.

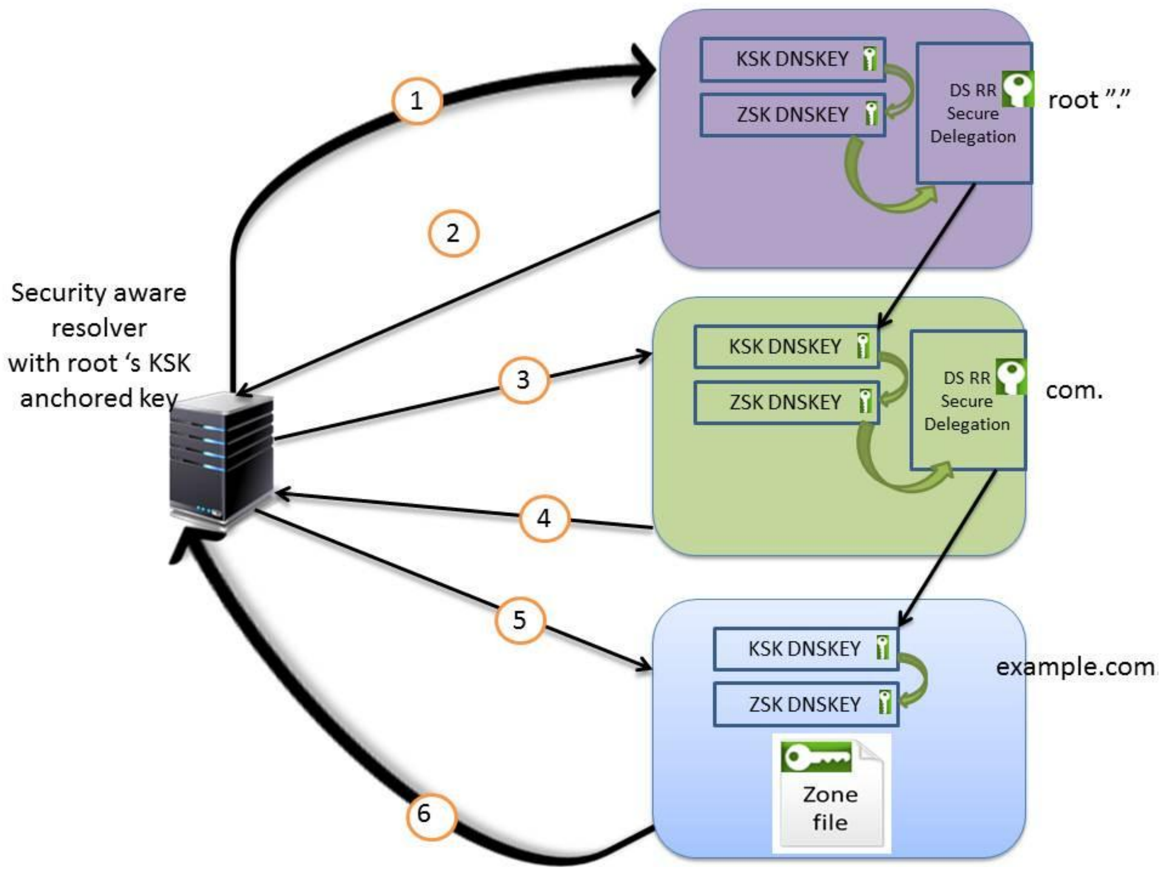


Figure 4.17 Secure iterative queries within the "Chain of Trust" [53].

When a security-aware resolver faces a name resolution for "www.example.com", it first checks in its cache to see if it can find an entry related to that domain name or not. In this case, there is no entry in the cache, so the resolver starts its queries from one security-aware name server of the root. The resolver should set the DO bit in the Z field of EDNS0 OPT header to indicate that it is a security-aware resolver and it is asking for secured RRSet.

Now we follow the steps from the Figure 4.18 [20], [34], [38], [53].

1- The security-aware resolver sends the request for the IP address of the "www.example.com".

2- The root server responds with a referral to its TLD child zone ".com.". Root sends NS RRSet plus A RRSet that contains the IP addresses and names of the name server s of the child zone. It includes in the response the DS RR with its covering RRSIG RR. Root also replies with the DNSKEY RRSet. This RRSet contains the public keys. Root used the ZSK for signing all RRSets in its zone and used KSK for signing the DNSKEY RRSet. When the root server sends the response, it includes the RRSIG RR that covers the DNSKEY RRSet.

When the resolver receives the DNSKEY RRSet it should verify its signature. The resolver uses its anchored KSK key to decipher the signature using the cryptographic algorithm mentioned in the KSK’s DNSKEY RR. The resolver compares the deciphered signature with the DNSKEY RRSet. If the result of the comparison was true then the keys in the DNSKEY RRSet are authentic.

After authenticating ZSK, resolver deciphers the RRSIG of the DS and compares it with DS RR. If the result of the comparison is true then the DS RR is authenticated. The resolver now follows its query from “.com” zone and uses the DS RR to authenticate the KSK DNSKEY of “.com” zone.

3- The resolver sends a message and requests for IP address of “www.example.com” from the name server of the “.com” zone.

4- The name server responds with the referral to its child zone, “example.com.”. It includes the DS RR with its covering RRSIG RR in the response. The name server also responds with the DNSKEY RRSet and includes its covering RRSIG RR. The resolver looks in the DNSKEY RRSet to find a key with the “SEP” bit set. After finding the KSK, the resolver creates a hash from the KSK DNSKEY RR. It uses the cryptographic algorithm that is mentioned in the RDATA of the DS RR, which was verified in step 2. Then, the resolver compares the result with the digest of the KSK from the digest field in the DS RDATA. If the result of the comparison is true then the KSK DNSKEY is authenticated. Then it uses the KSK DNSKEY to verify the RRSIG RR, which covers the DNSKEY RRSet. After the signature verification, all keys that are included in the DNSKEY RRSet are authenticated. Now the ZSK, which was used for signing the RRSet in the “.com.” zone, is authenticated.

The resolver deciphers the enclosed RRSIG and compares it with the original DS RR. If the result of the comparison is true then the DS RR is authenticated. The resolver now follows its query from the “example.com” zone and uses the DS RR to authenticate the KSK DNSKEY of “example.com” zone.

5- The resolver sends a message and requests to the IP address of “www.example.com” from the name server of the “example.com”.

6- The name server has the A RR in its zone, which contains the IP address of the “www.example.com” webserver. It includes this information in the response with the covering RRSIG RR, which was generated with ZSK. The name server also responds with the DNSKEY RRSet and includes its associated RRSIG RR in the response message. The resolver looks in the DNSKEY RRSet to find the KSK. After finding the KSK, the resolver creates a hash from the KSK DNSKEY RR using the cryptographic algorithm that is mentioned in the RDATA of the DS RR. The resolver compares the result with the digest of KSK, which was obtained from the RDATA field of the DS RR. If the result of the comparison is true then the KSK DNSKEY is verified. It uses the KSK DNSKEY to validate the RRSIG RR, which covers the DNSKEY RRSet. After validating the signature, all keys that are included in the DNSKEY RRSet are authenticated.

Now the security-aware resolver can verify the RRSIG RR’s signature using the authenticated ZSK DNSKEY. Finally, the resolver receives the IP address of “www.example.com” within the secure iterative queries.

Chapter 5 DNSSEC security and availability issues

In this chapter, we will analyse the current implementation of the DNSSEC. We will go through the security issue related to this implementation. It is important to become familiar with these security issues and try to avoid any implementation, which makes the DNS infrastructure vulnerable to the DNSSEC related attacks.

5.1 Zone enumeration

In the previous implementation of the DNSSEC protocol, NSEC provided proof of non-existence by creating a list from all RRSets in a zone. However, attackers are able to reconstruct the zone by using the next owner name field of the NSEC RR. In addition, each NSEC RR contains a list of the RR Types for the owner names it is covering. This threat is known as "Zone Walking" or "Zone Enumeration" [26]. The chain of RRSets makes it easy for attacker to find the existing owner names in the zone. This attack is not the direct attack to the DNS but it reveals the existing hosts and servers in the zone, which has been enumerated. Attacker can use this information for the other attacks.

NSEC3 [6] is another implementation of the next secure RR, which was proposed to counter the zone enumeration attack. NSEC3 is not replacing the NSEC. Both next secure types should be supported by the signing systems. The goal of NSEC3 is to add more complexity and more cost for the attacker to perform the zone enumeration. When a zone is signed using NSEC3, all owner names are hashed, with the SHA-1 or SHA-256 algorithm. These hashed owner names are sorted in the hash order of the signed zone. The zone signer creates a list of the NSEC3 RRs. Like NSEC RR, one field is assigned for the next owner name in the NSEC3 RR. The "Next Hashed Owner Name" field in NSEC3 RR contains the next hashed owner name in the hash order of the signed zone [6].

When a name server receives a query for an owner name, if there is no such owner name in its zone, NSEC3 RRs are sent as the proof of non-existence [7]. On the other side, when the resolver receives NSEC3 RRs as a proof of non-existence, it creates a hash from the queried name. The resolver tries to find if the hash falls in the interval of the hashes in the owner name and the next owner name field. This will prove that the queried name does not exist.

The NSEC3 uses two parameters, salt and iteration, for hashing. Random salt value is appended to the zone owner name before hashing. This will increase the cost of creating a prebuilt hash table and make the offline attack much harder for attacker. Nevertheless, it is still possible for attacker to create a large table of all possible owner names from the zone and check the returned hash name with the hash of entries of its prebuilt table.

By iteration, the result of the hash is used as an input for creating another hash. This procedure is performed several times. Iteration must be used with salting to resist against the zone enumeration.

