



www.euro6ix.net

| Title: | Document Version: |
|---|---|
| **Technical Report TR4.1A.5**<br>**DNSSEC** | 1.6 |

| Project Number: | Project Acronym: | Project Title: |
|---|---|---|
| IST-2001-32161 | Euro6IX | European IPv6 Internet Exchanges Backbone |

| Contractual Delivery Date: | Actual Delivery Date: | Deliverable Type* - Security**: |
|---|---|---|
| 31/12/2002 | 25/02/2003 | R – PU |

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

| Responsible and Editor/Author: | Organization: | Contributing WP: |
|---|---|---|
| David Fernández | UPM | WP4 |

**Authors (organizations) in alphabetical order:**

Alvaro Vives (Consulintel), Félix Jesús García (UMU), Antonio F. Gómez-Skarmeta (UMU), Juan Luis Fernández (UPM).

**Abstract:**

This annex to WP4 deliverable D4.1A summarizes the activities carried out inside the DNSSEC subactivity during the second semester of the first year. Work invested has focused on getting experience on the configuration and management of DNS security extensions using BIND and to set-up the necessary infrastructure to run a limited DNSSEC pilot service over Euro6IX testbed.

**Keywords:**

BIND, DNSSEC, DNS Security Extensions, PKI, Transaction Signatures, TSIG.

© Euro6IX Consortium

# Revision History

| Revision | Date | Description | Author (Organization) |
|---|---|---|---|
| v1.0 | 21/11/2002 | Document creation | Antonio Skarmeta (UMU)<br>Félix Jesús García (UMU) |
| v1.1 | 12/1272002 | Contribution from Consulintel | Alvaro Vives (Consulintel) |
| v1.2 | 17/12/2002 | Contribution from Consulintel updated | Alvaro Vives (Consulintel) |
| v1.3 | 21/12/2002 | Contribution from UPM | David Fernández (UPM)<br>Juan Luis Fernández (UPM) |
| v1.4 | 26/12/2002 | Integration of UPM and Consulintel contributions | David Fernández (UPM) |
| v1.5 | 30/12/2002 | Final revision and corrections | David Fernández (UPM) |
| v1.6 | 25/02/2003 | Logos added an PDF generated | Jordi Palet (Consulintel) |

# Executive Summary

This annex to WP4 deliverable D4.1A summarizes the activities carried out inside the DNSSEC subactivity during the second semester of the first year. Work invested has focused on getting experience on the configuration and management of DNS security extensions using BIND and to set-up the necessary infrastructure to run a limited pilot service over Euro6IX testbed.

The subactivity has been organized around three main activities:

- Development of a DNS emulation environment, to allow the creation inside one computer of complex DNS scenarios made of complete name server hierarchies. Apart from its specific use as a tool for testing DNSSEC scenarios and configuration alternatives, the preliminary version of the tool has shown to be very useful as a DNS training tool, to easily test DNS example configurations in isolated environments, speeding up the DNS learning process, or as a tool to help the operation of DNS servers.

- Local tests made by each participating partner inside its testbed network, in order to acquire the basic know-how and to create the basic infrastructure needed for a DNSSEC service. These tests have allowed the creation of basic "cookbooks" about how to configure and manage DNSSEC.

- Inter-partner tests over the Euro6IX network. Based on the know-how acquired in the local and emulated tests, a limited pilot service between the three partners involved in this subactivity has been created. Although only basic tests have been carried out over it, this pilot service will be used during the second year to do deeper tests, concentrating on service management procedures (key synchronization and distribution, foreseen and unforeseen key changes, etc).

The long-term objective of this subactivity is the deployment of a secure DNS service over the Euro6IX network. For that purpose, work will continue during the second year, and a limited service will be deployed, in order to mature the management procedures and generate the documentation and recommendations necessary for extending the service to the whole network under the scope of WP3.

The document is organized as follows. After a brief introduction to the document in Section 1, a summary of the security extensions to DNS proposed inside IETF is made in Section 2. Section 3 presents the implementation selected for the tests and the reasons because it is chosen. Section 4 describes in detail all the activities carried out around DNSSEC inside Euro6IX. Finally, section 5 comes to some conclusions and Section 6 presents some ideas about future activities.

# Table of Contents

# Table of Figures

# 1. INTRODUCTION

The Domain Name System (DNS) is a hierarchically distributed database that allows convenient storing and retrieving of resource records fundamental to Internet operations, such as translating between human readable host names and Internet Protocol (IP) addresses. In view of the importance of the information served by DNS, there is a strong demand for securing communication within the DNS system. The current (insecure) DNS does not prevent attackers from modifying or updating DNS messages. Users accessing hosts on the Internet trust in the correct translation of host names to IP addresses by the DNS system. A typical attack, referred to as DNS spoofing, allows an attacker to manipulate DNS answers on their way to the users. If an attacker makes changes in the DNS tables of a single server, those changes will propagate across the Internet.

Being the DNS service so critic for Internet, there is at present an urgent need to secure it by providing methods to guaranty the authenticity of the information it provides. Moreover, the possibility to use a secure DNS service to publish digital certificates, and, as a consequence, make public-key based applications like e-commerce a reality, makes the task of securing DNS even more attractive.

DNS Security (DNSSEC) technology is made of a set of extensions to the Domain Name System (DNS) protocol that provide data integrity and data origin authentication to security aware resolvers and applications, mainly through the use of public-key cryptography. Confidentiality is not required as the information stored in the DNS database is supposedly public.

# 2. DNSSEC ROADMAP IN THE IETF

## 2.1 Overview of DNS

DNS [1] is hierarchical and distributed database. The data is divided hierarchically into a tree with a root where each node in the tree represents a partition of the overall database, called a domain. Each domain can be divided into subdomains. The administration of the names is distributed, allowing local control of the segments of the global database. A parent node in the DNS tree knows of the servers that manages the DNS segments of its child nodes. The server programs in the DNS are called name servers.

A zone is a contiguous part of the domain tree for which a domain server has complete information and over which it has authority. The data contained in a zone file is composed of entries called Resource Records (RRs). Records can be of different types, but the address type is the most common one.

A master name server is the one name server where a zone is administered. One or several slave name servers know the complete information of a zone. The slave name server gets the zone data from the master name server by a zone transfer. A name server might be able keep a cache of non-local DNS data. As DNS data may change over time, there is a timeout mechanism causing the data to be discarded eventually.

A DNS client program is called a resolver. A resolver can perform DNS queries. A query is a message sent to a name server to acquire some DNS data. There are two kinds of resolvers: real resolvers and stub resolvers.

- A **stub resolver** is basically a library that needs to be installed on every host that wants to access the DNS database. Every time a query needs to be sent, functions of this library are called and the process of retrieving the desired information is run. Specially, the stub resolver sends a recursive query to a resolver which will reply with the information needed.

- A **resolver** is generally located on a DNS server and serves a group of stub resolvers. When a recursive query is received, the resolver usually sends an iterative query to one of the root DNS servers serving the root domain. Iterative queries allow a DNS server, which does not have the requested mapping, to indicate the next server in the chain which is “closer" to the authoritative server for those queries.

In the example the following figure, the local name server (local resolver) receives a recursive query for the IP address of the web server www.euro6ix.org from a client host. The local resolver then sends an iterative query to a root DNS server, which returns the IP address of the DNS server authoritative for the .org zone. Then the local resolver will query the name server authoritative for euro6ix.org which will return the IP address of the name server authoritative for euro6ix.org. Finally, the DNS server of euro6ix.org is queried by the local resolver and returns the IP address of www.euro6ix.org for which it is authoritative. This answer is then forwarded by the local resolver to the client stub resolver. The entire process is called resolving.

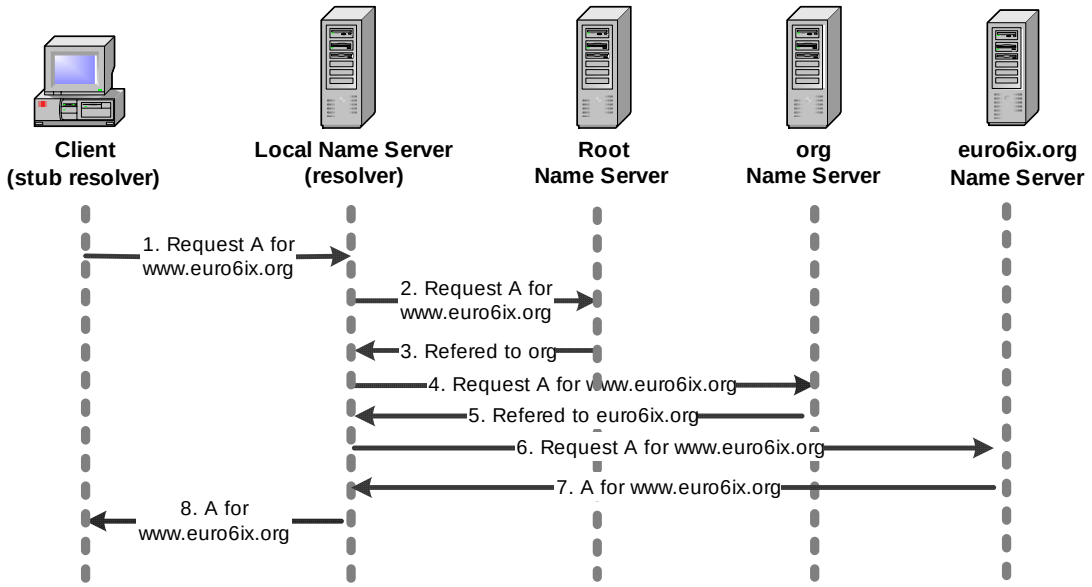


**Figure 2-1: DNS resolving**

Currently there are 13 root DNS servers in the world. Its work is essential to the functionality of the DNS system. Finally we want to underline the use of caching techniques that are employed to reduce the number of requests in order to speed up the resolving process and to reduce network traffic. In the following figure, the resolver immediately returns the IP address queried.

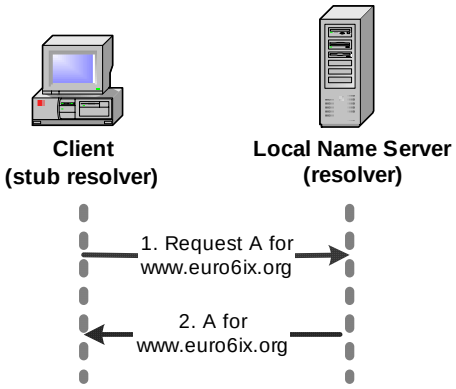


**Figure 2-2: DNS resolving (cache)**

Consequently, each RR that is returned from a DNS server has a certain time-to-live (TTL) which is the time the RR can be cached. DNS is described in RFCs 1033, 1034, 1035 and later RFCs.

### 2.1.1 DNS & IPv6

Several documents exist to describe how IPv6 addresses are supported in DNS. Figure 2-3 shows a table of related RFCs and drafts.

| URL | Title | Summary |
|---|---|---|
| RFC 1886<br>December 1995 | DNS Extensions to support IP version 6 | This document defines the changes that need to be made to the Domain Name System to support hosts running IP version 6 (IPv6). The changes include a new resource record type to store an IPv6 address, a new domain to support lookups based on an IPv6 address, and updated definitions of existing query types that return Internet addresses as part of additional section processing. The extensions are designed to be compatible with existing applications and, in particular, DNS implementations themselves. |
| RFC 2874<br>July 2000 | DNS Extensions to Support IPv6 Address Aggregation and Renumbering | This document defines changes to the Domain Name System to support renumberable and aggregatable IPv6 addressing. The changes include a new resource record type to store an IPv6 address in a manner which expedites network renumbering and updated definitions of existing query types that return Internet addresses as part of additional section processing.<br>For lookups keyed on IPv6 addresses (often called reverse lookups), this document defines a new zone structure which allows a zone to be used without modification for parallel copies of an address space (as for a multihomed provider or site) and across network renumbering events. |
| RFC 3363<br>August 2002 | Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS) | This document clarifies and updates the standards status of RFCs that define direct and reverse map of IPv6 addresses in DNS. This document moves the A6 and Bit label specifications to experimental status. |
| RFC 3364<br>August 2002 | Tradeoffs in Domain Name System (DNS) Support for Internet Protocol version 6 (IPv6) | The IETF has two different proposals on the table for how to do DNS support for IPv6, and has thus far failed to reach a clear consensus on which approach is better. This note attempts to examine the pros and cons of each approach, in the hope of clarifying the debate so that we can reach closure and move on. |
| RFC 3226<br>December 2001 | DNSSEC and IPv6 A6 aware server/resolver message size requirements | This document mandates support for EDNS0 (Extension Mechanisms for DNS) in DNS entities claiming to support either DNS Security Extensions or A6 records. This requirement is necessary because these new features increase the size of DNS messages. If EDNS0 is not supported fall back to TCP will happen, having a detrimental impact on query latency and DNS server load. This document updates RFC 2535 and RFC 2874, by adding new requirements. |
| draft-ietf-dnsext-rfc1886bis-01<br>October 2002 | DNS Extensions to support IP version 6 | This document defines the changes that need to be made to the Domain Name System to support hosts running IP version 6 (IPv6). The changes include a resource record type to store an IPv6 address, a domain to support lookups based on an IPv6 address, and updated definitions of existing query types that return Internet addresses as part of additional section processing. The extensions are designed to be compatible with existing applications and, in particular, DNS implementations themselves. |

**Figure 2-3: IPv6 RFCs and Drafts related to DNS**

The IETF had begun the process of standardizing two different address formats for IPv6 addresses AAAA (RFC 1886) and A6 (RFC 2874) and both are at proposed standard. This had led to confusion and conflicts on which one to deploy. The goal of RFC 3363 is to clarify the situation. This document affirms that:

a) AAAA records are preferable at the moment for production deployment of IPv6

b) A6 records have interesting properties that need to be better understood before deployment.

c) It is not known if the benefits of A6 outweigh the costs and risks.

