Chapter 6
Challenges and Solutions for DNS Security in IPv6

Hosnieh Rafiee
Hasso Plattner Institute, Germany

Martin von Löwis
Hasso Plattner Institute, Germany

Christoph Meinel
Hasso Plattner Institute, Germany

ABSTRACT

The Domain Name System (DNS) is a necessary component of the Internet that allows hosts on the Internet to communicate with other hosts without needing to know their cryptic IP addresses. When this protocol was first introduced it did not contain robust security features because scalability was an issue. One of the useful features added to DNS was the DNS update mechanism that allowed other hosts to dynamically change DNS entries. This feature, though, exposed new vulnerabilities to DNS servers which necessitated the implementation of new security protocols. Some of the security protocols introduced to address these issues were Transaction SIGnature (TSIG) and DNS Security Extension (DNSSEC). Although, in IPv4, these mechanisms did resolve most of the security issues dealing with authentication between a node and a DNS server, they are not viable in IPv6 networks. This is because the Neighbor Discovery Protocol (NDP) introduced to organize the large IPv6 address space automatically does not support DNS authentication or have an option for secure DNS updating. In this chapter, the authors first explain the common approaches used in IPv4 to address these security issues. Then they explain the differences between the use of these approaches in IPv4 and IPv6, where the focus is on new research with regard to authentication mechanisms between hosts and DNS servers.

DOI: 10.4018/978-1-4666-4514-1.ch006

INTRODUCTION

DNS (Mockapetris, 1987) establishes a naming system for computers, or any other service or device, connected to a network. Without the DNS protocol, Web addresses would become long, confusing, and difficult to remember. The importance of DNS lies in how it makes the Internet and other networks easier to use. It does this by translating domain names into IP addresses. For example, the domain name www.example.com might translate to the IP address 192.168.204.6. Therefore, changing an IP address of a server, such as Web or email, would have a profound effect on a large number of users and systems that make use of the services on those servers on the Internet.

Even though DNS is a very critical element of the Internet, it only supports basic security mechanisms. Also, new DNS functions, such as Dynamic DNS (DDNS), open up new security issues concerning DNS, such as to how to prevent attackers from changing DNS records–in other words, how to authenticate the host's desire to change Resource Records (RRs) on DNS servers. To address this problem, two different protocols were introduced: Transaction SIGnature (TSIG) (Vixie, Gudmundsson, Eastlake 3rd, & Wellington, 2000) and DNS Security Extension (DNSSEC) (Arends, Austein, Larson, Massey, & Rose, 2005). The extensions to these security protocols could thus resolve the authentication problems in Internet Protocol version 4 (IPv4). But the main problem that exists with the IPv4 network is a lack of IP addresses. According to the IANA exhaustion counter, the last blocks of IPv4 addresses have already been given to the local Internet registries. It is for this reason that the next generation of Internet Protocol, i.e. IPv6 (Deering, & Hinden, 1998) was proposed. The number of unique IPv6 addresses is 2^{128-32} times greater than those of IPv4. To organize this large address space, two different mechanisms have been proposed: Dynamic Host Configuration Protocol (DHCPv6) (Droms, Bound, Volz, Lemon, Perkins, & Carney, 2003) and Neighbor Discovery Protocol (NDP) (Narten, Nordmark, Simpson, & Soliman, 2007). These two mechanisms, together, are known as IPv6 Autoconfiguration. Unfortunately, security, in the DNS update process, is also the main issue with these two mechanisms. For example, when using DHCPv6, no options have been added to the DHCPv6 messages to handle host authentication of the DNS server. Another main problem, with these mechanisms, is the changeable nature of IPv6 addresses. Because of privacy reasons, and in order to prevent attackers from tracking a node in IPv6 networks, the IPv6 addresses are valid only for a short period of time, which is dependent on network policy. Moreover, in one of these addressing mechanisms, i.e., NDP, there is no control over the nodes that can join the IPv6 networks. These unmanageable and temporary addresses create several issues for the updating of DNS records. There is another issue which concerns resolver authentication. Usually, the DNS client (stub resolver) on the client's computer sends its queries to another recursive DNS server in order to recursively query other DNS servers and to translate a name to an IP address. It then sends back the result to this client. The DNS client often does not support any secure mechanisms, like DNSSEC, and thus only relies on the source IP address authentication process. An attacker is thus able to spoof this IP address and then send the wrong response to this client. The attacker will then direct the victim to a computer of his choice which, in fact, might be one of his own servers. On many occasions, like checking a bank account, it is important for users to ensure that the query response received from the DNS server was originated by the real recursive DNS server and has not been spoofed by an attacker.

The main focus of this chapter will be on the security mechanisms needed to ensure a securer DNS update and on the proper authentication process. Several books already discuss DNS implementation and configuration. We will therefore just briefly mention this background information. The remaining sections of this chapter are organized as follow:

First – DNS concepts and functions are briefly explained. Second – IPv6 autoconfiguration mechanisms, threats, and the differences between DNS in IPv4 and IPv6 are explained. Third – divides DNS functions into two categories, introduces DNS threats in each category, describes the current security mechanism in place for IPv6, introduces the most recent research on DNS security for IPv6, and explains future trends. The last section summarizes this chapter.