In our solution, we will use the NSEC3 but with some considerations because this implementation is still not secure against the zone enumeration.

5.2 Replay attack

In the DNS design, it is not reasonable that resolver starts the iterative queries from the root servers for every name resolution. Considering this fact, for better performance DNS resolvers rely on caching.

Before implementing a security mechanism for the DNS, the resolvers cached the resolved names and stored them for a limited time. This time was defined by the TTL value. After deploying DNSSEC, RRSet and its related RRSIG and the DNSKEY that verifies the RRSIG are also cached in the resolvers. These RRSets stays at the cache of the resolver until their covering signature has expired. Different scenarios can be defined for the replay attacks but they have one common property. In these attacks, attacker keeps the RRSet and the covering RRSIG and replays these records until the covering signature has expired.

As an example, when the KSK is compromised or accessed by an unauthorized user, the administrator of the zone will revoke the key (see section 6.5.1) and replace it with a new key. However, the information of the DNSKEY RRSet stays in the cache of the resolvers and is considered as valid until the covering signature expires. The attacker who is aware of the compromised key can make use of such a case and replay the DNSKEY RRset and its RRSIG RR. The DNSSEC cannot protect the DNS if the key is compromised. The attacker can perform different forms of attacks before the cached information has expired. We should consider this security issue in our solution.

The only countermeasure for this vulnerability is to lower the validity period of the signature however at the same time it will raise some operational problems.

5.3 Unsigned referral

As we discussed earlier (see section 4.2.2) the referrals (NS RRSet and A RRSet) are not authoritative RRSets in the parent zone so they are not signed. If the attacker forges the NS and A RRSet in the answer, the resolver cannot detect the forgery. These records have no covering signatures and the resolver can be redirected to a spoofed zone.

However, DS RR and its covering signature are included in the response. DS carries the digest of the KSK from the child zone. DNSSEC aware resolver checks the DS RR before going to the spoofed zone. Resolver first authenticates the KSK, using the DS RR, and then uses it to authenticate the DNSKEY RRSet. Attacker does not possess the private part of the KSK of the child zone. Therefore, attacker is unable to sign and send the DNSKEY RRSet to the resolver. The Resolver will understand that it is redirected to the spoofed zone and the attack is pulled off.

The only security concern with the unsigned referrals is that if the KSK of the child become compromised the attacker can misdirect resolvers to the spoofed zone. As it is suggested in [5], in the signing systems NS and A RRSet are better to be signed with the ZSK of the parent. If the KSK of the child zone become compromised, the attacker cannot endanger the child zone, as the resolver will detect any forgery in the NS and A RRSet.

5.4 Amplification attack

The DNS amplification attack is one type of DDoS attacks. The main goal of the amplification attack is to saturate the bandwidth of the communication line of the target name server [59].

In the amplification attack, the attacker tries to redirect traffic to a DNS server, which is the victim of the amplification attack. The attacker uses IP spoofing. The attacker sets the source address of the DNS queries, generated from each machine in the Bot-Net, as the IP address of the target DNS server. These queries are sent to one or more misconfigured open DNS name servers that the attacker previously added a large TXT RR in their zone files. The answer of these queries will contain this large TXT RR. After triggering the attack by the attacker, queries are generated from the machines in the Bot-Net and sent to the misconfigured open DNS server. The misconfigured open DNS server returns answers to the victim and will overwhelm the communication line of the DNS server.

With DNSSEC, it is much easier for attackers to perform DNS amplification attack [60]. Before DNSSEC, attackers used large TXT RR to increase the size of the DNS messages. Attackers can now send queries to the DNSSEC enabled name servers. When the DNSSEC is deployed, the DNS messages get considerably large due to the addition of RRSIGs and DNSKEY RRSet. The main reason is the usage of the RSA cryptographic algorithm, as its key and signature sizes are relatively large. The DNSSEC message size in some cases can exceed to 4096-bit [48].

In addition, the attackers do not need to take control of the DNSSEC enabled name servers to add large RRs. Proper countermeasure should be defined for this threat in our solution.

5.5 Other security issues

DNSSEC is still using the same DNS header format. This protocol defines no mechanism for checking the integrity of the DNS header. In our design, we should consider this fact as the attacker is able to modify the AD bit in the DNSSEC header and poison the cache of the non-validating resolver.

DNSSEC introduced the concept of absolute timing. The RRSIG RR has a validity period (defined by the inception and expiration time). These timing values are taken from the clock of the signing system. The attacker is able to perform attacks to change the current time of the signing system. In this case, if the date and the clock of the signing system are moved backwards by the attacker, this system will serve expired signatures to the resolvers. We need to consider this security issue in our solution.

Chapter 6 Deploying DNSSEC

In this chapter, we will cover the deployment of the DNSSEC within a particular organization (VOLVO IT) and discuss potential issues and solutions. The main security threats regarding the DNSSEC protocol were demonstrated in the previous chapter, there are some available solutions for these issues that are used as the countermeasure in our solution. However, for a few cases the solutions do not completely remove the threats. In addition, to set up the DNSSEC solution many considerations should be taken into account to assure the security of the DNS infrastructure.

6.1 DNS infrastructure

A case study was performed to get a complete understanding of the DNS infrastructure of the VOLVO IT company to determine the requirements for deploying the DNSSEC. VOLVO IT is an international company, which provides IT services to the different customers (mainly to the VOLVO Group company). In this report, an abstract model is used to demonstrate the DNS infrastructure of this company. Customers are located in different geographical locations. Due to this fact, DNS infrastructure is extended in a wide area. For simplicity, the factor of the geographical location is omitted from this model.

We will use this abstract model to explain the required changes and modifications for enabling the DNSSEC in this particular environment. This model consists of three parts:

- External DNS
- Caching servers
- Internal DNS

6.1.1 External DNS

Name servers, in the external DNS infrastructure, are illustrated in Figure 6.1.

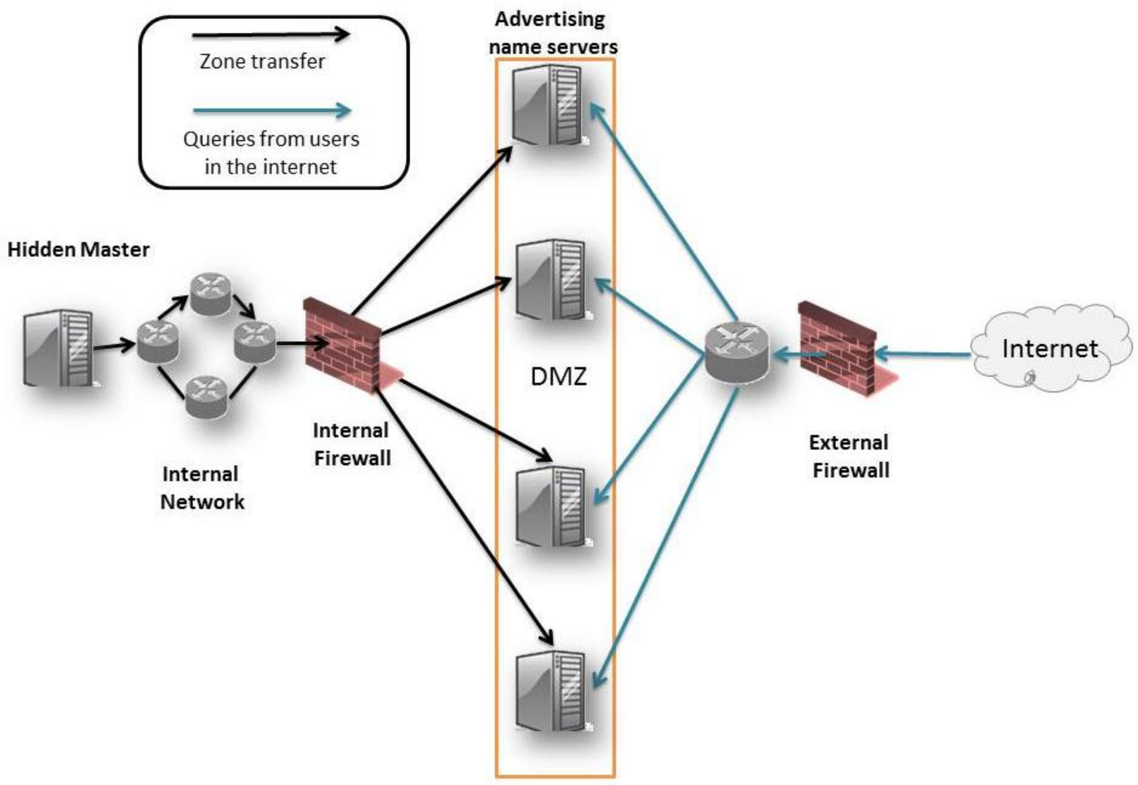


Figure 6.1 External DNS infrastructure - Name servers.

The External DNS provides name resolution for clients in the Internet. In Figure 6.1, one DNS server is assigned as a “Hidden Master”. The role of the hidden master is to publish zones to the secondary name servers, which are known as the advertising name servers. Zone changes are applied to the hidden master. Then zones are transferred to the advertising name servers. Data flow is from the hidden master to the advertising name servers. Keeping the hidden master server out of reach of the external users will guarantee the security of the public zone. In this infrastructure, four advertising name servers are used. Distributing users between different servers will increase the performance of the external DNS. When the hidden master becomes faulty, no interrupt in the external DNS service is sensed because advertising name servers are available to answer any request. Advertising name servers are placed in the Demilitarized Zone (DMZ).

6.1.2 Caching servers

Cache only name servers, resolvers, are responsible to perform the name resolution for the internal users. Caching name servers, in the external DNS infrastructure, are illustrated in the Figure 6.2.