The main arguments and the issues are covered in a separate document (RFC 3364) that reflects the current understanding of the issues. This document summarizes the outcome of these discussions.

## 2.2 Overview of DNSSEC

The main RFC on securing DNS, RFC 2535, introduces extensions to the Domain Name System to provide authentication and integrity for data received from the DNS databases, mainly through the use of public-key cryptography.

The general idea is that each node in the DNS tree is associated with a public key. Each message from DNS servers is signed under the corresponding private key associated with the public key of the domain. It is important to underline that each public key is associated with a domain (a node in the DNS tree), not with a specific DNS server.

It is assumed that one or more authenticated DNS root public keys are publicly known. These keys are used to generate a digital signature that binds the identity information of each top-level domain to the corresponding public key. The top level domains sign the keys of their subdomains and so on in a process where each parent signs the public keys of all its children in the DNS tree.

Considering our example in the figure above, see the following figure, the local resolver, that owns an authentic copy of the root's public key, will receive the IP address of the DNS server of .org from the root along with its public key all signed via a pre-specified digital signature algorithm. The public key for the .org zone is trusted since it is signed by the root and will be used to verify the public key of the DNS server of euro6ix.org. This process is repeated going down across the tree. To associate a domain name with a certain public key, a so called KEY RR is used.

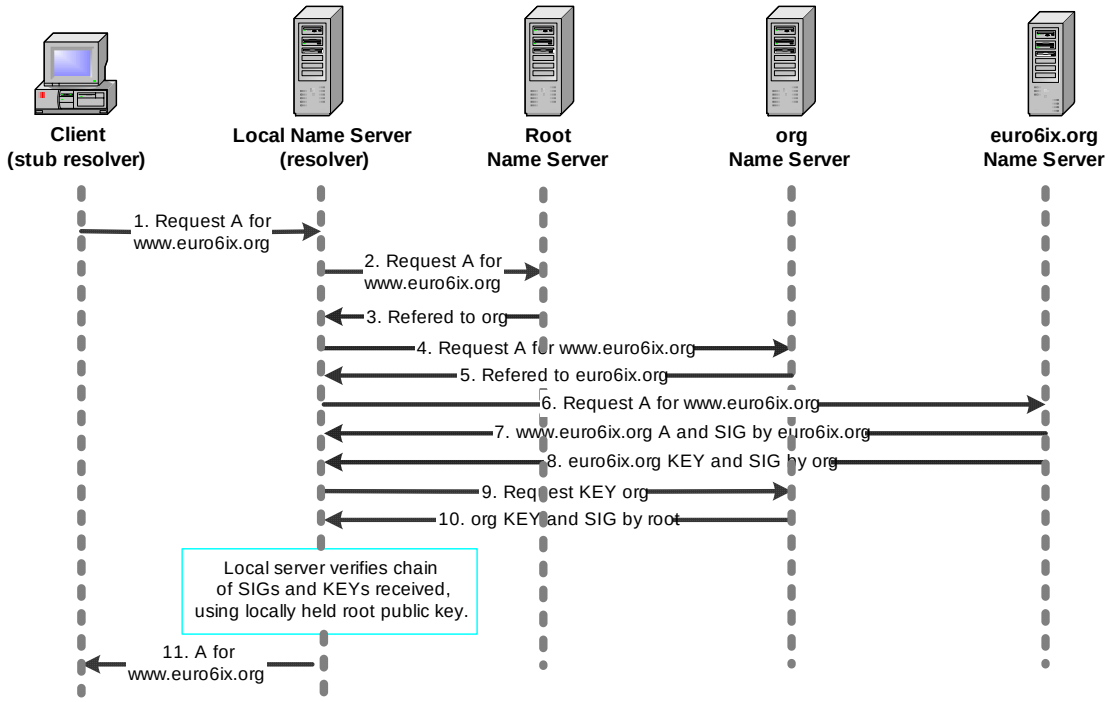


**Figure 2-4: DNSSEC resolving**

A brief description of the DNSSEC resource records follows:

- **SIG**. The signature (SIG) resource record is defined to store signatures in the DNS. If a server supports DNSSEC and is thus security aware, it will attempt to return the relevant RRs and the corresponding SIG records in an answer to a query.

- **KEY**. The KEY record is used to store a public key. The key is associated with DNS name. A resource record with the same name and same type can be associated with several KEY records. The KEY RR is authenticated by a SIG RR like other DNS resource records. It is possible to bind the key for use with TLS, e-mail, DNSSEC and IPSEC using the protocol field. A range of values are reserved for new protocols to be added in the future.

- **NXT.** The purpose of the NXT resource record is to be able to provide data origin authentication of a non-existent name or the non-existence of a certain resource record type for a DNS name. The NXT resource record contains the name of the next name in the zone, thus stating that there can be no resource records between the owner name and the next name. The last NXT record in a zone will contain the zone name, treating the name space as circular. As with KEY records, the NXT records are authenticated by a SIG record.

- **CERT**. The CERT resource record can be used to store certificates in the DNS. The types of certificates currently defined are X.509, SPKI and PGP certificates. As the CERT record can contain a certificate, it is possible to use DNS for storage of public keys. It is intended that personal public keys should be stored in the DNS using the CERT record, and not by using the KEY record.

Two different transaction security mechanisms are defined: transaction signatures (TSIGs) based on symmetric techniques, and public-key signatures which are abbreviated by SIG(0). Both of these signature types can be added to the end of an update packet, authenticating the complete packet. Transaction signatures (TSIG) are created using symmetric encryption methods, meaning that the parties involved in the communication need to have a shared secret. It is convenient to use TSIG to secure dynamic updates or zone transfers between master and slave servers. SIG(0) is similar to TSIG but employs public-key signatures. SIG(0) may not be practical to use on a large scale but it is useful in case integrity protection and authentication of the message as a whole are desired. SIG(0) could be used to authenticate requests when it is necessary to check whether the requester has some required privilege.

### 2.2.1 DNSSEC and IPv6

DNSSEC extensions are defined independently of the IP version used. There is only one document where a relation between DNSSEC and IPv6 is found, RFC 3226, although it is casual. This document mandates support for EDNS0 (Extension Mechanisms for DNS) in DNS entities claiming to support either DNS Security Extensions or A6 records. This requirement is necessary because these new features increase the size of DNS messages.

## 2.3 Interrelationship of DNSSEC documents

Several documents exist to describe the DNSSEC extensions and the implementation-specific details regarding different digital signing schemes. A brief overview of what can be found in each document follows.

All of these documents are under the larger umbrella group of DNS base protocol documents. The DNSSEC set of documents can be partitioned into five main groups. It is possible that some documents fall into more than one of these categories, such as “Domain Name System Security Extensions” [2], and should follow the guidelines for the all of the document groups it falls into. These groups are the following:

- The "DNSSEC protocol" document set refers to the document that makes up the groundwork for adding security to the DNS protocol.

- The "New Security RRs" set refers to the group of documents that define additional Resource Records to the set of base DNS Record types.

- The "DS Algorithm Impl" document set refers to the group of documents that describe how a specific digital signature algorithm is implemented to fit the DNSSEC Resource Record format.

- The "Transactions" document set refers to the group of documents that deal with the message transaction sequence of security-related DNS operations.

- The final document set, "New Security Uses", refers to documents that define how to use the proposed DNS Security extensions for other security related purposes.

More information about the IETF's DNSEXT Working Group [3] can be found in IETF's web server: http://www.ietf.org/html.charters/dnsext-charter.html . A table of related RFCs and drafts is shown in Figure 2-5:.

| Main topic | URL | Title | Summary |
|---|---|---|---|
| **DNSSEC Protocol** | RFC 2535<br>March 1999<br><br>Obsoletes RFC 2065<br>Updated by RFC 2181, RFC 1035, RFC 1034 | Domain Name System Security Extensions | Extensions to the Domain Name System (DNS) are described that provide data integrity and authentication to security aware resolvers and applications through the use of cryptographic digital signatures. These digital signatures are included in secured zones as resource records. Security can also be provided through non-security aware DNS servers in some cases. |
| | RFC 3007<br>November 2000 | Secure Domain Name System (DNS) Dynamic Update | This document proposes a method for performing secure Domain Name System (DNS) dynamic updates. The method described here is intended to be flexible and useful while requiring as few changes to the protocol as possible. The authentication of the dynamic update message is separate from later DNSSEC validation of the data. Secure communication based on authenticated requests and transactions is used to provide authorization. |
| | RFC 3008<br>November 2000 | Domain Name System Security (DNSSEC) Signing Authority | This document proposes a revised model of Domain Name System Security (DNSSEC) Signing Authority. The revised model is designed to clarify earlier documents and add additional restrictions to simplify the secure resolution process. Specifically, this affects the authorization of keys to sign sets of records. |
| | RFC 3090<br>March 2001 | DNS Security Extension Clarification on Zone Status | The definition of a secured zone is presented, clarifying and updating sections of RFC 2535. RFC 2535 defines a zone to be secured based on a per algorithm basis, e.g., a zone can be secured with RSA keys, and not secured with DSA keys. This document changes this to define a zone to be secured or not secured regardless of the key algorithm used (or not used). To further simplify the determination of a zone's status, "experimentally secure" status is deprecated. |
| | RFC 3225<br>December 2001 | Indicating Resolver Support of DNSSEC | In order to deploy DNSSEC (Domain Name System Security Extensions) operationally, DNSSEC aware servers should only perform automatic inclusion of DNSSEC RRs when there is an explicit indication that the resolver can understand those RRs. This document proposes the use of a bit in the EDNS0 header to provide that explicit indication and describes the necessary protocol changes to implement that notification. |
| | RFC 3226<br>December 2001 | DNSSEC and IPv6 A6 aware server/resolver message size requirements | This document mandates support for EDNS0 (Extension Mechanisms for DNS) in DNS entities claiming to support either DNS Security Extensions or A6 records. This requirement is necessary because these new features increase the size of DNS messages. If EDNS0 is not supported fall back to TCP will happen, having a detrimental impact on query latency and DNS server load. This document updates RFC 2535 and RFC 2874, by adding new requirements. |
| | draft-ietf-dnsext-dnssec-opt-in-04<br>November 2002 | DNSSEC Opt-In | In RFC 2535, delegations to unsigned subzones are cryptographically secured. Maintaining this cryptography is not practical or necessary. This document describes an "Opt-In" model that allows administrators to omit this cryptography and manage the cost of adopting DNSSEC with large zones. |
| | draft-ietf-dnsext-dnssec-roadmap-06<br>September 2002 | DNS Security Document Roadmap | DNS Security (DNSSEC) technology is composed of extensions to the Domain Name System (DNS) protocol that provide data integrity and authentication to security aware resolvers and applications through the use of cryptographic digital signatures. Several documents exist to describe these extensions and the implementation-specific details regarding specific digital signing schemes. The interrelationship between these different documents is discussed here. A brief overview of what to find in which document and author guidelines for what to include in new DNS Security documents, or revisions to existing documents, is described. |
| | draft-ietf-dnsext-dnssec-intro-03<br>October 2002 | DNS Security Introduction and Requirements | The Domain Name System Security Extensions (DNSSEC) provide data origin authentication and data integrity. This document introduces these extensions and describes their capabilities and limitations. The services that the security extensions provide and do not provide are discussed. Lastly, the group of documents that describe the DNS security extensions and their interrelationships is discussed. |
| **New Security RRs** | RFC 2931<br>September 2000 | DNS Request and Transaction Signatures (SIG(0)s) | This memo describes a protocol utilizing security concepts necessary for establishing Security Associations (SA) and cryptographic keys in an Internet environment. A Security Association protocol that negotiates, establishes, modifies and deletes Security Associations and their attributes is required for an evolving Internet, where there will be numerous security mechanisms and several options for each security mechanism. The key management protocol must be robust in order to handle public key generation for the Internet community at large and private key requirements for those private networks with that requirement. The Internet Security Association and Key Management Protocol (ISAKMP) defines the procedures for authenticating a communicating peer, creation and |