DOMAIN NAME SYSTEM (DNS)

DNS and its Functions

DNS is a hierarchical database that stores data in a particular format in what are called Resource Records (RRs). These RRs are distinguished by their types – MX, NS, AAAA, A, etc. Each RR specifies information about a particular object. The server uses these records to answer queries from hosts in its zone (Mockapetris, 1987).

To provide a host with DNS functionality, it is necessary to install software that implements the DNS protocol. One of these implementations is the Berkeley Internet Name Domain (BIND) that was first introduced in the early 1980s at the University of California at Berkeley. This implementation is also viewed as the reference implementation for the Internet's DNS. The latest versions (version 8+) of this implementation support Dynamic Update which will be explained in the next section.

A DNS-capable host is called a Name Server (NS). Name Servers can be divided into two categories: Authoritative and Recursive.

- **Authoritative:** An authoritative name server provides the answers to DNS queries. For example, it would respond to a query about a mail server IP address or Website IP address. It provides original, first-hand, definitive answers (authoritative answers) to DNS queries. It does not provide 'just cached' answers that were obtained from another name server. Therefore it only returns answers to queries about domain names that are installed in its system configuration.
  There are two types of Authoritative Name Servers:
  - **Master Server (Primary Name Server):** A master server stores the original master copies of all zone records. A host master is only allowed to change the master server's zone records. Each slave server gets updated via a special automatic updating mechanism within the DNS protocol. All slave servers maintain identical copies of the master records.
  - **Slave Server (Secondary Name Server):** A slave server is an exact replica of the master server. It is used to share the DNS server's load and to improve DNS zone availability in cases where the master server fails. It is recommended that there be at least 2 slave servers and one master server for each domain name.
- **Recursive (Something which Repeats or Refers Back to Itself):** A recursive name server responds to queries where the query does not contain an entry for the host in its database. It first checks its own records and cache for the answer to the query and then, if it cannot find an answer there, it may recursively query name servers higher up in the hierarchy and then pass the response back to the originator of the query. This is known as a *recursive query* or *recursive lookup*.

In principle, authoritative name servers suffice for the operation of the Internet. However, with only authoritative name-servers operating, every DNS query must start with recursive queries at the root zone of the Domain Name System and each user system must implement resolver software capable of a recursive operation. In fact, the majority of DNS query responses are generated from the cache of recursive servers, which are responsible for obtaining the IP address of the site or computer you are trying to reach. Many of the security compromises and breaches that occur today are the result of vulnerabilities in the recursive or caching DNS server code.

For example, when a user types something like "www.xxx.com" into his browser, in order to resolve this address, a request is sent to a Name Server. The IP address of this local DNS server is configured either via Dynamic Host Configuration Protocol (DHCP – a network protocol that is used to configure network devices so that they can communicate on an IP network) or by other mechanisms that were explained in the section entitled "DNS for IPv6". The Name Server resolves the address by the use of different types of queries which can be either recursive queries (the DNS server has to reply with the requested information or with an error message because the DNS server cannot provide a referral to a different DNS server [Microsoft Library, 2013]) or iterative queries (the DNS server provides the best answer which can be the resolved name or a referral to a different DNS server [Microsoft Library, 2013]). It will then return that Website's IP address. It is at this time that a connection will be established and the content on that Website will become visible.

Mechanisms to Update DNS

DNS update (Wellington, 2000) is a mechanism for adding, changing, or removing a RR record in a DNS zone file. There are two mechanisms used to update DNS RRs: manual updating and Dynamic DNS Update (DDNS) (Wellington, 2000). In manual updating, an administrator needs to edit the zone file in order to process the modifications. If the DNS service needs to restart in order to apply the modifications, then this would have an adverse effect on DNS performance. During the restart process of a DNS service, the DNS servers (name servers) will be unable to process DNS queries that are asked of it by other hosts on the Internet. To address this problem DDNS was introduced. By the use of this mechanism it is possible for the name server to change one or several records, in one particular zone, with just one DNS update request, while at the same time, being able to respond to user queries. The clients, or servers, can thus automatically send updates to the authoritative name servers in order to modify the records they want to modify. The authoritative name server then checks to make sure that certain prerequisites have been met. The prerequisites contain a set of RRs that must exist on the primary master server in the particular zone that needs to process this update packet. DDNS can also be used, in conjunction with DHCP, to update RR records when a computer's IP address is changed. To do this, clients send update messages, which should contain an additional DHCP option that provides their Fully Qualified Domain Name (FQDN) (Stapp & Volz, 2006), along with instructions to the DHCP server as to how to process the DNS dynamic updates. (For example, if the client name is "hostx" and the parent domain name is "mynetwork.com," then the FQDN is "hostx.mynetwork.com). An example of an application needing DDNS would be a small business, where a static IP address is not available for use by their servers, or the IP addresses are set dynamically by Neighbor Discovery Protocol (NDP) or DHCPv6.

DNS FOR IPV6

Internet Protocol Version 6 (IPv6)

Internet Protocol Version 6 (IPv6) (Deering, & Hinden, 1998) represents the next generation of Internet protocol. The main reason for its creation was due to the exhaustion of IP addresses that currently exists in the current version of the Internet

Protocol, Internet Protocol version 4 (IPv4). IPv4 allows for only about 4.2×10^9 unique addresses worldwide while IPv6 allows for about 3.40×10^{38} addresses – a number unlikely ever to run out. These IPv6 IP addresses are in a hexadecimal format – such as (fe08:1a63:2001:50e9::). It is because of this hexadecimal format that it is difficult to memorize IPv6 addresses. Administrators do not want to have to manually set these IP addresses for the hosts on their network and thus look for other mechanisms for their management. In order to organize such a large address space, two IPv6 autoconfiguration mechanisms were introduced: Dynamic Host Configuration Protocol v6 (DHCPv6) and Neighbor Discovery Protocol (NDP).