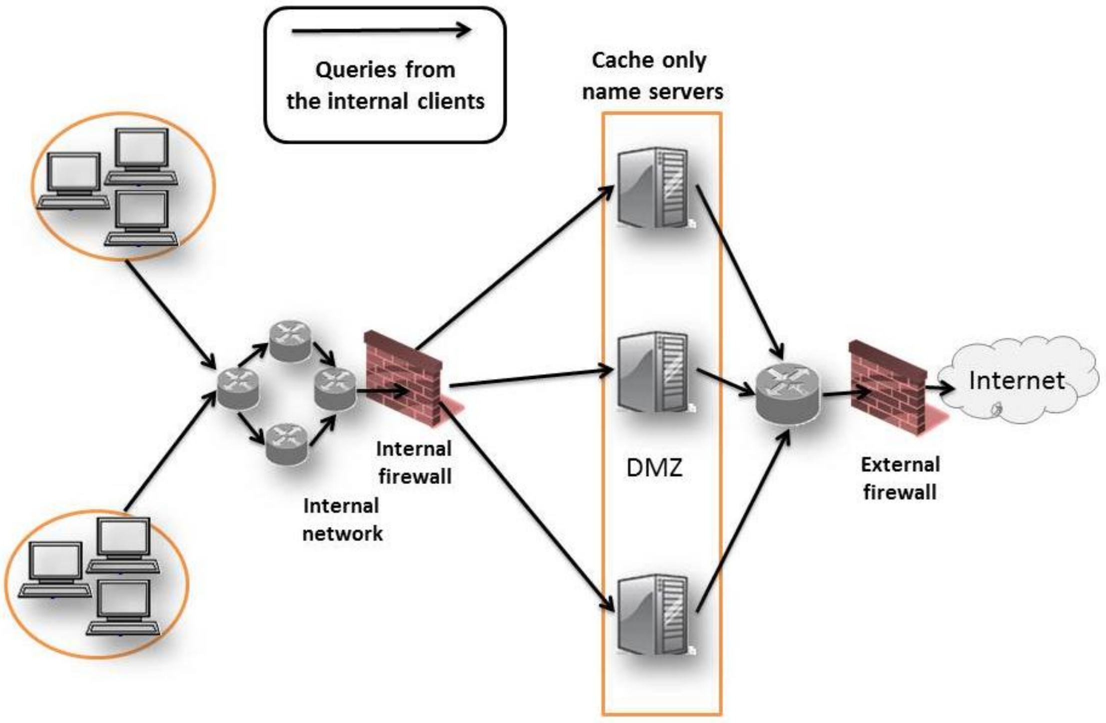


Figure 6.2 External DNS infrastructure – Cache-only name servers.

If the queried name does not belong to the internal domain, it will be forwarded by the proxy servers to the caching name servers. The query from the client is routed through the internal routers and firewalls and it is delivered to one of the caching name servers. If the caching name servers have no information regarding the queried name in their caches, they will follow the query by asking the authoritative servers in the Internet. After the iterative queries, the resolved name is sent back to the client and cached in these servers. If one of the caching name servers becomes unavailable, other servers are responsible for resolving. Caching name servers are also placed in the DMZ. When the routers and firewalls detect that a DNS packet is not following the DNS specification they will discard it.

6.1.3 Internal DNS

VOLVO IT is an enterprise company, due to this fact an internal DNS tree has been set up. This internal tree acts like the Internet DNS tree and distributes the administration and DNS database to the different zones. Recursive name servers, in the internal DNS infrastructure, are illustrated in Figure 6.3.

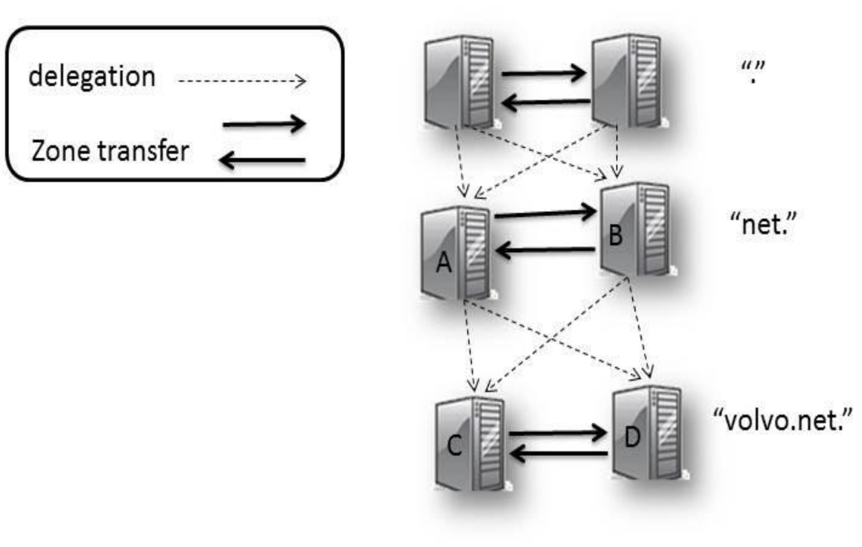


Figure 6.3 Internal DNS infrastructure – Recursive name servers.

The internal root gets queries from the internal resolvers. This server is the highest-level zone server in the internal DNS. To provide fault tolerance, every server has a backup server with the same configuration and zone file. Under root, different top-level domains exist for example. "net.". Information of the "net." zone is stored in two DNS recursive name servers, A and B, to provide fault tolerance. Two other servers, C and D, will host the information of the "volvo.net." zone. Delegations are established between the parent zone servers and the child zone servers. Internal resolvers start their queries from the root server. They will follow the DNS tree to find their answers.

6.2 Key generation and maintenance

Asymmetric key cryptography is the fundamental concept in DNSSEC. Many considerations regarding key generation and protection should be taken into account.

6.2.1 Key generation

Some factors play essential roles in the key generation namely key size, key duty and the quality of the random number generator.

Separating keys

We generate two different key pairs and separate their duties. The ZSK is used for signing all RRSets in the zone and the KSK is used for signing the DNSKEY RRSet [40]. This separation will reduce the complications of the key rollover.

The DS RR in the signed parent is authenticating a DNSKEY RR in the signed child zone. If one key is used at the same time for signing a zone and as a SEP to that zone any change in this key should be applied to the DS RR in the parent zone. As a result, if this key changes frequently, the DS and its covering signature must be updated. This will make the administration of the entire chain of trust more challenging. In our solution, the DS authenticates the KSK and the KSK will authenticate the ZSK.

The KSK has a larger size compared to the ZSK. Due to the size of the signed content, the KSK is used only for signing the DNSKEY RRSet, the probability of compromising the KSK is lower than the ZSK [35]. These two factors, large key size and smaller signed content, will increase the lifetime of the KSK. Therefore, there is no need to change the KSK and update the DS in the parent zone as frequently as the ZSK.

In the internal and external DNS, different KSKs and ZSKs will be used. If the key related to one zone become compromised, only that zone, which belongs to the internal or the external DNS, will be vulnerable to attacks.

Key size

The key size has a direct relation to the size of the zone it is signing [48]. If one zone has a large size or many zones are signed with one key, many signatures are generated with this key. If the attacker has enough resources for gathering large number of DNSSEC messages and performs cryptanalysis, he will be able to find the key. Therefore, it is more secure that for each zone we have a separate key to increase the resistance against attacks [54].

The key size affects the performance of the signing systems and security-aware resolvers for signature verification. We also need to consider if the size of the messages gets larger, the possibility that these packets is fragmented or dropped by the intermediate devices in the network gets higher.

Different sizes are required for ZSK and KSK. The size of the ZSK is better to be from 1024-bit to 2048-bit, using the RSA-SHA1 or the RSA-SHA-256 algorithm [29]. We recommend choosing 1024 key size, using RSA-SHA-256, for the ZSK to let the signing systems perform the zone signing and resolvers perform the validation faster.

A compromise of the KSK will endanger the entire zone. That is why we recommend larger key size for the KSK. If the KSK is compromised, the attacker can use the KSK for signing the DNSKEY RRSet of its own zone. We recommend the KSK with the size of size 2048-bit using RSA-SHA-256.

Random Number Generator (RNG)

We need to generate a large number of keys for signing different zones. For this reason, a strong Random Number Generator (RNG) is required. If the generated keys do not have a good source of entropy (randomness), then the DNSSEC infrastructure is open to DNS attackers who can guess the next key that is going to replace an old key. We recommend using a hardware random number generator in the external DNS. The external DNS has higher security priority because its traffic is sent through the Internet and the possibility that keys become compromised is much higher. The best choices for the hardware RNG are dedicated hardware security module (HSM) and Trusted Platform Module (TMP) [54].

6.2.2 Storing keys

When the private part of the key is not used, it should be kept off-line. We need to store the private keys in a secure location.

Keys in the external DNS

With HSM, we can generate and store keys in a safe location. They provide different security features ex. they can be tamper proof. Signing system and administrator can set up an SSL connection to these devices. Some HSMs also provide hardware acceleration for signing the zones.

We recommend the HSM for generating keys and hashes with higher quality and higher speed in the external DNS [54]. In the external DNS, signing system requests keys from the HSM and then it uses the KSK and ZSK for signing.

Keys in the internal DNS

In the internal DNS, some name servers are performing dynamic updates to update the IP addresses of their hosts. In this case, it is not possible to keep the private part of the ZSK offline because whenever the zone is updated it should be signed again. To decrease the impact of the possible key compromise from the name servers, which are performing dynamic updates, zone split [5] is used. RRs, which are required to be updated, are stored in a new subzone. For example, we have a zone named x.volvo.net we need to create a subzone named “dynamicupdate.x.volvo.net.”. Dynamically updated RRs are placed in this zone. The private key of the ZSK of the subzone is kept online, in the subzone, while the ZSK of the original zone can be kept offline.