| | | | |
|---|---|---|---|
| | | | management of<br>Security Associations, key generation techniques, and threat mitigation (e.g. denial of service and replay attacks). All of these are necessary to establish and maintain secure communications (via IP Security Service or any other security protocol) in an Internet environment. |
| | RFC 2538<br>March 1999 | Storing Certificates in the Domain Name System (DNS) | Cryptographic public key are frequently published and their authenticity demonstrated by certificates. A CERT resource record (RR) is defined so that such certificates and related certificate revocation lists can be stored in the Domain Name System (DNS). |
| **DS Algorithm Impl** | RFC 3110<br>May 2001 | RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS) | This document describes how to produce RSA/SHA1 SIG resource records (RRs) and, so as to completely replace RFC 2537, describes how to produce RSA KEY RRs.<br>Since the adoption of a Proposed Standard for RSA signatures in theDNS (Domain Name Space), advances in hashing have been made. A new DNS signature algorithm is defined to make these advances available in SIG RRs. The use of the previously specified weaker mechanism is deprecated. The algorithm number of the RSA KEY RR is changed to correspond to this new SIG algorithm. No other changes are made to DNS security. |
| | draft-ietf-dnsext-ecc-key-02<br>May 2002 | Elliptic Curve KEYs in the DNS | A standard method for storing elliptic curve cryptographic keys in the Domain Name System is described which utilizes DNS KEY resource record. |
| | draft-ietf-dnsext-rfc2539bis-dhk-02<br>May 2002 | Storage of Diffie-Hellman Keys in the Domain Name System (DNS) | A standard method for storing Diffie-Hellman keys in the Domain Name System is described which utilizes DNS KEY resource records. |
| **Transaction** | RFC 2845<br>May 2000 | Secret Key Transaction Authentication for DNS (TSIG) | This protocol allows for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates as coming from an approved client, or to authenticate responses as coming from an approved recursive name server. |
| | RFC 2930<br>September 2000 | Secret Key Establishment for DNS (TKEY RR) | This document describes a Transaction Key (TKEY) RR that can be used in a number of different modes to establish shared secret keys between a DNS resolver and server. |
| | draft-ietf-dnsext-tkey-renewal-mode-02<br>September 2002 | TKEY Secret Key Renewal Mode | This document defines a new mode in TKEY (RFC2930 [4]) and proposes an atomic method for changing secret keys used for TSIG (RFC2845 [5]) periodically. Originally, TKEY provides methods of setting up shared secrets other than manual exchange, but it cannot control timing of key renewal very well though it can add or delete shared keys separately. This proposal is a systematical key renewal procedure intended for preventing signing DNS messages with old and non-safe keys permanently. |
| **Implementation Notes** | draft-richardson-ipsec-rr-00<br>August 2002 | A method for storing IPsec keying material in DNS | This document describes a new resource record for DNS. This record may be used to store public keys for use in IPsec systems. |
| | draft-ietf-secsh-dns-01<br>November 2002 | Using DNS to securely publish SSH key fingerprints | This document describes a method to verify SSH host keys using DNSSEC. The document defines a new DNS resource record that contains a standard SSH key fingerprint. |

**Figure 2-5: DNSSEC RFCs and Drafts**

# 3. DNSSEC OVER IPv6 IMPLEMENTATION: BIND9

The most widespread DNS implementation in use nowadays is *Berkely Internet Name Domain (BIND)*. It is an open source [6] implementation distributed by the Internet Software Consortium (ISC). There are several estimates about how widely BIND is used as a name server, with results varying from 88 to 99% of the zones being administered by BIND. Besides, most of other existing DNS implementations available are derived from BIND. Currently version 9 of BIND [7] is capable of acting as an authoritative server for DNSSEC secured zones, as well as providing support for DNS queries over IPv6.

In summary, BIND is the DNS reference implementation and implements almost all the main extensions being defined for DNS. For all these reasons, BIND will be used for the DNSSEC tests described in this document.

## 3.1 IPv6 Support

BIND9, a mayor rewritten of BIND software, comes with full support for IPv6, including:

- Answers to DNS queries over IPv6 sockets
- IPv6 resource records (A6, DNAME, etc)
- Bitstrings labels
- Experimental IPv6 resolver library

However, on some operating systems, IPv6 and IPv4 sockets interact in unexpected ways. To reduce the impact of these problems, BIND separates IPv4 and IPv6 support, including independent configuration commands to activate each protocol.

BIND does not listen by default for requests on IPv6 addresses. To accept DNS queries over IPv6, the command "listen-on-v6" must be included in the "options" section of the main configuration file ("named.conf"). In IPv4, this command allows the specification of the interfaces where bind should listen on for DNS requests: if the word "any" is included, bind will listen on every network interface; on the contrary, if a list of IPv4 addresses is specified, bind will only listen for request on that interfaces.

Unfortunately, as stated on BIND documentation and as concluded from some of the tests described later in this document, the configuration of BIND to listen on specific IPv6 interfaces is not yet implemented. For IPv6, only the option "any" works appropriately.

The lack of implementation of the complete "listen-on" feature over IPv6, which works as expected over IPv4 interfaces, prevents from launching several copies of BIND9 inside a computer, each one listening on a different IPv6 interface. This feature is very useful, for example, to limit the answering of DNS requests to some interfaces (for example in hosts with public and private interfaces), or to maintain experimental name servers on the same machine that runs the production server. As mentioned later, this problem will complicate the creation of a DNS emulation environment to facilitate DNSSEC tests.

Finally, BIND server includes a new feature that allows it to automatically convert RFC1886-style recursive lookup requests into RFC2874-style lookups, when enabled using the new option "allow-v6-synthesis". This allows stub resolvers that support AAAA records but not A6 record chains or binary labels to perform lookups in domains that make use of these IPv6 DNS features. However, this useful feature is becoming less important, as the support for A6 records have been declared experimental.

In summary, BIND comes with a high degree of IPv6 support, although it has some important IPv6 features missing, like the “listening-on” specific interfaces option. As the server has been completely rewritten from version 9, and new features are being added version-by-version, it should be expected to have a full IPv6 support in the near future.

## 3.2 DNSSEC Support

BIND9 is capable of acting as an authoritative server for DNSSEC secured zones. This functionality is said to be stable and complete except for the lack of support for wildcard records in secure zones.

Below is a brief overview of the state of the DNSSEC implementation in the latests release of BIND9 (9.2.1 at the time of writing this document).

a) Serving Secure Zones. When acting as an authoritative name server, BIND9 includes KEY, SIG and NXT records in responses as specified in RFC2535 when the request has the DO flag set in the query.

b) Secure Resolution. Basic support for validation of DNSSEC signatures in responses has been implemented but should still be considered experimental.

   When acting as a caching name server, BIND9 is capable of performing basic DNSSEC validation of positive as well as nonexistent responses. This functionality is enabled by including a "trusted-keys" clause in the configuration file, containing the top-level zone key of the DNSSEC tree.

   Proof of insecure status for insecure zones delegated from secure zones works when the zones are completely insecure. Privately secured zones delegated from secure zones will not work in all cases, such as when the privately secured zone is served by the same server as an ancestor (but not parent) zone.

c) Secure Dynamic Update. Dynamic update of secure zones has been implemented, but may not be complete. Affected NXT and SIG records are updated by the server when an update occurs. Advanced access control is possible using the "update-policy" statement in the zone definition.

d) Secure Zone Transfers. BIND 9 does not implement the zone transfer security mechanisms of RFC2535 section 5.6, and there are no plans to implement that feature in the future as it is considered inferior to the use of TSIG or SIG(0) to ensure the integrity of zone transfers.

# 4. DNSSEC ACTIVITIES INSIDE EURO6IX

Due to the importance of DNS service and the lack of security of the currently used protocols and implementations, it was decided to launch a set of activities inside WP4 in order to gain experience in the use of DNSSEC implementations and on the procedures for managing secure DNS services based on it. Besides, the possibilities offered by a secure DNS to store certificates and work together with a PKI, makes the experimentation with DNSSEC even more attractive.

The final objective of these activities is to launch a secure DNS service over Euro6IX network. However, the lack of maturity of current implementations, as well as the lack of experience in the management of the service, requires to experiment first over isolated DNS servers (inside WP4) before deploying the service over the production environment (WP3).

DNSSEC activities carried out during the second semester of the first year have focused on:

- Development of a **DNS emulation environment**, to allow the creation of complex DNS scenarios made of complete name server hierarchies inside one computer, in order to test different scenarios and configuration alternatives before deploying the service to real servers.

- **Local tests** made by each participating partner inside its testbed network, in order to acquire the basic know-how and to create the basic infrastructure needed for a DNSSEC service. These tests, together with the ones made over the emulated environment, will try to produce basic "cookbooks" about how to configure and manage DNSSEC, to be used in further tests.

- **Inter-partner tests** over Euro6IX network. Based on the know-how acquired in the previous local and emulated tests, a limited pilot service between the three partners involved in this subactivity was created. Although only basic tests have been carried out over it, this pilot service will be used during the second year to do deeper tests, concentrating on service management procedures (key synchronization and distribution, foreseen and unforeseen key changes, etc).

This dual approach we have followed –doing tests over an emulated environment and over real networks- has shown to be powerful and versatile. It has allowed us to test and debug the configurations locally before deploying them to the different sites, with the resulting saving in time. Besides, when a problem is detected over the real network testbed, the real configurations can be reproduced over the emulated environment in order to diagnose and experiment the solutions that later be implemented over the testbed. The approach has been very useful in the activities carried out till now; but we think it will be even more useful for next year activities, when the pilot experiments will involve a higher number of domains, servers and organizations.

The rest of the section is organized as follows. Section 4.1 presents the two different scenarios selected to test DNSSEC. Section 4.2 summarizes the main administrative tasks associated to the management of a secure zone. Section 4.3 describes the DNS emulation scenario, a set of scripts designed to emulate a complete hierarchy of DNS servers inside a machine. Finally, sections 4.4 and 4.5 describe the tests carried out over the Euro6IX network.

## 4.1 DNSSEC Testing Scenarios

In order to authenticate responses from DNS servers, a resolver needs to know the public keys of a trusted zone above the zone being queried. Typically resolvers will be configured with the public keys of one or more root servers, where the "chain of trust" begins. But it is also possible to configure them with the public key of an intermediate zone between the zone queried and the root servers. This fact allows the creation of secure zones inside an insecure DNS service and facilitates the migration to a secure service.

For the above mentioned reasons, two possible DNSSEC testing scenarios have been defined:

1) A completely **private scenario (**Figure 4-1**)**, where several name servers are used to create a complete DNS hierarchy including private root servers. In this case, resolvers will be configured with the trusted public keys of private root servers. Although the scenario is private, to avoid problems due to possible missconfigurations, a fictitious top level domain (TLD) will be defined (.e6), with subdomains delegated to partners participating in the tests, for example: upm.e6, umu.e6 or consulintel.e6.