IPv6 Autoconfiguration

There are two different types of autoconfiguration mechanisms used in IPv6 – Stateful Autoconfiguration and Stateless Autoconfiguration. Stateful Autoconfiguration, DHCPv6, is the equivalent of IPv4's DHCP but for IPv6. It is used to pass out addresses and service information in the same manner that DHCP does it in IPv4. This is called "stateful" because both the DHCP server and the client must maintain state information in order to keep addresses from conflicting, to handle leases, and to renew addresses, over time. Stateless autoconfiguration allows a host to propose an address which will probably be unique and to offer it for use on the network. Because no server has to approve the use of the address, or pass it out, stateless autoconfiguration is simpler. This is the default mode of operation for most IPv6 systems, which includes servers.

Dynamic Host Configuration Protocol Version 6 (DHCPv6)

DHCPv6 (Droms et al., 2003) is a protocol that can be used to allow a DHCP server to automatically assign an IP address to a host from a defined range of IP addresses configured for that network. DHCPv6 operates in many modes compared to DHCPv4:

- **Stateless Mode:** In combination with stateless IP configuration, DHCPv6 delivers router advertisements to a host with DNS server information along with other information, like options for SIP phones and other services.
- **Stateful Mode:** A host can also configure it's IP address (as with DHCPv4) with DHCPv6
- **DHCPv6 Prefix Delegation (DHCPv6-PD) (Troan, & Droms, 2003):** This is an extension to DHCPv6 that enables an IPv6 host, running DHCPv6 protocol, to ask for a network prefix from that DHCPv6 server. This DHCPv6 server can be an Internet Service Provider ISP).

When a device connects to the IP network, it sends out a Router Solicitation (RS) message. The responding message, a Router Advertisement (RA) message, contains two flags. The O flag is set to indicate the existence of "other" information on the DHCPv6 servers. The M flag indicates the use of managed mode. Managed mode is the action where the client should ask DHCPv6 for an IP address and not configure one statelessly. The big difference between DHCPv4 and DHCPv6 is the way in which the device identifies itself if a host wants to assign the addresses itself, instead of selecting addresses dynamically from a pool.

Neighbor Discovery Protocol (NDP)

Neighbor Discovery (ND) (Narten et al., 2007) enables hosts to discover their neighboring routers and hosts and presents a technique which allows the host to obtain router information from routers. It also enables all nodes on the network to check for the reachability of other neighboring hosts and routers. ND and Stateless Address

AutoConfiguration (SLAAC) (Thomson, Narten, & Jinmei, 2007), together, comprise NDP. This protocol defines five different Internet Control Message Protocol version 6 (ICMPv6) message types, which perform functions for IPv6 similar to those of Address Resolution Protocol (ARP) and Internet Control Message Protocol (ICMP), Router Discovery, and Router Redirect protocols for IPv4.

A NDP-enabled node can configure its IP address automatically as soon as it is plugged into a new network. This newly joined node first generates its Interface Identifier (IID), which is represented by the 64 rightmost 128 bits of the IPv6 address. It concatenates the IID with the local link layer prefix that starts with fe08, and sets this on its network adapter. It then sends a RS message to all neighboring routers requesting router information. Routers respond to this message with a RA message that contains routing information and subnet prefixes (64 bits). The subnet prefix consists of the 64 leftmost 128 bits of the IPv6 addresses. The node then sets its global IP address (as a temporary address) with the subnet prefix obtained from the RA message, and sends a Neighbor Solicitation (NS) message to all nodes on that network in order to prevent the possibility of collisions with its IP address (process Duplicate Address Detection [DAD]). If it does not receive any Neighbor Advertisement (NA) messages after a certain time, (a standard of about 1 second) from any nodes claiming to own its IP address, then it changes the status of this IP address to permanent and starts using it. Otherwise, it will generate a new IID and repeat this process.

The Differences between DNS for IPv4 and IPv6

Even though DNS resides far above the Internet Protocol in the TCP/IP protocol architecture suite, it works intimately with IP addresses. For this reason, changes are necessary to afford it the ability to support the new IPv6. These changes include the definition of a new IPv6 address RR (AAAA) as a replacement for the A RR in IPv4. When a node wants to know the IP address of a sample domain, the response to this query will be obtained by retrieving that IP address from the AAAA RR record on the DNS server.

Another change occurs in Reverse mapping. Reverse mapping maps a particular IP address to a particular host and allows nodes, on the Internet, to look for a domain name associated with an IP address. An IPv4 reverse map uses IN-ADDR.ARPA Pointer Resource Records (PTR RRs) for reverse mapping. For example, if the IP address is 192.168.254.17, the reverse lookup domain name is 17.254.168.192.IN-ADDR.ARPA. On the other hand, IPv6 makes use of the IP6.ARPA domain PTR RRs for reverse mapping. Each hexadecimal digit of the 32-digit IPv6 address (zero compression and double-colon compression notation cannot be used) becomes a separate level, in inverse order, in the reverse domain hierarchy, when the namespace for reverse queries is created. For example, the IPv6 address is 2001:0db8:0000::/48 so the reverse lookup domain name is 0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.