In the internal DNS, we can generate and keep the keys in the name servers but with some consideration. When the key generating utility inside of the DNSSEC-enabled name server is executed, the shell of the operating system that invokes this utility should only be accessible by the administrator of that server. In addition, the permission level for accessing the zone file directory should be set to the administrator level.

Another solution is to use a software based security modules ex. SoftHSM [66]. This module is a software implementation of the crypto key storage that provides communication interface to the signing system based on the PKCS#11 (Crypto token interface standard) [71]. PKCS#11 is a standard that is used in the cryptographic key storages.

SoftHSM defines different layers of security for accessing the key generating utility. Whenever the signing system requests for accessing the SoftHSM, after accepting the request an empty slot is assigned to it and a token is generated. This token is used for setting up a session to the user management layer. After authenticating the session, a secret key is generated for each session. This secret key is used for accessing the crypto abstraction module for generating and fetching the asymmetric keys [17].

It is ideal to have hardware based HSM for internal name servers but these devices can become really expensive. The choice of the software or hardware based HSM is a trade-off between the cost and security. Hardware based HSMs provide more security (ex. tamper proof) while software based HSM are not costly.

6.3 Key rollover

In the DNSSEC specification, keys have no expiration date. However, it is not secure to use the keys for a long time. These keys should be replaced according to a fixed schedule. To rollover the keys we need new definition for keys and different methods to rollover the keys. The signing systems must support these requirements.

6.3.1 Rollover methods

When a new ZSK is used for signing the zone, other resolvers may still have signatures from the previous key in their caches. The resolvers still need to retrieve the old key from the signed zone until all signatures related to that key has expired [55].

In the chain of trust, whenever the KSK is changed, the parent should be informed. It should update the DS in the way that it will not cause any validation failure at the resolvers. Some delays exist from the time the KSK is created and activated, the DS become activated in the parent and it is cached in resolvers [55].

It is always possible to face misconfiguration in the chain of trust. By misconfiguration, we mean that the KSK is rolled over in the child zone but the parent is still pointing to the previous KSK. Misconfiguration is frequent while the procedure of the KSK rollover is done manually in many implementations. In this case, resolver cannot follow the chain of trust and use the DS to verify the KSK of the child zone. Therefore, the resolvers would not be able to validate the response. The failure in validation can also be due to the tampering of the DNS message by the attacker. Resolvers cannot differentiate between these two cases and as a result, the target zone is considered as insecure [64]. This issue will affect the availability of the DNSSEC enabled zones. We should define the KSK rollover procedures clearly in our solution to avoid misconfiguration in the chain of trust.

To deal with the stated problems different methods can be used [55]. Any changes in the ZSK and KSK should be performed in the way that the key rollover would not cause any validation failure. In our solution we use Pre-Published method for ZSK rollover and Double Signature method for KSK rollover.

ZSK Pre-Published method

In this method, two keys are created and included in the DNSKEY RRSet. Only one key is used for signing the zone. After some time, when the resolvers query the signed zone, they will get DNSKEY RRSet containing two keys. Whenever the time for rollover triggers, first key is retired, the third key is included in the DNSKEY RRSet and the zone is signed with the second key. Now resolvers can validate signatures using the first and second key.

ZSK Double signature method

In this method, two ZSK are created and included in the DNSKEY RRSet. Two keys are used for signing the zone. The resolvers have two keys and two signatures for every response. Whenever a key is rolled over, the resolvers can still verify the signatures with the other key. This method is not used in our solution because it increases the size of the zones and DNSSEC messages. This method is used for KSK rollover because it only signs the DNSKEY RRSet and KSK does not need to be rolled over as frequently as ZSK.

KSK Double signature method

In this method, a new KSK is included in the DNSKEY RRSet. The DNSKEY RRSet is signed with both keys. When the resolvers query the signed zone, they will get the new DNSKEY RRSet and they will cache it. After some time the old DNSKEY RRSet will expire from the cache of resolvers. Now, the DS in the parent can be changed. We must wait for some time until information regarding the new DS is propagated in the DNS network to remove the old key from the DNSKEY RRSet.

6.3.2 Key life cycle

For all keys a life cycle is defined which contains different states [55]. The state of each key is stored as metadata for each key and used for key rollover automation by the signing system. These states are illustrated in Figure 6.4.

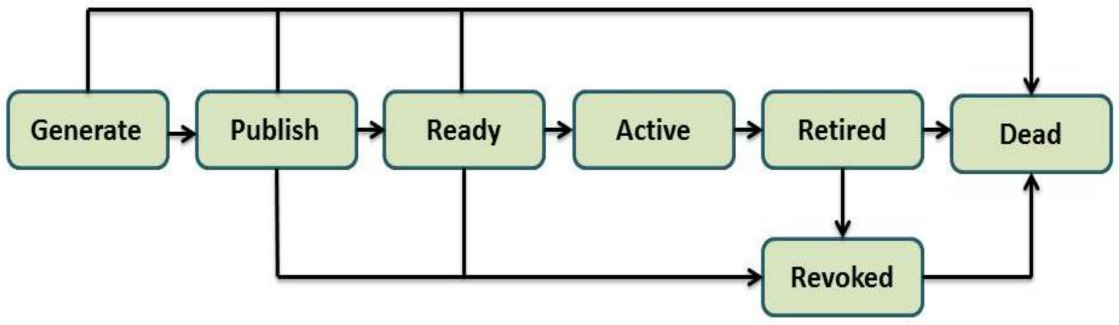


Figure 6.4 States of a Key [56].

Each key has seven states:

1. **Generated:** In this state, the key is generated but it is not used for signing. In the internal and external DNS, keys are stored in the HSMs.

2. **Published:** The DNSKEY RR is published in the zone but it is not used for signing. This key is not propagated yet through the DNS network.

3. **Ready:** Enough time has been passed from the publishing time of the key. The spare key is in this state.

4. **Active:** The private part of the key is used for signing. KSK is used for signing DNSKEY RRSet and ZSK is used for signing the zone.

5. **Retired:** In this state, another key is used for signing but this retired key still exists in the zone. Some resolvers may still have the signature related to this key in their caches. We need to wait for some time until these signatures have expired.

6. **Dead:** In this state, there is no information regarding signatures from this key in the resolvers. Now the key can be removed from the zone.

7. **Revoked:** In this state, the revoke bit of the key is set. This bit is used by resolvers, which support automatic update. This state is only used for the KSK.

Recommended active time for the KSK, using RSA-SHA-256 with the size of 2048 bits is from 12 to 24 months. The ZSK active time using RSA-SHA-256 with the size of 1024 bits is from one to three months [29].

6.3.3 Rollover procedure

The key rollover is better to be automatic to avoid any mistake. At the same time, it should be possible that the key rollover is performed manually for emergency key rollovers. Key rollover procedure is periodical so before using the second key for signing, the new key should be generated and become the spare of the second key for the next rollover.

ZSK rollover

In the ZSK rollover [35], [29] whenever a new key is included to the DNSKEY RRSet, the KSK is used to sign the DNSKEY RRSet. We must wait for a time greater than the TTL of the DNSKEY RRSet, until the old cached DNSKEY RRSet has expired from the cache of the resolvers and they request for the new DNSKEY RRSet. New key must not be used for signing the zone until enough time is passed and the new DNSKEY RRSet is cached in the resolvers.

In the initial phase, two keys are generated and the public part of each key is placed in the DNSKEY RRSet. One key is marked as active and the other is marked as published. The private part of the active key is used for signing the zone data while the private part of the published key is stored off-line. Whenever the active key is not used for signing, the private part of the active key must also become offline.

When the rollover time triggers, the active key is marked as retired and the published key becomes active. A new ZSK is generated and the public part of the key is placed in the DNSKEY RRSet and marked as published. The private part of this new key is kept offline. The new active key is used for signing the zone. After some time, in the order of a few days, the retired key is marked as dead. This key can then be removed from the zone at any time. ZSK rollover can become automated in many signing systems because it does not require interaction with the parent. The dead key must be disposed.

KSK rollover in the external DNS

When the advertising name servers become connected to the chain of trust, a procedure should be specified for changing the KSK. Whenever this key is changed, the information related to DS RR in the registrar, should also be updated to avoid any disconnection from the chain of trust.

Two methods can be used for updating the DS RR in the registrar. If the registrar supports handling of two DS for a domain, a new KSK is fetched from the HSM and the public part is added to the DNSKEY RRSet. The DNSKEY RRSet is signed with the new KSK. The new DS is published to the registrar. We must wait for some time, to ensure the DS is properly published in the registrar. We need to wait for one more TTL of the old DS, until information regarding the old DS has expired from the resolvers. The DNSKEY RR of the old key is removed from the DNSKEY RRSet. Now the old DS can be removed from the registrar [55].

If the registrar does not support handling of multiple DS for a domain, we must wait until the new DNSKEY RRSet is published from advertising name servers to the resolvers. After that, we can publish the DS of the new key in the registrar. The registrar will replace the new DS with the old one. We need to wait for the longest TTL time of the old and new DS. We wait until the old DS has expired from the cache of the resolvers and new DS is cached in the resolvers. After this time, we can remove the old KSK from the DNSKEY RRSet and sign the DNSKEY RRSet with the new key [55]. The dead key must be disposed.

KSK rollover in the internal DNS

The keys related to the internal root and zones that are locally signed must be anchored to the resolvers (see section 6.6.5). Whenever these keys are rolled over resolvers must also update their anchored keys. We will use automatic trust anchor update in the resolvers (see section 6.5.1).

The KSK rollover in the internal DNS will be as follows [35], [29].