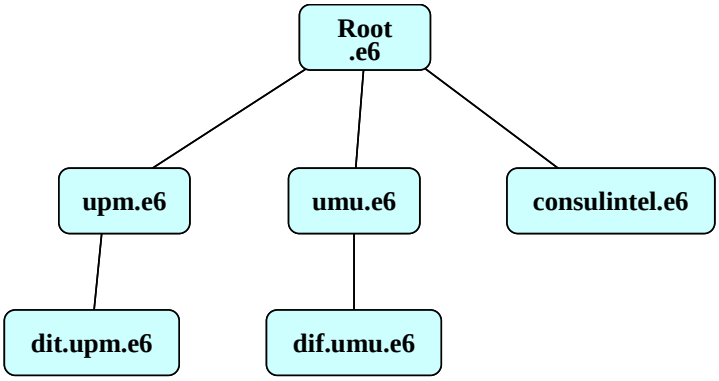


**Figure 4-1: Private DNSSEC testing scenario**

2) A **secure subdomain** (Figure 4-2) inside one of the public domains controlled by Euro6IX (euro6ix.org, euro6ix.net or euro6ix.com). For example, sigz.euro6ix.org zone and all sub-zones within that hierarchy could be implemented as secure zones. Delegated sub-zones within sigz.euro6ix.org could be created and subdelegated to different partners participating in the tests, for example, umu.sigz.euro6ix.org, upm.sigz.euro6ix.org or consulintel.sigz.euro6ix.org. In this case, resolvers will be configured with the trusted public keys of the secure subdomain (sigz.euro6ix.org).

Both scenarios have been considered interesting for different reasons. In the first case, the private scenario has the advantage of being a "complete secure scenario", allowing the testing of DNSEC procedures as they would be when secure root servers are deployed. Experiences obtained from this test could even serve as recommendations for the migration of root servers or country TLD servers.

In the second case, the creation of secure zones will allow the testing of a scenario that could be easily deployed over a production network without waiting for the availability of secure root servers. That will most probably be the scenario used to deploy the secure DNS service over Euro6IX network in the future.

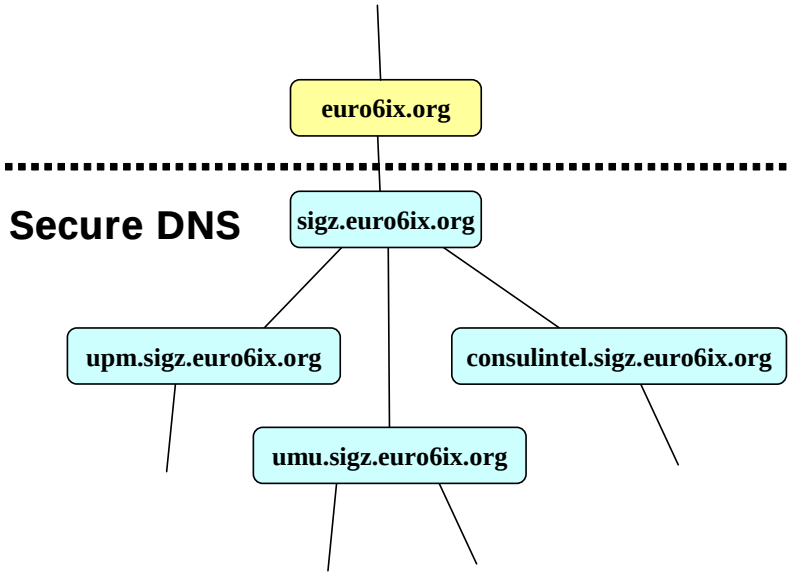


**Figure 4-2: Secure subdomain DNSSEC testing scenario**

In any case, as DNS configuration and management is difficult and sometimes very tricky, it is desirable that the DNS servers used for testing DNSSEC are different from the production ones, to avoid interferences with the standard DNS service. Besides, the problems described in the previous section (impossibility to start two or more different DNS servers over the same machine bound to different IPv6 addresses) imply that the experimental DNS servers should run over different machines.

## 4.2 Managing a secure DNS zone

This section summarizes the main administrative tasks associated with creating and maintaining a secure zone. A brief description of each task follows:

a) Securing zone transfers

This paragraph describes how a primary server and a secondary server must be configured to enable TSIG for zone transfers. TSIG is based on a secret shared and is used to sign the content of each DNS packet. To configure TSIG one has to perform the following steps:

- Create and distribute a shared secret: use the dnskeygen tool included with BIND distribution.
- At the primary server: create an access list specifying which keys are allowed transfer.
- At the secondary server: tell which keys to use when contacting which primary servers.
- Synchronize clocks (for example using NTP).

b) Generation of "zone keys"

DNSSEC authentication is based on the use of public/private zone keys. A zone creates public/private key pairs. The private key is known only by the zone and is used to sign the zone records. The public key is made widely available by placing it in a KEY record. Resolvers who trust the public key can verify signatures and authenticate records received from the zone.

Generating new keys is a simple process that can be done with standard tools such as the dnskeygen tool included with BIND distribution. A crucial task for a DNSSEC zone is keeping private keys private and insuring that key pairs expire in a reasonable time.

A good advice is to generate four keys for each zone: DSA and RSA key to sign other keys, and DSA and RSA key to sign the zone. But the sub-zones are free to determine the key algorithms, numbers, and lifetimes as required to meet their operational and administrative needs.

c) Signing of the zone file

Now that once the zone keys are generated, administrative activities can continue with signing of the zone file. Signing the records with the zone keys will produce the SIG records. A signed zone file should include one SIG record per zone key. The SIG record contains the digital signature and related information which can be used to authenticate the data. The NXT records are also stored in the zone file.

The dnssigner tool provided with the BIND distribution automatically orders the zone file and generate the NXT records.

Care must be taken to keep the private keys and to insure that the signatures are given the proper expiration dates. A signature expiration date should be no later then the anticipated expiration date for the corresponding public key. It is expected that different keys will have different lifetimes.

After editing the resource records and updating the SOA, an administrator must re-sign the changed (or new) records.

d) Signing zone keys and administering delegations

A zone only stored the name servers associated with its sub-zones. Interaction between a zone and a sub-zone was required only if the sub-zone changed name server locations. DNSSEC requires a much higher degree of coordination between a zone and its sub-zones. The increased coordination is a result of key exchanges and key signing that must occur between a zone and its sub-zones.

A good advice is that a zone or sub-zone only generates and stores its keys. This rule insure that any change in a sub-zone never require a corresponding change in the parent's zone file.

A domain is responsible for insuring that its KEY set is signed by its parent at the appropriate times. The domain should send new KEY sets whenever the SIG records from the parent expire or whenever the domain generates new KEY records. Key signing is currently accomplished by an off-line exchange of keys. After receiving the SIG records from its parent, a sub-domain must check each SIG record before adding the SIG record to the zone file. After the off-line KEY exchange and SIG verification is complete, the sub-domain stores its KEY record set and the accepted SIGs in its zone file.

## 4.3 DNS Emulation Environment

### 4.3.1 Introduction

Testing distributed applications or services -as it is the DNS- is generally difficult due to the number of machines involved and the distance between them (either physical or even administrative, as they are typically managed by different organizations). In order to test DNS security extensions in an easy way, we sought for an emulation environment where complex DNS hierarchies could be tested inside one machine, speeding up the process of learning how to configure and manage the service.

The basic requirements for the DNS emulation environment we wanted to build were the following:

- Several independent name servers running inside the same machine, each one with its own configuration files. Server should run the same software as it would be used in a real server.

- Communication between servers through internal loopback interfaces, as well as communication with external servers through standard IP connectivity, in order to allow mixed scenarios including real and emulated servers.

- Possibility to maintain different testing scenarios (although not testing them simultaneously).

- Automatic procedures to start and stop the set of servers defined for each testing scenario.

- Easy way of visualization of messages interchanged between name servers, as well as log messages of each name server.

With all this requirements on mind, we investigated how to achieve that goals and come to a set of UNIX scripts, written in “shell” and “perl” languages derived from the testing software included in BIND distribution (see bin/test/system/ directory).

Using the scripts, the main steps to create a DNS emulated environment are briefly described in the following paragraphs:

1. **Create a new test directory** (for example, “e6”), under the main tests directory.

2. **Create name servers configurations**. Each server has its own subdirectory under the test directory named “nsX”, being X the number of the server (from 0 to 9 at present, although it can be easily extended to a higher number). The directory must contain the “named.conf” configuration file, as well as the zone files associated with that name server.

3. **Create new IP subinterfaces**, associating new IP addresses to an existing network interface or to the loopback interface. Each name server will specifically “bind” to one of these addresses, only answering requests destined to that address. There is a script named “ifconfig.sh” to create or destroy these subinterfaces, allowing the creation of IPv4 only, IPv6 only or IPv4 and IPv6 tests. Each name server configuration must include the “listen-on” command in the options section of “named.conf”, in order to bind to only one of the addresses created.

4. **Start name servers**, using the "run.sh" script. That script will search for directories named "nsX" inside the testing directory and will start a new name server daemon (named) using the configuration files found on the directory. Traces from each server started will be directed to the "named.run" inside its directory. The script will wait till all servers have started (it makes a test query to each one) and show the message "Type any key to finish the test" and wait for the order to stop all name servers.

5. **Make the DNS tests** using standard tools like dig or host (nslookup is not recommended as it is obsoleted). Queries can be directed to any of the servers in the hierarchy by using the corresponding option. For example:

   **> dig @10.53.1.1 –t AAAA h1.upm.e6**

   will send a query to the server listening on 10.53.1.1 address to know the AAAA record associated with "h1.upm.e6" name.

6. **Message interchange** can be easily traced using standard "sniffers" like tcpdump or tethereal (the textual version of Ethereal [8]). Filters can be used to restrict the messages showed to just DNS messages. For example:

   **> tcpdump –n –i lo0 port 53**

   will only show packets destined to DNS (port 53) interchanged over the loopback interface ("-n" option avoids the inverse conversion from addresses to names made by tcpdump). Or:

   **> tcpdump –n –i lo0 port 53 and host 10.53.1.1**

   will restrict the previous filter to messages with source or destination in a specific server (10.53.1.1).

7. **Finish the test** by just typing any key on the window where the test was started.

### 4.3.2 Emulated Environment Utility

We think that such an emulation environment has a lot of advantages, several of them exceeding the scope of the tests described here. It can be used:

- As a general **DNS testing tool**. As mentioned, it can be used to test complex DNS scenarios over a single machine, without involving several different computers or interfering with production services. Tests are made using the same software and the same configurations as they would be used in a real scenario. Besides, the exchange of messages between the different servers can be easily showed.

- As a **DNS training tool**. In general, it can be used to easily test DNS example configurations in isolated environments, speeding up the DNS learning process. Moreover, if combined with good configuration examples specially oriented to learn IPv6 DNS extensions, could be a valuable educational tool to teach network administrators how to organize the transition to IPv6 of their DNS services.

- As a tool to help the **operation of DNS servers**. DNS managers can use the tool to maintain an "emulated copy" of their DNS servers, allowing them to test changes before deploying them to real servers. Different bind versions could even be mixed in an emulated scenario in order to make interoperability tests.

### 4.3.3 Testing DNSSEC

Using the emulation environment described above, the basic scenario shown in Figure 4-3 was tested. That scenario reproduces a complete DNS hierarchy, including a root server for a fictitious TLD (.e6) and three subdomains, one for each participating partner. It is implemented with six dual stack name servers:

- **ns1.e6** (root server for e6 domain), listening on 10.53.1.1 and 3ffe:ffff:0:5::1 addresses.
- **ns1.upm.e6** (primary server for upm.e6 domain), listening on 10.53.1.2 and 3ffe:ffff:0:5::2 addresses.
- **ns1.dit.upm.e6** (primary server for dit.upm.e6 domain), listening on 10.53.1.3 and 3ffe:ffff:0:5::3 addresses.
- **ns1.umu.e6** (primary server for umu.e6 domain), listening on 10.53.1.4 and 3ffe:ffff:0:5::4 addresses.
- **ns1.dif.umu.e6** (primary server for dif.umu.e6 domain), listening on 10.53.1.5 and 3ffe:ffff:0:5::5 addresses.
- **consulintel.e6** (primary server for cons.e6 domain and secondary for .e6), listening on 10.53.1.6 and 3ffe:ffff:0:5::6 addresses.