Because IPv4 and IPv6 coexist, and because different formats are used for reverse mapping, the reverse zone files need to be defined separately.

DNS SECURITY

DNS is an essential protocol used on the Internet. When this protocol was first introduced it did not contain robust security features because of scalability issues (Scalability refers to the fact that the database, which must be kept in sync, is distributed over the entire Internet.). In today's environment, security has become a big issue for DNS. Safeguarding the DNS operation is a primary concern to everyone using the Internet today. DNS critical functions can be divided into two categories: reading data from a DNS server and writing data to a DNS server. There are some security issues associated with each of these functions. In the following sections these issues, and the current solution for them, will be described.

Reading Data from a DNS Server

Any host on the Internet can query DNS servers. No authentication is needed to do this. The host needs only to support a client DNS application called a resolver. When a host, such as a client or a mail server, wants to resolve the domain name or IP address of another host on the network, it sends its query to a resolver. The resolver can be a server, located in an Internet Service Provider's (ISP) network, which gives the host access to the Internet. The task of the resolver is to query, recursively, the root DNS server, and other DNS servers, to find the authoritative DNS server. The resolver then sends a query to that name server in order to read the content of RRs on that DNS server. The authoritative DNS server then responds to the query with the relevant content contained in the RRs. After receiving the response from the authoritative DNS server, the resolver forwards this response to the host that initiated the query. Figure 1 shows the process of querying name servers to find a sample domain (example.com). Some hosts do support the DNS resolving process themselves, while others rely on a resolver to resolve the query.

DNS Read Threats

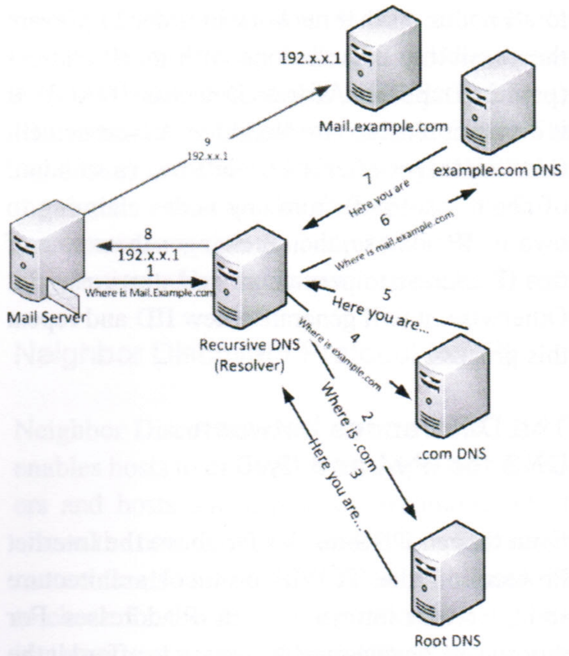
DNS queries are exchanged in plain text. Attackers can easily sniff these messages, manipulate them for their own purposes and, then, forward them on to the resolvers. Resolvers have no way of verifying this data. They forward these DNS responses to the query initiator without authenticating them or ensuring their integrity.

Another problem exists for clients initiating these queries. The basic authentication mechanism uses the source IP address. If the source IP address of the query response is the same as that of the resolver, then the clients' computer will accept that response. The clients thus have no way of ensuring that the query responses were originated by the real resolver and not as the result of having been spoofed by an attacker. The DNS queries are thus vulnerable to several types of attack, which are explained below.

Man in the Middle (MITM) Attack

When a nameserver and a resolver initiate a DNS request and response, an attacker might intercept both, and then attempt to spoof the messages exchanged between the name server and resolver. The attacker reroutes communication between a nameserver and a resolver through the attacker's computer. The nameserver and resolver have no idea that the attack is being perpetrated. The attacker is thus able to monitor and read the traffic before sending it on to the intended recipient. This attack is possible because the DNS queries

Figure 1. Reading data from DNS server



are only authenticated based on the source IP address. The attacker can thus spoof this IP address and initiate his attack. Other types of attack that are included in this category are packet sniffing and cache poisoning.

DNS Cache Poisoning Attacks

DNS cache poisoning is another form of MITM attack. Cache is a very important component of the DNS infrastructure. Many components within the DNS hierarchy maintain their own cache to avoid the necessity of accessing an external server's cache. A resolver nameserver receives a response and caches it for a certain period of time (dependent on network policy) so that further queries for the matching domain can be resolved using data from its cache. Not only does this reduce the amount of DNS traffic, but it makes the resolution process more efficient. An attacker might also respond to a requestor's DNS query with spoofed messages. Whenever a nameserver sends out a query to another nameserver, it verifies the response through the execution of the following steps:

- Ensure that the Query ID (16 bit identifier for the request) in the response matches the one contained in the request.
- Ensure that the response arrives on the same User Datagram Protocol (UDP) port that was used for sending the request.
- Ensure that the question section in the response matches that of the question contained in the query.
- Ensure that additional information contained in the response belongs to the same domain as that which was queried.