In the authoritative name servers, the new KSK is generated and added to the DNSKEY RRSet. However, this key is not used for signing the DNSKEY RRSet. We will wait for the "Add hold down" time (see section 6.5.1) then set the revoke bit of the old KSK. Now, the DNSKEY RRSet is signed with both the old and the new KSKs. We will wait for the "Remove hold down" time. The revoked KSK is then removed from the DNSKEY RRSet and the DNSKEY is signed with the new active key. The dead key must be disposed.

Emergency key rollover

If the active key is accessed by an unauthorized user or if it is lost, due to failure in the key storage, we need to change the key immediately. This will interrupt the scheduled key rollover.

We are using the ZSK pre-published method. As a result, the existence of the published key in the DNSKEY RRSet is crucial for the emergency rollover.

As we are using the KSK double signature method, the DNSKEY RRSet is signed with two KSKs. In the case of becoming aware of the key compromise, this key and its signature over DNSKEY RRSet must be removed from the zone. The parent must be informed immediately to stop publishing the DS RR related to the compromised key for validation.

6.4 Zone Signing

Servers in the current internal and external DNS do not support DNSSEC. We need to enable DNSSEC on these name servers considering the following requirements. These requirements were determined to enhance the security and performance of the signing systems.

6.4.1 Choice of algorithm

In our solution, zone signers and resolvers must support all the cryptographic algorithms, including NSEC3, which are specified for DNSSEC.

RSA and DSA algorithms are two options for signing the zones. These two algorithms have roughly the same signature generating speed but in terms of signature verification, DSA is slower than RSA [37]. RSA is used to sign zones to decrease the operational load on the resolvers.

SHA-1 is the cryptographic algorithm that is used for a long time for creating digest. Hash of this algorithm has cryptanalysis issues [35]. For this reason, we will use SHA-256 or SHA-384 to create DS RR.

Right now, the Elliptic Curve DSA (ECDSA) algorithm is standardized for the DNSSEC and implemented [58] but it is not used by signers and validating resolvers. EC DSA decreases the size of the signed zone and the DNSSEC message. Curve P-256 has the same strength as SHA-256 with the key size of 3072 bits but its key size is 256 bits and its signatures are much shorter [58]. Using EC DSA algorithm will make DNSSEC servers less desirable for amplification attack because it decreases the size of the DNSSEC message considerably.

6.4.2 Security consideration regarding the NSEC3

In the signing systems, we will use NSEC3. NSEC3 solves the zone enumeration to some extent but this solution has its own defects. Salting adds no security if the salt value does not periodically change and attackers can perform the zone enumeration [61].

When the resolver computes the hash, to find out that the queried name falls in the interval of two hashed names in the proof of non-existence answer, it will use the salt value from the field with the same name in the returning NCES3 RR. This information is sent in clear text. The attacker can also see and use the salt for creating hashes from the entries of its rainbow table. The rainbow table is a huge list from the common owner names in the target zone. The attacker can create hashes and compare the returned hashes in the answer with the hashed entries of its rainbow table to find the owner names. This attack is performed offline and cannot be detected by controlling the incoming traffic to the name server.

In our solution, to decrease the threat of zone enumeration, we will change the salt value periodically. We must generate a new salt less than the time that is required for the attacker to create a large rainbow table. When the salt is changed, the prebuilt rainbow table of hashes that the attacker is using will become useless. The attacker should create the table from scratch to perform the zone enumeration [61]. Salt is created every time the signatures are refreshed. We also increase the number of iterations. At the same time, higher iteration will increase the computational load on signing systems and resolvers. We must choose the recommended number of iteration to decrease the computational load on the DNSSEC servers. The recommended number of iterations for the 1024-bit ZSK, using the RSA-SHA-256 algorithm, is 10 [65].

We must also monitor constantly the queries towards our DNS name servers to detect online brute force attacks for the zone enumeration. In the case of observing suspicious traffic, we must filter the source IP address that generates this traffic.

6.4.3 Jitter interval

When signatures have expired at different times, the load on the signing system for refreshing these signatures will be decreased. The jitter interval is the time interval where the signature expiration time is distributed randomly [35]. In the signing system, external DNS, the signature refreshing time and the jitter interval must be assigned for each zone independently to reduce the load on the signing system. In addition, the number of queries in a certain period of time will be reduced on the advertising name servers.

6.4.4 Signature validity period and TTL value

We will perform the key rollover procedure to replace keys. However, due to the caching mechanism in the DNS, DNSKEY RRSet that contains the previous key will remain in the cache of resolvers until the covering signature has expired. To reduce the threat of replay attack we will assign a shorter signature validity period.

Timing

If the KSK of a secured child zone become compromised, DS RR can be replayed as long as the RRSIG over DS RR has not expired [29]. We can minimize this threat by setting a shorter time for signature validity period of the DS RR. However, at the same time if the validity period is too short, in the case of configuration error in the child, administrators has a short time to fix the problem, DS signatures in resolvers will expire faster, and the child zone becomes insecure [35].

The validity period is recommended to be from few days to one week [29]. The DS signature must be refreshed a few days before the signature expiration. In the internal chain of trust, this time is the signature refreshing time of the parent zone. In the external DNS, we have no control on the validity period of the signature over DS RR. The signing policy of the parent determines the validity period of the signature over DS RR.

If the ZSK become compromised, the attacker can replay DNSKEY RRSet until KSK’s signature over the DNSKEY RRSet has expired. The validity period of the signature that covers DNSKEY RRSet is also set from few days to one week [29].

We also must assign smaller TTL value for DS and DNSKEY. In this case, data remains a shorter time in the cache of the resolvers. This mechanism will also increase the number of queries and load to the name servers but at the same time, updated information propagates faster through the DNS network.

6.5 Resolving

Servers in the current internal and external DNS do not support DNSSEC. We need to migrate these servers to the last version of implemented DNSSEC.

6.5.1 Automated trust anchor update

In the internal DNS, during the deployment of the DNSSEC, first zones are locally signed and then they are joined to the internal chain of trust. Due to the operational cost of deploying the DNSSEC to the network, instead of signing the entire internal tree, first, only some zones with higher security priority are signed locally.

KSKs of the internal root and locally signed zones change due to the key rollover. Resolvers which configured these keys as the trust anchors must become aware of these changes. In addition, caching resolvers must also become aware of key rollover of the Internet root. This procedure must become automated to avoid any complication in the administration and remove any probable misconfiguration. This mechanism is “Automated Update of Trust Anchors” [8].

When a new KSK becomes active, the signer will set the revoke bit for the old KSK and sign the DNSKEY RRSet with the new key. The revoke bit is the eighth bit in the Flags field of the DNSKEY. Resolvers use the “Finger Print”, digest of the old KSK, as a secure entry point to that signed zone. If the revocation bit is set, the fingerprint becomes different. From this change, resolvers become aware of the revocation. The resolver will update the old key by replacing it with the new active key.

Resolvers who support the automatic trust anchor update will use the “Add/Remove hold down Time” and “Active Refresh”. Add hold down time is used as a counter measure of any compromise of the trust anchor. If the active KSK become compromised, the attacker can insert its own trust anchor to DNSKEY RRSet and sign the new DNSKEY RRSet with the compromised key. The owner of the zone is acting the same when it becomes aware of the key compromise. The resolver is not able to determine which DNSKEY RRSet is original. Therefore, it adds hold down time. During this period, until the acceptance timer expires, it will not accept the new trust anchor. If before the hold down time, resolvers see any DNSKEY RRSet without the new trust anchor key, the new trust anchor is encountered as compromised and will not be accepted. Only using this mechanism will not avoid the trust anchor hijacking. Resolvers also use “Active Refresh” to perform the periodic lookup query for the DNSKEY RRSet and covering RRSIG to detect the changes in the trust anchor point. Add/Remove hold down time is considered 30 days and Active Refresh is at most 15 days [8].

Another security issues regarding the resolver is the anchored key that is stored in the zone file. If an attacker accesses the zone file, it can change the anchored key and set it as the key of its malicious name server. For this reason, servers must only be accessible by the administrators using secured connection. In addition, the permission level for accessing the zone file directory should be set to the administrator level.

6.5.1 Island of security

In the Internet, many zones exist that are signed but there is still no trust relation established between these zones and their parents. We should be able to validate responses from these islands of security in the Internet.

DNSSEC Look-aside Validation (DLV)

With the DNSSEC Look-aside Validation, domain owners can publish their trust anchors of their signed zones outside of the delegation chain [63]. Using this mechanism, the security aware resolvers validate responses from the signed zones, which are not connected to the chain of trust. In the DLV architecture, some domains are dedicated to publish SEP for the signed zones who are not their children. DLV RR is created from the KSK of the locally signed zone and then it is published to the DLV domain.

To validate responses from the islands of security, we need to add the validated entry of the DLV domains to the configuration file of the caching resolvers. In this way, first resolvers try to follow their iterative queries from the chain of trust starting from the root zone. If resolvers could not validate the signed zone through the chain of trust, they will follow their queries from the trusted DLV domain.

6.6 Modification to the current DNS infrastructure

To enable DNSSEC some modifications should be applied to the existing DNS infrastructure.

6.6.1 Preparing the DNS infrastructure

Processors and storage

Due to signature creation and verification, deploying DNSSEC to the existing servers will result in higher packet processing. This fact mandates adding more memory and computational resources to these servers. The storages of the name servers, who perform signing, should be increased. After signing, the size of the zone becomes larger. This increase is dependent to the signing algorithms and the type of the next secure RR.

Routers and Firewalls

The addition of the DNSSEC RRs will increase the size of the messages considerably. DNSSEC enabled name servers usually use UDP packets to avoid the overhead from the TCP connections. However, if there is no space for additional information in the UDP packet, first the EDNS0 mechanism is used and if the network devices do not support large UDP packets, a TCP connection will be established.