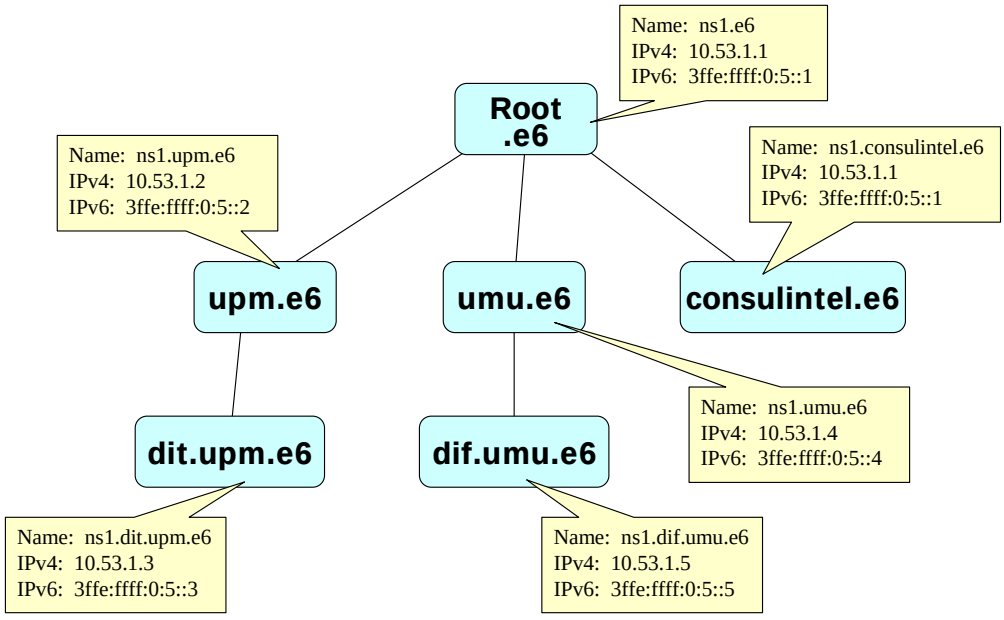


**Figure 4-3: Private scenario emulated**

In this scenario, which will be used as the starting point for more complex ones in the future, all server configurations include direct and inverse resolution and delegations for IPv4 and IPv6, as well as loopback addresses resolution. Besides, a primary-secondary relation exists between ns1.e6 (primary of .e6) and ns1.consulintel.e6 (secondary for .e6). Both servers are configured to use TSIG to exchange zone information. The rest of the servers use the DNS security extensions, including SIG, KEY and NXT records in their zone files.

To test the example, the steps described in the previous section have to be followed. First, the IP subinterfaces needed are created using the command "**sh ifconfig.sh up"**. After running it, the addresses configured can be seen with "ifconfig" command:

```
asterix2# ifconfig
….
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet 127.0.0.1 netmask 0xff000000
        inet 10.53.1.1 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::1 prefixlen 64
        inet 10.53.1.2 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::2 prefixlen 64
        inet 10.53.1.3 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::3 prefixlen 64
        inet 10.53.1.4 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::4 prefixlen 64
        inet 10.53.1.5 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::5 prefixlen 64
        inet 10.53.1.6 netmask 0xffffffff
        inet6 3ffe:ffff:0:5::6 prefixlen 64
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 552
faith0: flags=8002<BROADCAST,MULTICAST> mtu 1500
```

As it is shown, twelve new private addresses have been created on the loopback (lo0) interface (six IPv4 and six IPv6).

Later, the name servers are started using "**sh run.sh e6**" command. At this point the shell shows the next messages:

```
asterix2# sh run.sh e6
S:e6:Wed Dec 4 22:39:25 CET 2002
T:e6:1:A
A:System test e6
Type any key to finish the test
```

Once the "Type any key…" message is shown, the emulated environment is ready to accept queries (during the startup, some tests are made to be sure that the servers start correctly; if any problem occurs, an error message is shown and all servers are killed). As mentioned before, DNS server logs are saved into separate files (called named.run) under each name server directory for later analysis.

For example, the following query using "dig" tool with dnssec option could be made to ask "ns1.umu.e6" for the IPv6 address corresponding to "h1.upm.e6" name:

```
home-gw# dig @10.53.1.4 +dnssec -t AAAA h1.upm.e6

; <<>> DiG 9.2.1 <<>> @10.53.1.4 +dnssec -t AAAA h1.upm.e6
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45888
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;h1.upm.e6.                  IN      AAAA

;; ANSWER SECTION:
h1.upm.e6.          300     IN      AAAA    3ffe:ffff:10:10::1
h1.upm.e6.          300     IN      SIG     AAAA 1 3 300 20030112142137 20021213142137
57720 upm.e6. LBJMYrUaMQx/ioC2jkZRWqI2ZxDBKwk2zZPf2uJ4P/Ii0pWWtF4H46O3
29YRcTKCxl9iceH5VMaPPIy/1rUodA==

;; AUTHORITY SECTION:
upm.e6.             300     IN      NS      ns1.upm.e6.
upm.e6.             300     IN      SIG     NS 1 2 300 20030112142137 20021213142137
57720 upm.e6. ARshX48yuIDC6BOP1iplYDV2cODUkfOBNT1pcecADUt4I2I+akI6Bqd8
tt2FNiXXtm2iARPZZHvOmtUn3jceqg==

;; Query time: 109 msec
;; SERVER: 10.53.1.4#53(10.53.1.4)
;; WHEN: Thu Dec 30 18:25:27 2010
;; MSG SIZE  rcvd: 288
```

The traces captured with tetheral analyzer on the loopback interface show how the query made to ns1.umu.e6 (10.53.1.4) progressed to the root server (10.53.1.1), which sent a response to redirect the query to ns1.upm.e6 (10.53.1.2), which in fact sent the answer containing the AAAA record plus the signature associated:.

```
Capturing on lo0
  0.000  10.53.1.4->10.53.1.4  DNS Standard query AAAA h1.upm.e6
  0.021  10.53.1.4->10.53.1.1  DNS Standard query AAAA h1.upm.e6
  0.030  10.53.1.1->10.53.1.4  DNS Standard query response
  0.068  10.53.1.4->10.53.1.2  DNS Standard query AAAA h1.upm.e6
  0.078  10.53.1.2->10.53.1.4  DNS Standard query response AAAA 3ffe:ffff:10:10::1 SIG
  0.129  10.53.1.4->10.53.1.4  DNS Standard query response AAAA 3ffe:ffff:10:10::1 SIG
```

If the same query is repeated, the traces show how the response comes directly from the cache of the server queried:

```
349.393  10.53.1.4->10.53.1.4  DNS Standard query AAAA h1.upm.e6
349.409  10.53.1.4->10.53.1.4  DNS Standard query response AAAA 3ffe:ffff:10:10::1 SIG
```

### 4.3.4 Problems Found and Possible Solutions

When testing the above mentioned example, an important problem was discovered that prevents the correct working of servers over IPv6. In principle, on IPv6 it is possible to specify by means of the "listen-on-v6" command the IPv6 address the server should bind to. For example, for ns1.upm.es server, its "named.conf" file contains the sentence:

**listen-on-v6 { 3ffe:ffff:0:5::2; };**

This sentence should make the server to bind to that address and only answer queries sent to it. However, as it was concluded from tests and as it was discovered later in the documentation, this functionality is not implemented in IPv6 (at least on the latest version of BIND tested: 9.2.1); only the options "any" (listen on every interface) and "none" (do not listen over IPv6) are implemented at present. That fact prevents the emulation environment from working properly over IPv6.

The only way to start an emulated environment over IPv6 is to configure all servers with the "any" option. In this case, all servers start, but the queries directed to any of them are always processed by the first server started (the root server in our example), independently of the address the query is destined.

A side effect discovered when investigating this problem has to be with the communications between dual stack name servers. If a name server delegates a subzone inside its zone file, and includes both the IPv4 and IPv6 address of the name server of the subzone, when it receives a recursive query and has to query the subzone server, it always uses the IPv4 address. For example, making a query over IPv6 to :

For example, if a recursive query is sent to ns1.dif.umu.es over IPv6 asking for the A record of h1.dit.upm.e6, the root server receives the query (due to the above mentioned problem) and it asks ns1.upm.es using IPv4:

```
>  dig @3ffe:ffff:0:5::5 -t A h1.dit.upm.es

876 3ffe:ffff:0:5::5.3839 > 3ffe:ffff:0:5::5.53:  40903+ A? h1.dit.upm.e6. (31)
895 10.53.1.1.3387 > 10.53.1.2.53:  49540 [1au] A? h1.dit.upm.e6. OPT  UDPsize=2048
904 10.53.1.2.53 > 10.53.1.1.3387:  49540 0/1/4 (194)
924 10.53.1.1.3387 > 10.53.1.3.53:  12302 [1au] A? h1.dit.upm.e6. OPT  UDPsize=2048
933 10.53.1.3.53 > 10.53.1.1.3387:  12302* 1/1/2 A 10.10.10.1 (92)
950 3ffe:ffff:0:5::5.53 > 3ffe:ffff:0:5::5.3839:  40903 1/1/0 A[|domain]
```

This side effect should be investigated deeper, in order to see how to control what protocol is used for recursive queries.

Some solutions were initially investigated to solve the IPv6 bind problem:

1. **Start name servers over different ports**. This can be easily done as the "listen-on-v6" option allows the specification of a port. However, when two DNS servers want to communicate, they send their queries to the default DNS port (port 53), and this behavior can not be changed. That makes this solution unviable.

2. **Implement the complete "listen-on-v6" functionality on BIND.** This is the best solution, as it will allow in IPv6 the same behavior that has been successfully tested over IPv4. However, it will imply to study and modify a large program as it is BIND. A preliminary study of BIND code should be done to evaluate the effort needed. In any case, BIND developers will be contacted to know if they plan to implement this functionality in the near future.

3. **Use a virtualization environment,** to start each server on a different virtual machine. This option has been used, for example, in the Route Server subactivity to emulate an IX by starting several BGP daemons on the same machine using User Mode Linux virtualization software. This solution is the realest one, as each name servers will run inside a different virtual machine; however it requires more resources in terms of disk, CPU and memory, and it is much slower.

Solutions 2 and 3, apart from others that could arise, will be investigated deeper during the first months of the next year, in order to select and implement one of them and have a working DNS emulation environment for IPv6.

## 4.4 Local Test-Bed

This section describes some of the local tests done in partner's test-beds. Work has started with the testing of simple isolated security functionalities, like TSIG (transaction signatures) mechanisms, and gone to test complex scenarios were all DNSSEC security extensions were involved.

The intention of these tests description is not to make a detailed explanation of the different possibilities when using security extensions, but just to give a detailed view of the work we have done and provide a configuration "cookbook". That is why all commands and log messages are showed.

All the tests have been carried out using native only IPv6 configuration and using a recent version of BIND (9.2.0) that includes TSIG and DNSSEC extensions (TSIG support exists from version 8.2 of BIND, but DNSSEC support exists only since version 9 was released).

### 4.4.1 Local Tests Scenario

First we show the network we have used in the local tests. There is nothing special on it. We describe the configuration used in each server depending on the tests that we made and also the tests performed, which include the commands and log messages seen.

Although several possible scenarios exist to test secure DNS extensions –for example, the private domain described in the previous section-, the local tests have been carried out using a secure subdomain of one of the domains assigned to Euro6IX project:

**dnssec.consulintel.euro6ix.net**

Besides, for the local tests we have used a configuration made of two DNS servers in different IPv6 networks. One was the master (Bart) and the other the slave (Homer) server for the zones to be secured (Figure 4-4).


**Figure 4-4: DNSSEC local-tests network**

Both Homer and Bart must be synchronized in order to have a working security mechanism. This synchronization is needed because of the use of timestamps in the secured messages in order to avoid replay attacks. In BIND the maximum difference allowed is 5 minutes, not configurable.

The simplest solution we found is to use the *date* command in both servers at the same time, more or less. We made:

```
#> date 12091245
lun dic  9 12:45:00 UTC 2002
```

Other solution is to use **NTP** servers in order to have the servers synchronized. We are willing to add this functionality to our network and servers. This could be done in next stages of the DNSSEC activity.

It must be noted the importance of synchronization in a real security infrastructure.

### 4.4.2 TSIG Local Tests

**A - Server's configuration:**

As mentioned, we have used the *dnssec.consulintel.euro6ix.net* domain for testing purposes. The server called bart is the master for this zone and the server called homer is the slave.