When the verification process is successful, the response is accepted and the cache is updated. An attacker can try to guess this data by sending a query for the domain name that it wants to hijack to the nameserver it wants to poison. Knowing that the nameserver will soon reach out to external nameservers for resolution, the attacker starts flooding the nameserver with forged responses. Nameservers are designed to accept the first valid response – the rest of the responses are ignored. The chance of the attacker's valid looking response reaching the nameserver is high if the attacker is able to generate responses that reach the resolver before a valid response does. This process redirects other user's queries, of that domain, to the attacker's computer.

Packet Sniffing

This attack occurs when DNS sends an entire query or response in a single, unsigned, unencrypted UDP packet. Messages sent this way make spoofing easy. An attacker can capture DNS query packets, generate wrong responses, and then send them quickly to the resolver, before the correct response is received from the name server. As no source authentication or data integrity checks are supported, this will not be detected by the resolver.

Transaction ID Guessing

A transaction ID is a 16-bit field which identifies a specific DNS transaction. The transaction ID is created by the DNS request initiator and is copied to the DNS response by a DNS query responder. Using the transaction ID, the DNS client can match responses to its requests. It is not hard to guess, from the DNS request, without having to intercept packets on the LAN, the transaction ID of a DNS response. In practice, the client UDP port and the Transaction ID can be predicted from previous DNS requests/responses (DNS queries). It is common for the client port to be a known fixed value, due to firewall restrictions, or the port number will increase incrementally due to resolver library behavior. The DNS transaction ID generated by a client usually increases incrementally. This reduces the search space for an attacker (Atkins & Austein, 2004).

Distributed Denial of Service (DDoS) Attack

A DDoS attack is an attempt to make a DNS server unavailable in the network. Over the past several months a series of Distributed Denial of Service (DDoS) attacks have victimized DNS root and Top Level Domain (TLD) name server operators. Suppose an attacker targets nameserver A on the Internet. The attacker initiates several DNS request messages, using nameserver A as the source IP address of these messages, for the purpose of initiating a DDoS attack on nameserver A. These messages are sent from different infected computers, called botnets, around the Internet. If the resolvers cannot find a response to that query in their cache, they issue a DNS request message of their own, to the compromised name servers, in order to retrieve the response. The compromised name servers return the DNS responses to the resolvers. The resolvers then send these DNS response messages to the spoofed IP address, i.e., nameserver A. Nameserver A is thus bombarded with many DNS response queries which prevents it from answering new DNS queries. The nameserver A's services would thus not be available to real users.

Secure DNS Read

There are some mechanisms that can be used to secure DNS queries. Two of these are the DNS Security Extension (DSNSEC) (Arends et al., 2005) and the Internet Protocol Security (IPsec) (Kent & Seo, 2005).

DNS Security Extension (DNSSEC)

DNSSEC (Arends et al., 2005) was introduced by the Internet Engineering Task Force (IETF) as an extension to the DNS used in validating DNS query operations. It verifies the authenticity and integrity of query results from a signed zone. In other words, if DNSSEC is available from the requestor client to the resolver/caching name server to the authoritative name servers, then the client has a level of assurance that the DNS query response is signed and trustworthy, starting from the root and chaining all the way down to the domain and sub domains. It uses asymmetrical cryptography. This means that separate keys are used to encrypt and decrypt data in order to provide security for certain name servers with their respective administrators. When DNSSEC is used, all answers include a digital signature. This will prevent DNS spoofing attacks because the attacker does not have the same private key as the server and thus will be unable to sign his own response and send it to the victim. But the problem with DNSSEC is that the signatures are not created on-the-fly because DNS itself does not have access to the keys needed to sign its own responses. Thus, the administrator of that zone should sign each domain and sub domain manually, ahead of time, and then store those signatures in the SIG RRs of the DNS server. Moreover, the zone private key should be stored offline. This is the reason that it cannot fully support the Dynamic Update process. It cannot generate the signature, on-the-fly, in order to respond to real-time queries. Also using DNSSEC cannot guarantee the data's confidentiality because it does not encrypt the data but just signs it.

Internet Protocol Security (IPsec)

The Internet Engineering Task Force (IETF) (Kent & Seo, 2005) proposed the use of IPsec in order to provide access control, data authentication, integrity, and confidentiality for the data that is sent between communication nodes across IP networks. IPsec insures data security at the IP packet level (IP datagram). A packet is a unit of data that is assembled and routed across the network, including a header and payload. Other protocols that assist IPsec in securing packets are:

- **Encapsulating Security Payload (ESP):** A session protocol for data protection that provides confidentiality, authentication, and integrity.
- **Authentication Header (AH):** Provides authentication and integrity.
- **Internet Key Exchange (IKE):** Provides session key negotiation, key management, and Security Association (SA) management.

IPsec can be configured to operate in either of two modes: tunnel and transport mode. When the tunnel mode, the default IPsec mode, is used the entire original IP packet is protected by IPsec. This means that IPsec encapsulates the original packet, encrypts it, adds a new IP header to it and then sends it to the other side. In other words, the entire packet is the payload and this packet is then processed using IPsec. Tunnel mode is most commonly used to encrypt traffic between secure IPsec gateways – such as, between the Cisco router and PIX Firewall, or an end-station to a gateway, where the gateway acts as a proxy for the hosts behind it. Transport mode is used for end-to-end communications – such as, for a host to host IPsec SA. Only the original packet's payload is encrypted and protected using this mode. The IP header is not protected, so an attacker might eavesdrop to find the source and destination of this packet. A secure Telnet or secure Remote Desktop session, from a client to a server, are representative of the IPsec transport mode.