In the internal network, firewalls and routers are configured to allow UDP packets for DNS queries and TCP for zone transfer. All of the routing devices and firewall systems must be configured to allow TCP connections for DNSSEC transactions. In addition, routers and switches must provide more bandwidth for the DNSSEC traffic.

To migrate to a DNSSEC enabled infrastructure, routers and firewalls in the internal DNS must support large UDP DNS packets. Firewalls, which perform Application Layer Inception (ALI), are configured to reject large DNSSEC packets. Firewalls and routers must be configured to allow UDP DNSSEC packets to the maximum size of 4096 bytes [48].

TCP-based packets can be segmented if they exceed the Maximum Transmission Unit (MTU). There will always be a possibility that these fragmented packets are discarded by some intermediate devices. Therefore, DNS servers must use the Path MTU Discovery (PMTUD) for sending the DNSSEC responses to make sure that the large packets will not be discarded [48].

6.6.2 Signing systems

Signing systems in the internal DNS

In the internal DNS, each zone resides in its dedicated name server. These servers sign their zones using their generated ZSK keys. Whenever the ZSK’s active time ends, the automated procedure is used for the key rollover and the zone is signed with the new key. The secondary name server, with the same configuration as the primary server, is used as backup. It is always updated with the last version of the signed zone from the primary server. Both name servers store their own keys inside the installed HSM (software based or hardware base) module. Whenever the primary name server becomes faulty, the secondary name sever takes its place and serves the signed RRSets until the signature refreshing time. At this time, the secondary name server signs the zone with a new key.

Signing system in the external DNS

In the external DNS, due to its unique structure and the large number of zones, one module must be assigned independently as a signing system. This signing system must perform the key roll over and zone signing. It also should keep track of the state of each key, which is stored in the HSM, for the key rollover procedure. Information regarding the key role, key algorithm, key size, key timing, salt and signature refreshing intervals etc. must be configurable for each zone independently. The existing solution, which can perform the zone signing integrated with key rollover for large number of zones is OpenDNSSEC [62]. This open source signing system can interact with HSM for creating keys and zone signing and benefits from hardware acceleration of the HSM.

6.6.3 Synchronization

Absolute time was introduced by the DNSSEC. Each RRSet is assigned a signature, which contains an expiration time. This timing value is taken from the current time of the signing system. The signing system and the resolver who validates the signature should have the same understanding of the current time. This fact mandates that all servers in the DNSSEC infrastructure become synchronised with a reliable source of time. The DNSSEC aware name servers and resolvers will be synchronised with the Coordinated Universal Time (UTC) time using the NTP servers.

However, an attacker can perform the attack to the NTP clients, the signing systems and DNSSEC aware resolvers, and misconfigure their clocks. For example, if the attacker changes the clock of the signing systems backwards these servers will serve expired signatures to the validating resolver who has different understanding of the current time from the signing system. For this reason, NTP servers and clients must use the “Shared secret” or “Autokey” (asymmetric key encryption) [70] for the authentication and integrity checking of the NTP data packets.

6.6.4 Monitoring the traffic for the DDoS attacks

As we discussed in the previous section the threat of amplification attacks is magnified when we are deploying DNSSEC to our infrastructure. The attacker can trick our advertising name servers to send queries to the victim DNS server and overwhelm the communication line of the victim or he can set the IP address of our caching server as the target of his amplification attack.

Some DNS servers are designed to drop unmatched responses but this is not the case for all the name servers. Generally, we need to monitor the traffic to and from our name servers to detect any sudden increase. Large number of packets, which are received in a short period from the same source, can be a good alert for the amplification attack. One counter measure is to filter these addresses. However, this method will lead to filtering many legitimate servers while the attacker can change the source address of the generated traffic from the Bot-Net. The best solution is to drop packets or delay this type of traffic [60].

6.6.5 External DNS

In the external DNS, before publishing public zones they should first be signed.

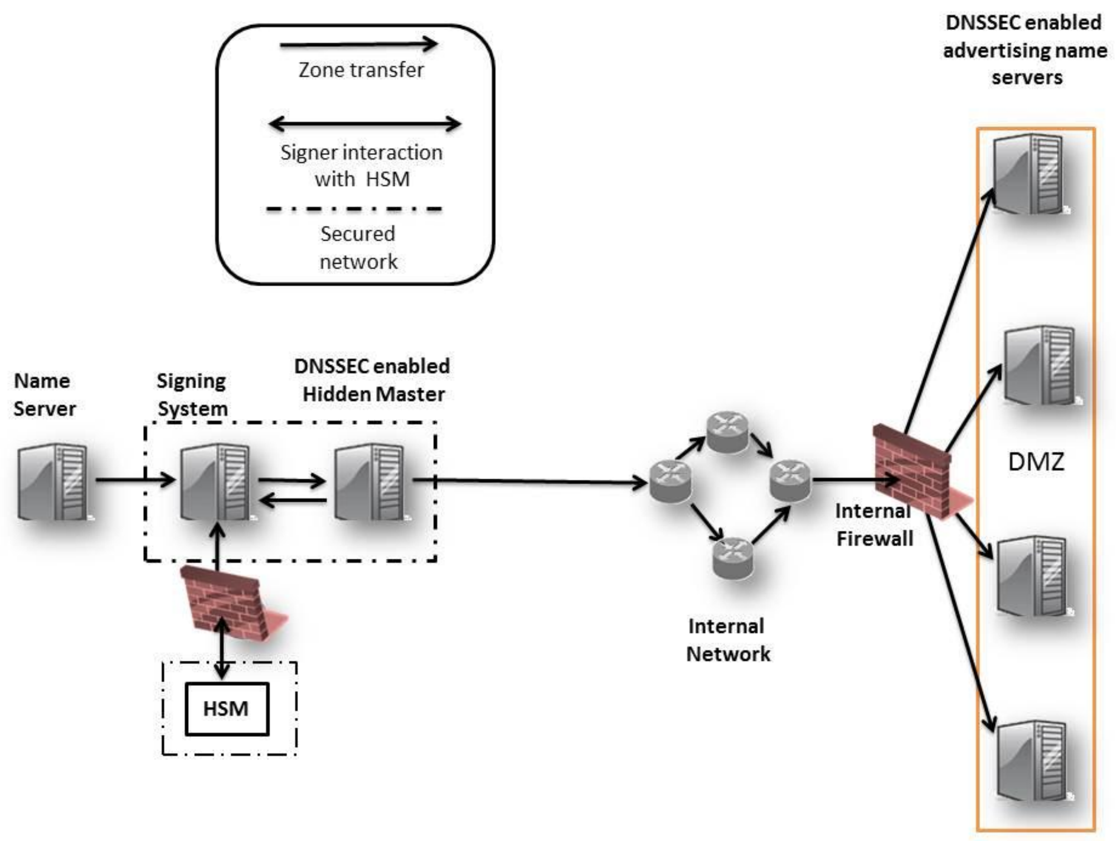

Zone transfer
Signer interaction with HSM
Secured network
DNSSEC enabled advertising name servers
Name Server
Signing System
DNSSEC enabled Hidden Master
HSM
Internal Network
Internal Firewall
DMZ

Figure 6.5 External DNS infrastructure modifications

As illustrated in Figure 6.5, a module is added separately as a signer to this infrastructure. Whenever a zone is updated in the name server, secure zone transfer is used to transfer the zone to the signer. TSIG will make the data transition secure and the attacker cannot compromise the integrity of the data.

The signer interacts with the HSM and HSM will perform the signing. The HSM is located in a secure network behind a firewall. Only the signer and administrator have access to the HSM using an SSL connection. We also need a backup mechanism if the HSM became faulty. This backup HSM will become active and take the place of the faulty device.

After signing, the signed zones are stored in the DNSSEC enabled hidden master. Due to software failure or interrupt in the connection with HSM, signed zones may contain some errors. In this case, the advertising name server will publish erroneous data to the resolvers. Therefore, after signing zones, they should be checked by the signing system. In the OpenDNSSEC a separated module, which is called policy auditor, is responsible for checking the signed zone. If the zone is signed correctly, then it will be transferred to the hidden master. Otherwise, the zone should be signed again. The hidden master will transfer zones to the advertising name servers using TSIG.

In the existing infrastructure, if any failure happens in the hidden master, the secondary name servers have a copy of the zone and reply to the queries. The external DNS design tolerates the failure of the hidden master. After deploying DNSSEC, if the signing system becomes unavailable the advertising name servers serve users until the signature of RRSet will expire. For this reason, a backup signing system should be considered to take the place of the faulty signing system.

Right now load balancing is performed using several advertising name servers in the external DNS. After deploying DNSSEC, all these servers should serve consistent information to the resolvers in the Internet. For this reason, all servers should have the last version of each signed zone. After any change in the signed zone, the hidden master will inform the advertising name servers using NOTIFY. Advertising name servers must perform periodical queries to check the SOA serial number of the zone in the hidden master with their own zone and become aware of any changes [29].

6.6.6 Internal DNS

As illustrated in Figure 6.6, in the internal DNS infrastructure DNSSEC aware validating resolvers are added to the internal DNS.