The file /etc/named.conf of both servers follows:

**BART named.conf file:**

```
options {
       directory "/var/named";
       listen-on-v6 { any;};
};

# ----------------------------------------
#      TSIG configuration
# ----------------------------------------
key bart-homer.dnssec.consulintel.euro6ix.net. {
       algorithm hmac-md5;
       secret "NG9NTSDHeSE3lRthncF1ww==";
};

# ----------------------------------------
...
zone "dnssec.consulintel.euro6ix.net" IN {
       type master;
```

```
        file "dnssec.consulintel.euro6ix.net.zone";
        allow-transfer {
                key bart-homer.dnssec.consulintel.euro6ix.net.;
        };
};
```

**BART named.conf file:**

```
options {
        directory "/var/named";
          listen-on-v6 { any;};
};

# ---------------------------------------
#       Configuracion de TSIG
# ---------------------------------------
key bart-homer.dnssec.consulintel.euro6ix.net. {
          algorithm hmac-md5;
          secret "NG9NTSDHeSE3lRthncF1ww==";
};

server 2001:800:40:2a21:205:1cff:fe0a:e2c3 { # bart
        keys {bart-homer.dnssec.consulintel.euro6ix.net.; };
};
# ---------------------------------------
...
zone "dnssec.consulintel.euro6ix.net" IN {
          type slave;
          file "dnssec.consulintel.euro6ix.net.zone.slave";
          masters {2001:800:40:2a21:205:1cff:fe0a:e2c3; # bart
        };
};
```

To generate the shared key we used the following command:

```
#>dnssec-keygen –a HMAC-MD5 –b 128 –n HOST bart-homer.dnssec.consulintel.euro6ix.net.
```

“-a HMAC-MD5” indicates the algorithm that will use the key; only hmac-md5 is supported by now.

“-b 128” indicates the number of bits of the key.

“-n HOST” indicates the type of key to generate.

This command creates two files, both of them containing the key.

Homer has the *server* statement’s *keys* substatement, which tells him to sign queries and zone transfers requests sent to a particular remote name server, Bart in our case.

Now we can add the allow-transfer statement in Bart, restricting zone transfers to those signed with the *bart-homer.dnssec.consulintel.euro6ix.net* key.

Bart also signs the zone transfer, which allows Homer to verify it.

**B - Running the tests:**

**BART:**

When the master is started we see in the logs (the zone has changed):

```
Dec  9 13:07:00 bart named[2082]: zone dnssec.consulintel.euro6ix.net/IN: sending
notifies (serial 2002120902)

Dec  9 13:08:22 bart named[2082]: client 2001:618:10:2a03:5054:5ff:fef6:7487#1074:
transfer of  dnssec.consulintel.euro6ix.net/IN': AXFR-style IXFR started
```

If the slave changes the key, resulting different keys in master and slave, we see in the server's log:

```
Dec  9 13:05:52 bart named[2041]: client 2001:618:10:2a03:5054:5ff:fef6:7487#53:
request has invalid signature: tsig verify failure
```

**HOMER:**

We started the server. We saw, when the server starts with a new version of the zone:

```
Dec  9 12:58:40 homer named[2088]: zone dnssec.consulintel.euro6ix.net/IN: transfered
serial 2002120902
Dec  9 12:58:40 homer named[2088]: transfer of 'dnssec.consulintel.euro6ix.net/IN'
from 2001:800:40:2a21:205:1cff:fe0a:e2c3#53: end of transfer
Dec  9 12:58:40 homer named[2088]: zone dnssec.consulintel.euro6ix.net/IN: sending
notifies (serial 2002120902)
```

We checked that the server resolves well the transferred zone.

If we change the shared key, we see in the slave logs:

```
Dec  9 13:05:52 homer named[2193]: zone dnssec.consulintel.euro6ix.net/IN: refresh:
failure trying master 2001:800:40:2a21:205:1cff:fe0a:e2c3#53: tsig indicates error
```

### 4.4.3 DNSSEC Local Tests

**A - Servers configuration:**

Each secured zone has a key pair associated with it: the **private key**, which is stored in a safe place in the server, and the **public key**, which is advertised as a new type of record, the **KEY** record.

The **SIG** record stores the private key in the server.

**BART:**

We **generated the key pair** for dnssec.consulintel.euro6ix.net zone:

```
#>dnssec-keygen –a RSA –b 512 –n ZONE dnssec.consulintel.euro6ix.net.
```

The public key is written to the file "Kdnssec.consulintel.euro6ix.net+001+49125.key" and the private key is written to "Kdnssec.consulintel.euro6ix.net+001+49125.private".

In order to sign our zone, we first add the KEY record, in the *.key file, to the zone date file:

```
#>cat "$INCLUDE Kdnssec.consulintel.euro6ix.net.+001+49125.key" >>
dnssec.consulintel.euro6ix.net.zone
```

This tells the signer program which key to use to sign the zone.

We used *dnssec-signzone* to sign our zone:

```
#>dnssec-signzone –o dnssec.consulintel.euro6ix.net.
dnssec.consulintel.euro6ix.net.zone
```

As the server daemon does not read the "named.conf" to know which zone the file describes, we had to use the parameter *–o dnssec.consulintel.euro6ix.net*.

This program automatically calculates the **NXT records**.

Now we had a new zone data file (note it has only write and read permissions for root, the user who generated it):

```
-rw-------    1 root     root   dnssec.consulintel.euro6ix.net.zone.signed
```

We had to change the /etc/named.conf zone entry:

```
zone "dnssec.consulintel.euro6ix.net" IN {
       type master;
       file "dnssec.consulintel.euro6ix.net.zone.signed";
};
```

**HOMER:**

Nothing to configure, by now.

**B - Running the tests:**

First we have to clarify that a DNSSEC-capable server include DNSSEC records (SIG, NXT and KEY) in a response only if the query indicates that it can handle DNSSEC. To indicate this capability a special flag in a *pseudosection* of the header is used.

Also, at this moment the **caching forwarder** is the only architectural element for which DNSSEC verification is implemented. There are no applications that will do their own verification yet, nor are there stub-resolvers that are able of chasing and verifying resource records.

The caching forwarder will not do cryptographic verification of zones it is authoritative for. You always will get an answer from it because it assumes that the data from disk is secure.

Let's try to request the signed info from our DNSSEC configured server (BART):

```
#> dig +dnssec +norec any prueba1.dnssec.consulintel.euro6ix.net

; <<>> DiG 9.2.0 <<>> +dnssec +norec any prueba1.dnssec.consulintel.euro6ix.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20173
;; flags: qr aa ra; QUERY: 1, ANSWER: 4, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;prueba1.dnssec.consulintel.euro6ix.net.        IN ANY

;; ANSWER SECTION:
prueba1.dnssec.consulintel.euro6ix.net. 172800 IN CNAME
ns3.dnssec.consulintel.euro6ix.net.
prueba1.dnssec.consulintel.euro6ix.net. 172800 IN SIG CNAME 1 5 172800 20030109155815
20021210155815 49125 dnssec.consulintel.euro6ix.net.
0oDf94tF14DG4rqH9bpjXT7TIdOtf1R77slWPVX0IssFvmGfSmbwjZ9S
+hJUMPbOpYVhS8OhUXXX34RK3rxZxg==
```

```
prueba1.dnssec.consulintel.euro6ix.net. 172800 IN NXT dnssec.consulintel.euro6ix.net.
CNAME SIG NXT
prueba1.dnssec.consulintel.euro6ix.net. 172800 IN SIG NXT 1 5 172800 20030109155815
20021210155815 49125 dnssec.consulintel.euro6ix.net.
PyNvT1ZtH49ohamsOkXH9Jlq6Yd1nQGtZ2z9e76Fmf1+vaeCkLb1rImR
dMYOdL4f/dV5Ce+ChITYIViwllKJHw==

;; AUTHORITY SECTION:
dnssec.consulintel.euro6ix.net. 172800 IN NS    ns2.dnssec.consulintel.euro6ix.net.
dnssec.consulintel.euro6ix.net. 172800 IN NS    ns1.dnssec.consulintel.euro6ix.net.
dnssec.consulintel.euro6ix.net. 172800 IN SIG   NS 1 4 172800 20030109155815
20021210155815 49125 dnssec.consulintel.euro6ix.net.
VxZy2o4zz0Z6gnrpbMc3Ojb9boOKdIpfKZ5r71I6p4Xt3EeCFXThnM9F
fdP9wpCtHL0M1AezTw6HijiVzZzm+g==

;; ADDITIONAL SECTION:
ns1.dnssec.consulintel.euro6ix.net. 172800 IN AAAA 2001:800:40:2a23:5054:5ff:fef6:77df
ns2.dnssec.consulintel.euro6ix.net. 172800 IN AAAA 2001:618:10:2a03:5054:5ff:fef6:7487
ns1.dnssec.consulintel.euro6ix.net. 172800 IN SIG AAAA 1 5 172800 20030109155815
20021210155815 49125 dnssec.consulintel.euro6ix.net.
oXGG3Quy/fGJ4dYThN/4VLpZ7eMGmH25jJcE9XstH837ESLYh5OhaM8A
QzWT+m3LbMruIDxxDt1mgfjpt29baQ==
ns2.dnssec.consulintel.euro6ix.net. 172800 IN SIG AAAA 1 5 172800 20030109155815
20021210155815 49125 dnssec.consulintel.euro6ix.net.
C7GZ61OQrWuI0RbLWSdUt2VguvWkce+npmXku8pl+POT4gIYfqn32mJ0
vDxgxPHMMYyBBN1+TQ8lpnpY1V4JoA==

dnssec.consulintel.euro6ix.net. 172800 IN KEY   256 3 1
AQPTtWhIMNsuTeC0xQwJuMADJCANYQEHa0KW1Gd5QnOThH17HpLr/uDm
KfcV3ttviQxUVGNwfPHTwBZ+HGN5v+X3

;; Query time: 8 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Dec 11 09:36:20 2002
;; MSG SIZE  rcvd: 937
```

Some comments:

- We explicitly asked for DNSSEC response with +dnssec statement.
- To simulate a DNS server request we used the +norec parameter, to deactivate the recursion.
- We have the SIG records for many RRs, and also the KEY RR.

The DNS server could do two things with this info:

- Trust on the received KEY RR, and verify the received RRs.
- Ask the upper domain level to verify the subdomain KEY. This is not our case, as the parent server did not sign the KEY used.

In order to trust the received KEY RR, the server would need the trusted-keys statement, which makes the dnssec.consulintel.euro6ix.net domain a security root, below which our nameserver can verify any secure data.

As said in [9]: “The public key for any security root must be present in the configuration file’s trusted-keys statement…”. So, we would add to /etc/named.conf:

```
trusted-keys {
        dnssec.consulintel.euro6ix.net. 256 3 1
"AQPTtWhIMNsuTeC0xQwJuMADJCANYQEHa0KW1Gd5QnOThH17HpLr/uDm KfcV3ttviQxUVGNwfPH
TwBZ+HGN5v+X3";
```

```
};
```

### 4.4.4 TSIG+DNSSEC Local Tests

The use of TSIG or DNSSEC has advantages and disadvantages that must be taken into account at the time of using them.

TSIG:

- Uses shared secrets, what becomes less scalable.
- If one of the servers is "broken" the TSIG will be accessed.
- It is well suited for server-to-server communication: zone transfer, notify and an updater and a name server.

DNSSEC:

- Uses public/private keys, what solves the problems of scalability and "server security".

Based on this, we have tried a configuration that uses TSIG for zone transfers between our servers and DNSSEC to sign the zone data at the master server (BART). This test is the result of using the two configurations mentioned above at the same time.

In /etc/named.conf files we had:

**BART named.conf file:**

```
key bart-homer.dnssec.consulintel.euro6ix.net. {
    algorithm hmac-md5;
    secret "NG9NTSDHeSE3lRthncF1ww==";
};
zone "dnssec.consulintel.euro6ix.net" IN {
    type master;
    file "dnssec.consulintel.euro6ix.net.zone.signed";
    allow-transfer {
            key bart-homer.dnssec.consulintel.euro6ix.net.; };
};
```

**HOOMER named.conf file:**

```
key bart-homer.dnssec.consulintel.euro6ix.net. {
    algorithm hmac-md5;
    secret "NG9NTSDHeSE3lRthncF1ww==";
};
server 2001:800:40:2a21:205:1cff:fe0a:e2c3 { # bart
    keys {bart-homer.dnssec.consulintel.euro6ix.net.; };
};
zone "dnssec.consulintel.euro6ix.net" IN {
    type slave;
    file "dnssec.consulintel.euro6ix.net.zone.signed.slave";
    masters {2001:800:40:2a21:205:1cff:fe0a:e2c3; #bart.dnssec.consulintel.euro6ix.org
       };
};
```

## 4.5 Inter-Partner Test-Bed

This section describes the tests done over Euro6IX network. We start by describing the network scenario and the basic configuration used on each server. Later we detail the specific tests done, together with the configurations used and their associated commands and log messages.