Even though IPsec is a good approach for securing data at the packet level, in practice, because of the complexities involved, it is not easy to configure or to implement. For example, public key authentication used in IKE requires the use, and hence understanding, of X.509 certificates (Cooper, Santesson, Farrell, Boeyen, Housley, & Polk, 2008). Moreover, the secret key distribution mechanism requires an infrastructure whereby the traditional methods for authentication are Pre-Shared Keys and Digital Signatures. The use of a Pre-Shared Key is only feasible when the number of communicating hosts is small. This is why, in practice, IPsec is not used to secure DNS messages. On the contrary, IPsec benefits from the use of DNSSEC records for authentication in IKE processing (Merino, Martínez, Organero, & Kloos, 2006), but this method is beyond the scope of this chapter.

Can Existing Secure DNS Read Mechanisms be used in IPv6?

Two DNS secure read mechanisms were introduced in the prior section. Unfortunately, the human intervention required to apply the configurations needed by these mechanisms makes them difficult to use in both IPv6 and IPv4 networks. For example, pre-configuring all DNSKEYs used in DNSSEC mechanisms, for every root island of security, is not practical because of scalability and key management, in each resolver, are major issues in today's fast-growing Internet. In other words, the current resolver (stub-resolvers) installed on most of the world's computers would need to be replaced with ones that could handle DNSSEC messages. The fact that replies to DNS requests exceeding 512 bytes (due to the use of older preconfigured equipment) are blocked, poses a second problem. Typically, DNSSEC replies are four times larger. The third reason is that zones whose parents do not deploy DNSSEC cannot use a DNSSEC approach. Thus, the communication between a non DNSSEC enabled zone with a DNSSEC enabled zone is not protected and is prone to read attacks as mentioned in the previous sections.

Writing Data to DNS Server

Writing data to a DNS server is the process of adding, changing, or removing a RR record in a zone's master file. This is called a DNS Update. This process can be for the transfer of the entire zone from a master to slave, or updating just a few RRs in the master name server. The old mechanism used to process updates consisted of a manual update process. But this manual process had a negative impact on the performance of DNS servers because of the need for human intervention. Human intervention also opened the door for an increase in DNS attacks that were the result of human error. For example, a long delay could result when an update is needed and the administrator is not available. Another problem, with some implementations of DNS, is that the DNS service will need to be restarted before the changes will take effect. During a restart, DNS servers are unable to process DNS queries. To address these issues the Dynamic DNS update (DDNS) (Wellington, 2000) was introduced. DDNS enables real-time, dynamic updates to entries in the DNS database. By using this mechanism, it is possible for the name server to change one or several records in one particular zone with the use of only one DNS update request, while at the same time, responding to user queries. The clients or servers can thus automatically send updates to the authoritative name servers in order to modify the records they want modified.

DNS Write Threats

DDNS uses a basic protection mechanism in order to prevent other nodes from making unauthorized updates. This is done by checking whether or not the source IP address is the same as that on the list of authorized updaters. But there is a flaw here because attackers can spoof this IP address and update DNS RRs and then redirect all traffic to their desired hosts rather than the intended hosts. The attackers can also execute other attacks – such as, phishing attacks, infection of other computers, Distributed Denial of Service (DDoS) attacks, etc. They can also redirect traffic to the victim's host which will inundate that server with messages and render its service unavailable (DoS). The list of DNS write threats include the vulnerabilities of the Operating System (unauthorized access to zone file), zone file configuration vulnerabilities (human mistakes), Source IP spoofing, DNS update spoofed messages, and Zone file corruption (accidentally).

Vulnerabilities of the Operating System

The bugs (bug – an error or mistake in software code that does not allow the software to perform its task correctly) in a server Operating System (OS) might give attackers access to computers on its network, especially when they are always connected to the Internet and the servers' service is accessible via the Internet. For example, unnecessary open ports in an OS could allow remote code execution.

When an attacker gains access to the OS, he might access critical data, such as a DNS zone file, and update RRs for his own advantage. Overflowing the code execution buffer is another type of attack against the OS. This attack calls a subroutine that returns to a point in the main program defined by the attacker. This is one of many, similar types of OS level attacks, which exploit OS vulnerabilities on which the DNS service is running (Rooney, 2011).

Zone File Configuration Vulnerabilities

When an administrator configures the DNS service, any misconfiguration might allow for the extraction of critical data from a zone file to the attacker's computer or might lead to an improperly configured server.

Source IP Spoofing

If a host wants to update a RR record residing in a zone file on the DNS server, the DNS server checks to see that the source IP address, of the DNS requestor, is the same as that stored in the DNS configuration file. If it is the same, the DNS server will process the update request. An attacker can listen to DNS traffic, intercept an IP address, and then, illegally update a RR in a DNS server. He can then redirect the traffic to the computer of his choice.

Spoofed Message Attack

This type of attack is usually classified as a source IP spoofing attack. In this attack, the attacker listens to the DNS traffic, and when a legitimate host wants to update a RR in a zone file on the DNS server, the attacker intercedes and changes the content of the DNS update and resends it to the DNS server. Since the source IP address is now actually the same as the legitimate host IP address, the update request will be processed by the DNS server.