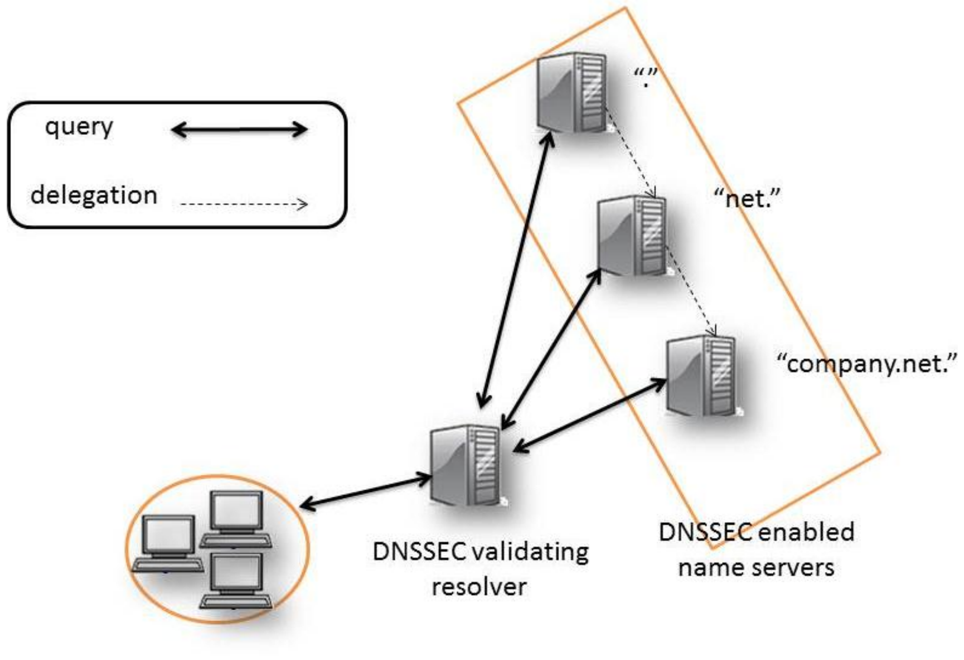


Figure 6.6 Internal DNS infrastructure modifications

Clients from different sites are non-validating stub-resolvers. Therefore, the signature validation duty will be assigned to the DNSSEC validating resolvers. These validating resolvers are located between the clients from the different sites and name servers in the internal DNS. These resolvers are also responsible for sending queries of the clients, for the Internet domain, to the caching resolvers. The DNSSEC validating resolver set the CD bit in the DNS header to inform the caching servers to pass the RRSets with their signatures. In this way, RRSets and their covering signatures will be sent to the DNSSEC aware resolvers for validation. The reason for this configuration is that DNSSEC is using the same UDP DNS packets for the DNSSEC queries and responses. There is no integrity checking mechanism to check whether the man-in-the-middle attacker set the AD bit in the header and forged the content of the DNS answer.

From the other side, if clients trust the result of validation from the validating resolver only by checking the AD bit, the attacker can modify the DNS answer and poison the caches of the clients. For this reason, we will establish a trust relation between clients and resolvers using the IPSec connection.

Chapter 7 Discussion

The security issues related to the DNS endangers any application or service that relies on this service for its operation. There is no doubt that the DNS must become secure as soon as possible to decrease the hazard of DNS attacks. DNSSEC is an Internet scale solution, which was proposed to tackle DNS vulnerabilities. However, the security of this solution is also dependent on many factors.

In this thesis, we explained many security considerations related to all parts of the network and devices that are involved in the deployment of DNSSEC. In our solution, we covered many security concerns related to the set up and maintenance of the DNSSEC and proposed the best practice solution for enabling DNSSEC.

Security of the keys

Keys play essential roles in DNSSEC. If a key is compromised, DNSSEC cannot protect DNS. To keep the keys secure some considerations should be taken into account for the key generation, namely the key size, key duty and the quality of the random number generator.

In addition, we need to keep the private part of the KSKs and ZSKs out of the reach of the non-privileged users. During the KSK rollover, when the key is handed in to the administrator of the parent zone, a secured channel must be used.

However, the security of the keys is not limited to the topics that were discussed in our solution and other security consideration should not be neglected. If someone had access to the specific HSM or signing system and now is not responsible anymore, his access to these devices must be prohibited. Validating resolvers should also be protected from unauthorized access because the attacker can misconfigure the anchored key and redirect resolvers to its own malicious servers.

Physical hardening of the signing system and HSM is another important consideration. These devices must be placed in locations with the highest security protection and only be accessible by administrators who have sufficient privileges.

Zone Signing

The DNSSEC solution has been used for a while so it is known by the attackers and they discovered lots of vulnerabilities related to this solution. In our solution, these vulnerabilities were addressed and the available solutions were discussed. We explained which algorithm has weaknesses and should not be used. We also described how we can resist against zone enumeration if the zone is signed with NSEC3 and how to choose the right TTL value to decrease the possibility of replay attacks while not losing the availability.

As we mentioned, the current algorithm (RSA) increases the size of the DNSSEC messages. It is also less secure, using the same key size, comparing to the EC DSA algorithm. The process of migration from the current RSA algorithm to the EC DSA will add lots of complexity and workload to the administrators. Therefore, it is important to consider this fact before enabling DNSSEC in the company’s environment, as this algorithm will be used in the near future [29].

DNSSEC is a solution for a distributed system. Due to this property, we need to interact with the other autonomous systems in the Internet. This property will limit us in choosing the best algorithm for signing our zones. We will not be able to use the EC DSA algorithm, which has been implemented, until it is supported by the other autonomous systems. Signatures from our name servers are not understandable for the DNSSEC aware resolvers of these autonomous systems and our zone is not secure from their point of view.

Automation

The process of the key rollover is a complex task. It is not possible for the administrators of the large-scale network to take care of the state of each key and rollover the key. This process, as it was mentioned in our solution, should become completely automated.

In our external DNS, we are interacting with the registrar for the KSK rollover. Therefore, the process of the automation cannot be applied for all the keys. If the registrar does not automate this process, the KSK rollover will be performed manually and may lead to misconfiguration due to human error.

Monitoring and Maintenance

DNSSEC introduces new definitions, problems and attacks to the DNS infrastructure. Before migrating to the DNSSEC aware environment, the administrators must have sufficient knowledge about this solution. They should also be aware of how to troubleshoot the problems related to both the signing system and the validating resolvers. They need to monitor the status of all signed zone and their chain of trust constantly.

One example of DNSSEC problems can be the inconsistency in the state of the advertising name servers. One server can lose its communication to the hidden master and serve the expired signatures to the clients. In this case, administrators must detect the problem before resolvers in the Internet receive expired signatures.

DNSSEC provide no protection against DoS attacks and it introduces a new type of DDoS attack. Administrators must always monitor the incoming traffic to the advertising name servers from the Internet and filter any malicious traffic, which are increasing and may saturate the communication line or cause failure in the name servers. This approach, filtering the traffic, has a negative side effect. We may filter the legitimate servers while these servers are just responding to the traffic from the Bot-Net.

We may also become the target of the amplification attack. Administrator should drop the packets that are increasing from a certain level from a specific address to avoid any harm to the other name servers or to become filtered by these servers.

In the current DNSSEC implementation lack of monitoring tools, which can provide automatic notification to the administrators about the DNSSEC related issues, made the administration of the DNSSEC environment much harder.

The defined parameters for timing and key size are relative and are not absolute values. Attackers always achieve higher performance devices for their cryptanalysis, zone enumeration and other attacks. The process of protecting DNS is a continuous task and these values must be updated.

Considerations from a company perspective

A security solution must not only provide integrity and confidentiality of the service, which it is securing, but also should guarantee the availability of that service. During our investigations for finding the best practice solution for deploying DNSSEC to the DNS infrastructure of the Volvo IT, we observed that a major concern for the DNS team was to maintain the availability of the DNS service. The customers cannot tolerate any failure in the DNS service. This fact must be considered before moving to a security enabled DNS environment. The availability issues related to the DNSSEC is related to its operational part. DNSSEC requires maintenance of the signed zone in the way that the signatures are always kept fresh. If the servers serve the expired signatures, the signed zone will not be available to the clients. Another availability issue is due to the key rollover, as it was discussed before, the KSK rollover in most cases is performed manually. If the administrator forgot to inform the parent zone regarding the new KSK and sign the DNSKEY RRSet with this key, the child zone becomes unsecure and the zone will not be available for the users.

From the VOLVO IT point of view, DNSSEC is a good solution to secure the DNS but they need to pay much attention to the way they are implementing it. They need to consider the security of the HSMs and the signing servers. Another consideration is to become more familiar with the key rollover on regular basis. The key rollover must be defined clearly for the administrators.

Based on the severity of the DNS security issues, VOLVO IT has decided to proceed with this solution in a systematic procedure. The first step is to deploy DNSSEC on the external DNS of their infrastructure in order to secure the main domains ex. Volvo.com, Mack.com or Renault-trucks.com. After a couple of months, they will decide to roll out the DNSSEC to all domains that are advertised to the Internet considering the load on the DNS servers and required manpower which will be measured in the first step. When it comes to the use of DNSSEC in the internal DNS, it is not decided yet and they will wait to see the outcomes of enabling the DNSSEC on the external DNS.

Before enabling DNSSEC entirely inside the VOLVO IT, the MTU size must be adjusted in the existing infrastructure. VOLVO IT needs to think about all aspects of the changes not only the side effect on servers but also on all devices, which allow DNS traffic such as routers and firewalls.

Chapter 8 Conclusion

In this thesis, the following was discussed and analyzed:

- The design of the DNS was analyzed with particular detail to show what made this system unsecure and vulnerable to many attacks. We discussed about the existing security threats towards the DNS system.

- We also described the objectives of the DNSSEC solution and explained how this solution protects DNS. We determined what security features are provided by DNSSEC and showed the security domain that DNSSEC covers. We also mentioned that DNSSEC is not covering all of the DNS security issues.

- We analyzed the current implementation of DNSSEC. We have gone through the common attacks and availability issue related to the DNSSEC implementation. This analysis helped us to become familiar with the DNSSEC vulnerabilities, which make the DNS infrastructure target of DNSSEC related attacks after deploying this solution.