### 4.5.1 Inter-Partner Tests Scenario

For inter-partner tests we used the delegated subdomain:

**sigz.euro6ix.org**

This domain was the security root, under which a subdomain was delegated to each partner participating in the tests: umu.sigx.euro6ix.org, consulintel.sigz.euro6ix.org and upm.sigz.euro6ix.org.

Figure 4-5 shows the inter-partner test configuration used.


**Figure 4-5: DNSSEC inter-partner test network**

### 4.5.2 TSIG Inter-Partner Tests

**A - Servers configuration:**

First we talked with UMU to tell them our key (ns1-ns2.consulintel.sigz.euro6ix.org.) and to know about their key. We had one key for each zone file to be transferred.

Both servers (now on UMU-NS1 and CONS-BART) are masters for their zone and slave of the other's zone.

In our server's /etc/named.conf we have got:

```
key ns1-ns2.umu.sigz.euro6ix.org.{
        algorithm hmac-md5;
        secret "pmYOV9SbEXv7nPwuq35o1Q==";
};
server  2001:800:40:2cff::5 { #ns1.umu.sigz.euro6ix.org
        keys {ns1-ns2.umu.sigz.euro6ix.org.; };
};
key ns1-ns2.consulintel.sigz.euro6ix.org.{
        algorithm hmac-md5;
        secret "U9ogLza92JMCCbgj239V9g==";
};

...
```

```
zone "consulintel.sigz.euro6ix.org" IN {
       type master;
       file "consulintel.sigz.euro6ix.org.zone";
        allow-transfer {
               key ns1-ns2.consulintel.sigz.euro6ix.org.;
       };
};
zone "umu.sigz.euro6ix.org" IN {
       type slave;
       file "umu.sigz.euro6ix.org.zone";
       masters {2001:800:40:2cff::5;
        };
};
```

**B - Running the tests:**

When we started our name server, CONS-BART, we saw that everything worked properly:

```
Dec 12 16:33:51 bart named[11202]: zone consulintel.sigz.euro6ix.org/IN: sending
notifies (serial 2002121214)
Dec 12 16:33:52 bart named[11202]: zone umu.sigz.euro6ix.org/IN: transfered serial
200210286
Dec 12 16:33:52 bart named[11202]: transfer of 'umu.sigz.euro6ix.org/IN' from
2001:800:40:2cff::5#53: end of transfer
Dec 12 16:37:01 bart named[11202]: client 2001:800:40:2cff::5#35150: transfer of
'consulintel.sigz.euro6ix.org/IN': AXFR-style IXFR started
```

### 4.5.3 DNSSEC Inter-Partner Tests

**A - Servers configuration:**

First we signed the sigz.euro6ix.org zone. To generate the public and private keys, each one in its own file:

```
#>dnssec-keygen -a RSA -b 1024 -n ZONE sigz.euro6ix.org.
```

The file Ksigz.euro6ix.org.+001+41995.key contains the public key, which is added to the zone file sigz.euro6ix.org.zone.

The private key is in the file Ksigz.euro6ix.org.+001+41995.private.

To sign the zone file:

```
#>dnssec-signzone -o sigz.euro6ix.org. sigz.euro6ix.org.zone
```

Now we have got a new zone file: sigz.euro6ix.org.zone.signed, so we had to change the /etc/named.conf file in CONS-BART:

```
zone "sigz.euro6ix.org" IN {
       type master;
        file "sigz.euro6ix.org.zone.signed";
};
```

The next step is that the server authoritative for sigz.euro6ix.org, and that is our *security root*, signs the subdomains servers' keys.

UMU-NS1 sent us their keyset: keyset-umu.sigz.euro6ix.org

```
$ORIGIN .
$TTL 3600       ; 1 hour
```

```
umu.sigz.euro6ix.org    IN KEY  256 3 1 (
                                AQPx0nMWl/fmM5OUjWg9EUGb8m5lMg6hg+U+S5/gJRkd
                                wJLOmTv+fjKq4Vkl/HkmZpWPhG1hFZWANpKBLmGkN9he
                                0Vqliq7K3R0CwjgAM4GRB3/8NS0FunWupApvwb1nwzBb
                                JQF5QD1c1hBEWENv5PJJ1AlkKbhM2f45XSVnw8+fqw==
                                ) ; key id = 53151
                        SIG     KEY 1 4 3600 20030111175504 (
                                20021212175504 53151 umu.sigz.euro6ix.org.
                                hFokYGhXXLKqxI/tQSCv4DeaiKp9vs+ft7aY1y8K/Oj9
                                YDYMQfZPgaiqKA9fOVrWnsETB07py6bkOSj4dB4cHose
                                W54vtn23ZG1TIPP3/i7Bt19/ntDy9yV80haeuHCsfYTu
                                KbPQ0Mk38NNQJZrcnlgGBF7YweVR6w6kkYW05+U= )
```

We signed their keyset:

```
#>dnssec-signkey keyset-umu.sigz.euro6ix.org  Ksigz.euro6ix.org.+001+41995.private
```

We obtained the file signedkey-umu.sigz.euro6ix.org., which we sent back to UMU-NS1.

UMU-NS1 includes the signed key in the zone file and signs it (dnssec-signzone).

We also wanted to sign the zone consulintel.sigz.euro6ix.org. We made the following:

```
#>dnssec-keygen –a RSA –b 1024 –n ZONE consulintel.sigz.euro6ix.org.

#>dnssec-makekeyset -t 172800 Kconsulintel.sigz.euro6ix.org.+001+41993.key
```

Then we signed the keyset:

```
#>dnssec-signkey keyset-consulintel.sigz.euro6ix.org.
Ksigz.euro6ix.org.+001+41995.private
```

First we added the signed key to the zone file:

```
$INCLUDE signedkey-consulintel.sigz.euro6ix.org.
```

Now the zone can be signed:

```
#>dnssec-signzone -o consulintel.sigz.euro6ix.org consulintel.sigz.euro6ix.org.zone
```

We had to change the zone statement in the authoritative server:

```
        file "consulintel.sigz.euro6ix.org.zone.signed";
```

One important thing that must be taken into account is the configuration of the master's key for a slave, in the chain of trust of DNSSEC.

For example, we first signed the sigz.euro6ix.org zone. We had got:

```
umu.sigz.euro6ix.org.   172800  IN NS   2001:800:40:2cff::5.sigz.euro6ix.org.
                        172800  KEY     49408 3 3
                        172800  SIG     KEY 1 4 172800 20030111161217 (
                                        20021212161217 41995 sigz.euro6ix.org.
                                        MOH4MsJkHhLpb1csJt9qTu+p+dg4oTat9rNd
                                        OGTIkqc5RT+yuOgjesOaVkaImUpch5TewO9o
                                        iLPuaokfQwx/fXQv9Hj4DhGaXJ3cJfm/OJlw
                                        ajVBWyUyid0NpwYbf4IbOxCEPe5BHj5UnH6t
                                        4a5CAz3lfBKmfwVh+MWuoFTIM/4= )
```

The KEY RR is what is called a **null key**, what means that this zone is verifiable insecure. The resolvers that follow the chain of trust via sigz.euro6ix.org, will consider the umu.sigz.euro6ix.org zone not to be secure.

As long as the NULL key is in the parent's zone, the child is not secured. But **only after** the child has included the KEY with the parental signature, the parent should change the NULL with the children's KEY[1].

So the KEY statement in the sigz.euro6ix.org's zone file should be:

```
umu.sigz.euro6ix.org.   172800  IN NS   2001:800:40:2cff::5.sigz.euro6ix.org.
                        172800  KEY  256 3 1 (
                                AQPx0nMWl/fmM5OUjWg9EUGb8m5lMg6hg+U+S5/gJRkd
                                wJLOmTv+fjKq4Vkl/HkmZpWPhG1hFZWANpKBLmGkN9he
                                0Vqliq7K3R0CwjgAM4GRB3/8NS0FunWupApvwb1nwzBb
                                JQF5QD1c1hBEWENv5PJJ1AlkKbhM2f45XSVnw8+fqw==
                                ) ; key id = 53151
                        172800  SIG     KEY 1 4 3600 20030111175504 (
                                20021212175504 41995 sigz.euro6ix.org.
                                NLF+AAz8nult7YmeukAGSXthS8SAKKcVyeIdY/0nN27O
                                FWdyn/vpY/P7oj+sLJ5JoU6N3QPXIS82leKHgVW8T1n3
                                JpmAfiOUEts497qGygoQuPctYv0Zo/qRYbtDYJ3RY2AB
                                   608kZEfzzX6njF2AuXf0B7F/rOfFPJns20N5vIk= )
```

**B - Running the tests:**

First of all we tested that the signed zone sigz.euro6ix.org was working properly.

UMU-NS1 added to their /etc/named.conf:

```
trusted-keys {
       "sigz.euro6ix.org." 256 3 1
"AQPdsgB8HRAMrPSgZQEYlrLFM2Xk/1/A3eRxV0Y0gvG+qAt8lyB1S8y5Dk6Fq/vXCVS3Sq7jieTTxwV5ClxNh
0x2X5avaDclTgIRPwIf1LS2LaBrYWBI2s2fh0bPAcuWiBadA1V3pbS+jCIwBfwQL7zg9WLddedd6A3Rip69lqQ
L0w==";
};
```

In a forwarder, once a trusted-key has been configured, data from that zone or its subzones will be verified by the caching forwarder. When the data have been verified, the forwarder sets the **ad bit**.

We made a query to their server and checked that the ad bit was active in the flags section.

```
#>dig @2001:800:40:2cff::5 aaaa ns1.sigz.euro6ix.org +dnssec

; <<>> DiG 9.1.0 <<>> @2001:800:40:2cff::5 aaaa ns1.sigz.euro6ix.org +dnssec
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15417
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;ns1.sigz.euro6ix.org.          IN      AAAA

;; ANSWER SECTION:
ns1.sigz.euro6ix.org.   172787  IN      AAAA    2001:800:40:2a21:205:1cff:fe0a:e2c3
```

```
ns1.sigz.euro6ix.org.   172787  IN      SIG     AAAA 1 4 172800 20030111161217
20021212161217 41995 sigz.euro6ix.org.
nF5ACIeAQhL8WA3IHtCQepfHiS2jaxS9OUr9h6B9qm6nItmsEC/fo7nT
7xdbWVylPDatbXhmonSyMZ/LzM+3opAsd5BCEM+HEi8V6+dJ5JTrckIM
utl26iW5R+AwKSvFNOGsRXyvjTJKeVqjFPxx8XKLVL/YaqcXXGhW/j+j GWo=

;; Query time: 97 msec
;; SERVER: 2001:800:40:2cff::5#53(2001:800:40:2cff::5)
;; WHEN: Thu Dec 12 18:14:24 2002
;; MSG SIZE  rcvd: 253
```

Next we tested if everything was working fine with the two secured subdomains: umu.sigz.euro6ix.org and consulintel.euro6ix.org.

As said above only a forwarder server with the appropriated trusted-keys statement is able to check the signatures and replay with the ad bit activated. For this reason we used another server to test umu and consulintel's subdomains.

### 4.5.4 TSIG+DNSSEC Inter-Partner Tests

The DNSSEC tests that were carried out used the existent configuration for TSIG. So The DNSSEC tests were, in fact, DNSSEC+TSIG tests.