Zone File Corruption

When the hard disk, where the zone file is stored, encounters a physical problem, which causes a part of data to be lost, a corrupted zone file is the result.

Secure DNS Write

Some security mechanisms and protocols were introduced to address the problems mentioned in earlier sections. TSIG (Vixie et al., 2000) is one such protocol that can be used to secure a Dynamic Update. The remainder of this section will explain the TSIG RR format and the messages exchanged between a client and a server when the DNS update is initialized.

Transaction SIGnature (TSIG)

TSIG is a protocol which provides endpoint authentication and data integrity by the use of one-way hashing and shared secret keys to establish a trust relationship between two hosts that can be either a client and a server or two servers. The TSIG keys are manually exchanged between these two hosts and must be kept in a secure place. This protocol can be used to secure a Dynamic Update or to give assurance to the slave name server that the zone transfer is from the original master name server and that it has not been spoofed by hackers. It does this by verifying the signature with a cryptographic key shared with that of the receiver.

The TSIG Resource Record (RR) has the same format as other records used in a DDNS update request. Some of the fields of this TSIG RR are: Name, Class, Type, Time To Live Resource Data (TTL RDATA), etc. For example, an RDATA field would specify the type of algorithm that would be used in a one-way hashing function. It could be a Hash-Based Message Authentication Code-Message-Digest algorithm (HMAC-MD5.SIG-ALG.REG.INT [HMAC-MD5]), a Generic Security Service Algorithm for Secret Key Transaction (GSS-TSIG), a Hash-Based Message Authentication Code-Secure Hash Algorithm (HMAC-SHA1), HMAC-SHA224, HMAC-SHA256, HMAC-SHA384 or HMAC-SHA512. These algorithms are defined by the Internet Assigned Numbers Authority (IANA). New algorithms should first be registered with IANA prior to their use. Some of the other fields residing in RDATA are; the time the data is signed, the Message Authentication Code (MAC) that contains a hash of the message being signed, and the Original ID which is the ID number of the original message.

Can Existing Secure DNS Write Mechanisms be used in IPv6?

As explained in prior sections, the new IPv6 addressing mechanism, NDP, creates a problem as to how to authenticate a DNS server during the DNS Update process without, or with minimal, human intervention, while staying within the goals of this protocol. The main reason NDP was proposed was to ease the management of the large address space in IPv6 networks and to reduce the need for human intervention in address configuration. This eliminates the need to memorize complex hexadecimal addresses. A node might join an IPv6 network and have its IP address automatically configured by the use of the NDP mechanism, which needs no further administrator intervention. Moreover, privacy is an important issue in IPv6 when nodes on the network need to frequently change their IP address in order to prevent being tracked by attackers. This makes it difficult to authenticate who the update requestor of the DNS RRs is, based solely on the source IP address. Other security mechanism, such as TSIG, need a manual key exchange or signature generation before starting the secure authentication process between the DNS server and a host. In IPv6 networks it becomes harder to apply this authentication mechanism. Although, in IPv6, the manual update process is a major concern, in IPv4 it is an acceptable procedure for the following reasons:

- Using Active Directory (AD) to simplify the authentication process
    - **Advantage:** Nodes are already authenticated so that they can update their DNS records
    - **Disadvantages:** The administrator manually adds the new node to this network.
- The addressing mechanism in IPv4 is not a completely automatic process – it is either totally manual or requires network administrator intervention for DHCPv4 server configuration. These administrators thus exchange the keys required for TSIG, or other current DNS update security mechanisms, between the DNS server and the DNS update requestor.

Two solutions for node authentication, when Secure Neighbor Discovery (SEND) is used with a DNS server, are Modified TSIG and Modified SEND. In Modified TSIG there are two different scenarios in play. One pertains to the authentication of a node, with a DNS server, in order to update the DNS records. The other pertains to the authentication of two DNS servers, such as a slave and a master. SEND can also be used to authenticate other DNS servers on the Internet. The solution offered in this chapter focuses primarily on the first scenario, but it can also be used to resolve the issues stated in the second scenario. To address the stated problem, the addition of an extension to the current TSIG protocol is proposed in order to automate the DNS Update authentication process when SEND is used.

In the second solution, modified SEND, there are again two different scenarios in play here – one where all the hosts are in the same local link with the DNS server and one where the DNS server is located outside the bounds of the local link.

In the following section we will explain the steps necessary to create and verify Update Messages.

Modified TSIG

The TSIG RR can be created by employing the same data used to generate a new IP address in a node – that is, from the key pairs (public/private keys), the output value of CGA generation function (Interface ID), and other required parameters. These values must be cached in the node's memory for later use.

The proposed solution to the Update Request vulnerability issue is presented by the following steps. As a node needs to change its IP address frequently in order to maintain privacy, key pairs and CGA parameters must be cached during the generation of the IP address when using SEND. The following steps show how to generate an IP address.