- DNSSEC is an extended solution. To enable this solution, many considerations should be taken into account, which adds lots of complexity for administrators. We covered the deployment of DNSSEC within a particular organization (VOLVO IT). We discussed how to set up and maintain DNSSEC within this organization. We explained the available solutions for some of the DNSSEC related security issues and used them as countermeasures in our solution. We used the best practice solutions to decrease the complexity of deployment of DNSSEC.

- We discussed challenges the company will face to implement this solution. Moreover, we talked about their plan for deploying DNSSEC. We also went through the delimitations of the DNSSEC solution.

Chapter 9 References

[1] P. Mockapetris, "Domain Names - Implementation and specification", IETF-Network Working Group, November 1987.

[2] Wikipedia, “Domain Name System”.
[Online]. http://en.wikipedia.org/wiki/Domain_Name_System

[3] NLnet Labs, “A short history of DNSSEC”, (2012,May) NLnet Labs. [Online]. http://www.nlnetlabs.nl/projects/dnssec/history.html

[4] D. Eastlake and C. Kaufman, "Domain Name System Security Extensions", IETF-Network Working Group, January 1997.

[5] H. Yang ,E. Osterweil ,D. Massey, S. Lu and L. Zhang , " Deploying Cryptography in Internet-Scale Systems:A Case Study on DNSSEC ".

[6] B. Laurie ,G. Sisson, R. Arends and D. Blacka, " DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", IETF-Network Working Group, March 2008.

[7] M. Gieben and M. Mekking, " Authenticated Denial of Existence in the DNS", January 2012.

[8] M. StJohns, " Automated Updates of DNS Security (DNSSEC) Trust Anchors", IETF-Network Working Group, September 2007.

[9] Wikipedia, “Alternative DNS root”.
[Online]. http://en.wikipedia.org/wiki/Alternative_DNS_root

[10] J. Anderson, "A brief history of DNSSEC and Neustar", (2011, September) Neustar UltraDNS. [Online]. http://blog.ultradns.com/dns-matters/a-brief-history-of-dnssec-and-neustar/

[11] L. Green, "DNS Spoofing by The Man In The Middle", SANS Institute InfoSec Reading Room-GSEC Practical Assignment, version 1.4c-option 1, 2005.

[12] J. Bau and J. Mitchell, "A Security Evaluation of DNSSEC with NSEC3", Stanford University Stanford, CA, March 2010.

[13] C. Sander, "Understanding Man-In-The-Middle Attacks", Space and Naval Warfare Systems Command (SPAWAR), March 2010.

[14] Sainstitute, "Attacking the DNS protocol", Security Associates Institute, October 2003.

[15] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)", IETF-Network Working Group, August 2004.

[16] Steven M. Bellovin, ” Using the Domain Name System for System Break-Ins” [Online]. http://www.scs.stanford.edu/nyu/05sp/sched/readings/dns.pdf

[17] OpenDNSSEC, SoftHSM Design, ”Initialising a token” [Online].

https://wiki.opendnssec.org/display/SoftHSM/Sequence1InitToken

[18] C. L. Schuba, "Addressing Weaknesses in the Domain Name System Protocol", Purde University, Augest 1993.

[19] T. Olzak, "DNS Cache Poisoning: Definition and Prevention", March 2006.

[20] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol Modifications for the DNS Security Extensions", IETF-Network Working Group, March 2005.

[21] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for the DNS Security Extensions", IETF-Network Working Group, March 2005.

[22] S. Cheun, "Denial of Service against the Domain Name System: Threats and Countermeasure", Department of Homeland Security, vol. CSL Technical Report SRI-CSL-05-02, July 2005.

[23] P. Mockapetris, "Domain names - concepts and facilities", IETF-Network Working Group, November 1987.

[24] S. Thomson, H. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE) ", IETF-Network Working Group, April 1997.

[25] B. Wellington, "Secure Domain Name System (DNS) Dynamic Update", IETF-Network Working Group, November 2000.

[26] eu Insights, "Overview of DNSSEC deployment worldwide",.eu Insights, October 2010.

[27] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements", IETF-Network Working Group, March 2005.

[28] T. Jonson, "Public-key Cryptography: PGP, SSL, and SSH", Global Information Assurance Certification Paper-SANS Institute, 2002.

[29] R. Chandramouli and S. Rose, "Secure Domain Name System (DNS) Deployment Guide, Recommendations of the National Institute, vol. Special Publication 800-81r1 , April 2010.

[30] R. Gieben, "The parent-child and keyholder-keysigner relations and their communication in DNSSEC", NLnet Labs, 2000.

[31] ICAAN, root signing status, ICANN and VeriSign, 2009-2012.

[32] Free Software Foundation, "The GNU Privacy Handbook", Free Software Foundation, 1999.

[33] Network Associates Inc. and its Affiliated Companies, "An Introduction to Cryptography", United States of America: Network Associates, 1990-1999.

[34] M. Gieben, "DNSSEC: The Protocol, Deployment, and a Bit of Development" , USA: The Internet Protocol Journal, June 2004.

[35] O. Kolkman, W. Mekking and R. Gieben, "DNSSEC Operational Practices, Version 2 draft-ietf-dnsop-rfc4641bis-13", DNSOP, September 2012.

[36] D. EastLake, "DSA KEYs and SIGs in the Domain Name System (DNS)", IETF-Network Working Group, March 1999.

[37] D. Eastlake 3rd, "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)", IETF-Network Working Group, May 2001.

[38] A. Clegg, "DNSSEC Exposed Deploying DNSSEC in Real Life", Internet Systems Consortium, July 2010.

[39] D. Eastlake, "DNS Security Operational Considerations", IETF-Network Working Group, March 1999.

[40] O. Kolkman, J. Schlyter, and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR)-Secure Entry Point (SEP) Flag", IETF-Network Working Group, April 2004.

[41] O. Gudmundsson, "Delegation Signer (DS) Resource Record (RR)", IETF-Network Working Group, December 2003.

[42] J. Ed. Schlyter, "DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format", IETF-Network Working Group, August 2004.

[43] J. Jansen, "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", IETF-Network Working Group, 2009.

[44] D. Eastlake, "RSA/MD5 KEYs and SIGs in the Domain Name System (DNS)", IETF-Network Working Group, March 1999.

[45] R. Elz and R. Bush " Clarifications to the DNS Specification ", IETF-Network Working Group, July 1997.

[46] M. Larson, "DNSSEC Overview" , Verisign Labs, February 2012.

[47] D. Conrad, "Indicating Resolver Support of DNSSEC", IETF-Network Working Group, December 2001.

[48] Cisco Security Intelligence Operations, "Preparing for DNSSEC: Best Practices, Recommendations, and Tips for Successful Implementation", Cisco Security Intelligence Operations.

[49] P. Vixie, "Extension Mechanisms for DNS (EDNS0)", IETF-Network Working Group, August 1999.

[50] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", December 1998.

[51] O. Gudmundsson, "DNSSEC and IPv6 A6 aware server resolver message size requirements", IETF-Network Working Group, December 2001.

[52] B. Wellington and O. Gudmundsson, "Redefinition of DNS Authenticated Data (AD) bit", IETF-Network Working Group, November 2003.

[53] G. Huston, "DNSSEC – A Review", The ISP Column, June 2010.

[54] "DNSSEC Key Management", Sarion Systems Research, January 2011.

[55] S. Morris, J. Ihren and J. Dickinson " draft-ietf-dnsop-dnssec-key-timing-03", Internet Engineering Task Force, January 2012.

[56] Opendnssec, "Key States", (2011,November). [Online].

https://wiki.opendnssec.org/display/DOCS/Key+States#KeyStates-StandbyKSKstates

[57] Wikipedia, "Registrar". [Online].

http://en.wikipedia.org/wiki/Domain_name_registrar

[58] P. Hoffman and W.C.A. Wijngaards, " Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", Internet Engineering Task Force (IETF), April 2012.

[59] D. Piscitello," Anatomy of a DNS DDoS Amplification Attack", [Online].

http://www.watchguard.com/infocenter/editorial/41649.asp

[60] G. Lindsay, " DNSSEC and DNS Amplification Attacks", [Online].

http://technet.microsoft.com/en-us/security/hh972393.aspx

[61] C. Strotman, " Take your DNSSEC with a grain of salt", [Online].

http://info.menandmice.com/blog/bid/73645/Take-your-DNSSEC-with-a-grain-of-salt

[62] Opendnssec, " Opendnssec", [Online].

https://wiki.opendnssec.org/display/DOCS/OpenDNSSEC+Documentation+Home

[63] S. Weiler, "DNSSEC Lookaside Validation (DLV)", IETF-Network Working Group, November 2007.

[64] Comcast– DNS Engineering, “Analysis of DNSSEC Validation Failure”, January 2012.

[65] Japan Registry Services, “DNSSEC Technology Experiment Report Verification of Functionality and Performance”, September 2010.

[66] Opendnssec,”SoftHSM”, [Online].
https://wiki.opendnssec.org/display/SoftHSMDOCS/SoftHSM+Documentation+Home#SoftHSMDocumentationHome-Keymanagement

[67] US-CERT,” Understanding Denial-of-Service Attacks”, [Online].

http://www.us-cert.gov/cas/tips/ST04-015.html

[68] ICANN,” ICANN-Accredited Registrars”, [Online].

http://www.icann.org/registrar-reports/accredited-list.html

[69] Microsoft,” Domain Name System Overview”[Online].

http://technet.microsoft.com/en-us/library/bb727007.aspx

[70] “Understanding and using the Network Time Protocol” [Online].

http://www.ntp.org/ntpfaq/NTP-s-algo-crypt.htm

[71] RSA Laboratories, “PKCS #11: Cryptographic Token Interface Standard” [Online].

http://www.rsa.com/rsalabs/node.asp?id=2133