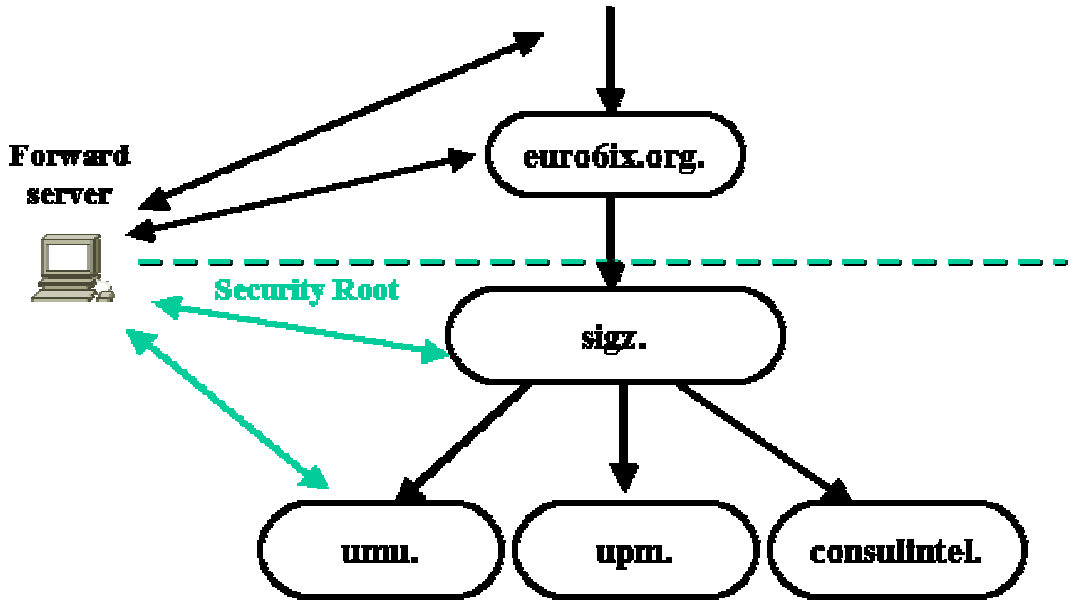


**Figure 4-6: DNSSEC hierarchy**

[1] This is what literature said. We found that when we changed sigz.euro6ix.org zone and "resigned" it, the NULL keys disappeared. The signing software detected the file containing the umu.sigz.euro6ix.org KEY and didn't add any KEY RR. Without NULL KEY it's implicit that the subzone is secure.

# 5. SUMMARY AND CONCLUSIONS

Although basic security extensions to the Domain Name System are becoming mature, standardization work is still in progress to add new extension or make slight modifications to the existing ones [10]. The reference implementation used for testing security extensions in Euro6IX, BIND9, implements most of the basic functionalities. However, BIND9 does not support DNSSEC completely, and lacks some important advanced administration tools, for example, to re-sign of zones, to do TTL management, etc.

Administrative processes are very important in secure environments and have to be considered carefully (key management, TTL, etc.). As a consequence, the administration of a secure DNS service is more complicated than today's insecure service. But there is a strong demand for securing the communications within the DNS system: a secure DNS based on DNSSEC extensions is the way for a lot of applications that need a global, hierarchical, distributed and secure database.

The creation of a DNS emulation environment based on BIND has shown to be very useful to speed up the process of learning how to configure DNS and specially to test security extensions over complete DNS scenarios, without the complexities associated with the use of several machines.

The adaptation of BIND name server to IPv6 seems to work well for simple scenarios. However, the lack of support for binding to specific IPv6 interface address is shown as an important drawback to use BIND in real configurations, and specially has complicated the development of the DNS emulation environment.

The basic tests carried out, first locally over partner's test-beds and later over Euro6IX network, have allowed us to gain experience in the configuration and management of a secure DNS service, as well as derive some preliminary conclusions:

- The importance of **time synchronization**, not just in DNSSEC but in all security architectures.

- DNSSEC produces an important increase in the amount of data stored in zone files, as well as in the size of messages interchanged and the processing time required. This must be taken into account when designing and dimensioning the service.

- The **caching forwarder** is the only architectural element for which DNSSEC verification is implemented at present. It will not do cryptographic verification of zones it is authoritative for.

- The configuration of servers using DNSSEC is not very difficult. The tools that come with BIND distribution make it easier. The difficulty comes from the management of the whole system: changes are propagated.

- DNSSEC is sensible to the lack of connectivity between servers, as the chain of trust is broken. One possible solution is to have trusted-keys statements of slave zones.

Finally, although the generation of recommendations for the deployment of a secure DNS service over Euro6IX network will be one of the main activities for next year inside this subactivity, some preliminary ideas, based on the actual status of BIND implementation, can be outlined.:

- Deployment should probably begin limited to a **secured subdomain** under one of the Euro6IX domains (for example, sigz.euro6ix.org). Each partner should have a subdomain under it (for example, upm.sigz.euro6ix.org). In a later stage, when management procedures are clear and matured, security could be extended to all Euro6IX domains.

- Secured subdomains could be added to existent DNS servers or to new ones specifically deployed to serve those subdomains. During pilot experiments the second option should always be preferred to avoid instabilities.

- DNS servers under secure zones must use DNSSEC for signing their zone information. As the root and gTLD servers actually do not use DNSSEC, the “chain of trust” must begin in the highest secure domain under Euri6IX domains. If during the life of the project a secure root server is available, effort should be invested to extend DNSSEC experimentation to include it.

- Verification of authenticity of information should be based on independent **cache forwarding** servers, as they are the only ones already available to carry out that function. Each partner should install one or more of this servers with the trusted key of the highest secure domain (sigz.euro6ix.org or, later, euro6ix.org) and configure the revolvers of end systems to point to them.

- DNS servers should use TSIG for zone transfers. As TSIG and DNSSEC can be used independently, TSIG based secure transferences could be put in place before DNSSEC is deployed.

# 6. FUTURE WORK

Work invested in DNSSEC subactivity during the second semester of the first year has been basically devoted to get experience on the configuration and management of DNSSEC and to set-up the necessary infrastructure to make the tests.

The main activity during the second year of the project will be the deployment of a secure DNS service over the Euro6IX network. For that purpose, during the first months of year 2, a limited service restricted to partners participating in this subactivity will be deployed, in order to mature the management procedures and generate the documentation and recommendations necessary for extending the service to the whole network under the scope of WP3. Experiences from other DNSSEC initiatives in progress, for example [11] [12], will be reused.

Moreover, effort will be invested also in:

- **Improvement of DNS Emulation Environment.**

  Solutions for the problems already described will be studied and implemented, in order to have a working DNS emulation tool for IPv6. Besides, all the scripts will be improved to easy the use of the environment: the present version is just a "proof of concept" implementation.

  Finally, a set of good examples, including secure, insecure and mixed configurations will be generated and distributed, in order to help in the better understanding of IPv6 DNS particularities and facilitate network administrators the transition to IPv6.

- **DNSSEC as an IX service**

  Studies will be initiated to see how DNSSEC services could be integrated in IXs to improve their efficiency. The process of DNSSEC resolving is costly; the transaction security mechanisms are expensive. The transaction signatures (TSIGs) based on symmetric techniques, and public-key signatures (SIG(0)) increase the size of DNS messages and the latency. In view of these disadvantages a possible improvement is to have a DNSSEC resolver as an IX service.

  Considering Figure 6-1, the local resolver that is located in the IX will query and receive all messages decrease the latency as well as to minimize the round trip time with regard to the schema of the Figure 2-3.

  Naturally this schema presents an IX service that can be attractive for a client or an ISP. Others schemas are possible, for example, one is to have a local resolver and another consist in having a resolver located in the ISP of the client.

  Another justification is that the IX needs a DNSSEC to resolve the names of the IX services (www, ftp, etc.). But this justification introduce to DNSSEC as an element of the IX deployment, and not as a service.

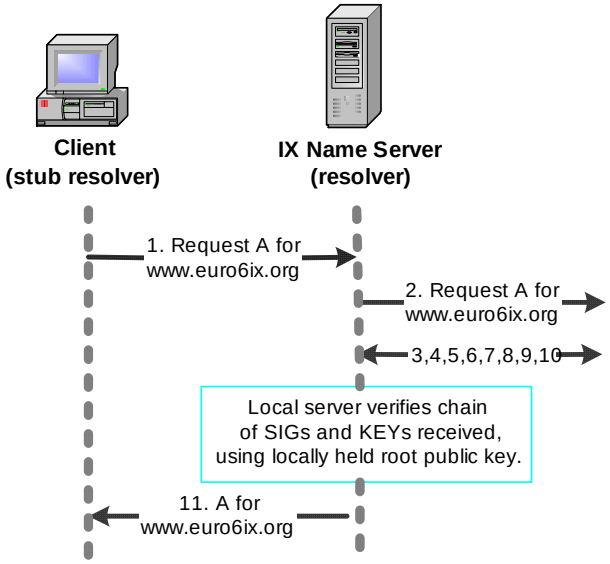


**Figure 6-1: DNSSEC resolving (resolver located in the IX)**

- **Approach to publish certificates using DNSSEC**

  A secure DNS service could play an important role as a repository of digital certificates. A preliminary study has been already started to show how DNSSEC can be used for that purpose by the updating DNS data dynamically. An example can introduce you to better understanding of the idea proposed. That is, a Dynamic Host Configuration Protocol (DHCP) server could add new DNS data when a new IP address is granted to a machine that is connecting to the network. In this manner, using a DNS dynamic update, we are sure that all machines will have name assigned.

  The CERT resource record can be used to store certificates in the DNS. It is intended that personal public keys should be stored in the DNS using the CERT record, and not by using the KEY record. A PKI could add, modify and delete CERT records of a zone using dynamic updates in the DNS. In view of the importance of updates, the updates have to be secure, so using DNSSEC.

  Figure 6-2: shows a schema to publish certificates using DNSSEC. A client generates a certificate request with email "fgarcia@sigz.umu.euro6ix.org" in the PKI (for example, located in an IX) through https. The PKI generates the corresponding certificate and updates in the DNS server that administer the domain sigz.umu.euro6ix.org adding the certificate for "fgarcia.sigz.umu.euro6ix.org". The user gets the certificate with a DNS query for CERT fgarcia.sigz.umu.euro6ix.org to the local resolver. The local resolver gets the certificate from the DNS server for the domain sigz.umu.euro6ix.org.

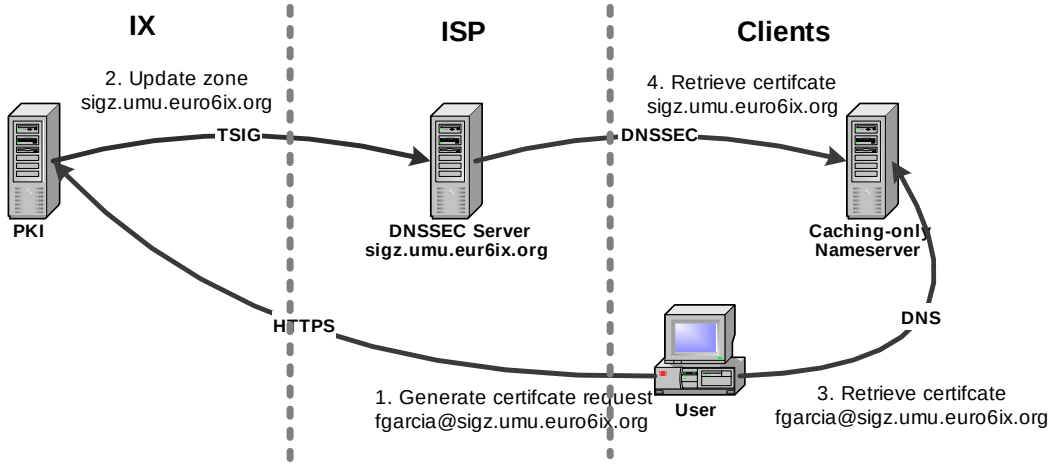


**Figure 6-2: Publishing digital certificates using DNSSEC**

TSIG is used to get a secure communication between PKI and DNS server, and DNSSEC between the resolvers. There are multiples combinations and simplifications of this schema. This schema is a version preliminary and an approach to future work.

# 7. REFERENCES

[1] DNS and BIND. Paul Albitz and Cricket Liu. Fourth edition. O'Reilly, 2001.

[2] Domain Name System Security Extensions. D. Eastlake. RFC 2535. March 1999.

[3] IETF's DNS Extensions Working Group, http://www.ietf.org/html.charters/dnsext-charter.html

[4] Secret Key Establishment for DNS (TKEY RR). D. Eastlake. RFC 2930. September 2000.

[5] Secret Key Transaction Authentication for DNS (TSIG). P. Vixie, O. Gudmundsson, D. Eastlake and B. Wellington. RFC 2845. May 2000.

[6] Open Source Initiative. http://www.opensource.org

[7] BIND DNS Server via Internet Software Consortium (ISC), http://www.isc.org/products/BIND/

[8] Ethereal Protocol Analyzer. http://www.ethereal.com

[9] BIND 9 Administrator Reference Manual

[10] R. Arends, "Protocol Modifications for the DNS Security Extensions", draft-ietf-dnsext-dnssec-protocol-00, October 2002.

[11] DNSSEC Operational HOWTO. Olaf M. Kolkman. RIPE NCC. http://www.ripe.net/disi/Course/TechCourse.pdf

[12] NTnet Labs DNSSEC resources. http://www.ntnetlabs.nl/dnssec