1. Use a RSA algorithm or other CGA supported algorithms in order to generate key pairs (public/private keys).
   The shared secrets generated in TSIG may be equivalent to key pair generation in the CGA process. When somebody wants to generate

TSIG secrets, he uses the dnssec-keygen comment (in Linux) which then generates a set of .key and .private files in the current working directory. Then, from the content of the .private file, the base-64 string of data is obtained (starting from the word "key") and this is called the shared secret. However, this shared secret is only shared between two hosts, a client and a DNS server, and for each pair of hosts this process should be repeated. The DNS server must also be configured in order to know what key to use for which host. This means that every time a host changes its IP address this process must again be repeated. But when using the modified TSIG, the same key pairs used in the initial IP address generation are used in the modified TSIG process. These keys can be generated on the fly, for first time IP address generation, using the RSA algorithm or other types of algorithms, such as ECC (Brian, 2009). This way of using CGA is called the CGA-TSIG algorithm, which will be added as an algorithm type to TSIG RDARA. The algorithm which is used in the signed message, and key generation process, can be added to the algorithm type field, which is a part of the CGA-TSIG Data field in the other Data section of TSIG RDATA.

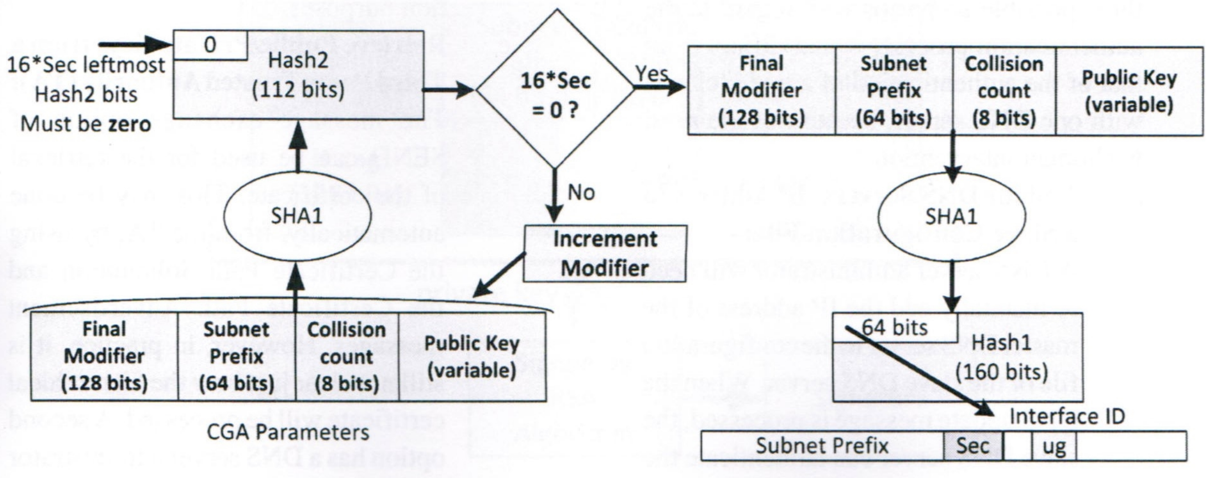
2. Call the CGA generation function in order to generate an IP address.

   A newly joined node generates its IP address by calling the CGA generation function. The steps used for the generation of the CGA algorithm are, as depicted in Figure 2, as follows:

   The node

   a. Generates a random modifier
   b. Concatenates a modifier consisting of a zero value for the prefix (64 bits), a zero value for the collision count (1 bit) and a RSA public key
   c. Executes a Secure Hash Algorithm (SHA1) on the output of step 2 and takes the 112 bits of the resulting digest and names it Hash2
   d. Compares the 16×Sec leftmost bits of Hash2 to zero. If the condition is not met, increments the modifier and repeats steps 2 thru 4. If the condition is met, then execute the next step.
   e. Concatenates the modifier using the prefix, collision count, and public key. It then executes another SHA1 on that output and calls it Hash1. It takes the

Figure 2. CGA algorithm



first 64 bits from the Hash1 output and calls this the Interface ID (IID). It then sets the first 3 left-most bits to the Sec value. It also sets bits 7 and 8 to one (called u and g).

f. Concatenates the subnet prefix with the IID and executes Duplicate Address Detection (DAD) in order to avoid address collision on the network. It sends all CGA parameters (modifier, subnet prefix, collision count, public key), along with the messages, so that other nodes will be able to verify its address ownership.

The steps necessary for the generation of the modified TSIG (CGA-TSIG) are as follows:

1. **Obtain Required Parameters from Cache:** The CGA-TSIG algorithm obtains the old IP address, modifier, subnet prefix, public key from the cache. It concatenates the old IP address with the CGA parameters, i.e., modifier, subnet prefix, public key and collision count (the order of the CGA parameters are shown in step 2 in the prior paragraph describing the IP address generation) and adds them to the initial part of the CGA-TSIG data.

   In the case of multiple DNS servers (authentication of two DNS servers) there are three possible scenarios with regard to the authentication process, which differs from that of the authentication of a node (client) with one DNS server, because of the need for human intervention.

   a. **Add the DNS Servers' IP Address to a Slave Configuration File:**
   A DNS server administrator will need to manually add the IP address of the master DNS server to the configuration file of the slave DNS server. When the DNS update message is processed, the slave DNS server can authenticate the master DNS server based on the source IP address, and prove the ownership of this address by the use of CGA. The disadvantage to the use of this approach occurs when the IP address of one of these DNS servers is changed. Then it becomes necessary to update the IP address in the DNS configuration file. Another possible solution for automating this process entails saving the public key, of the sender of the DNS Update message, on the other DNS server, after the source IP has been successfully verified for the first time. In this case, when the sender generates a new IP address by executing the CGA algorithm using the same public key, the other DNS server can still verify it and add its new IP address to the DNS configuration file automatically.

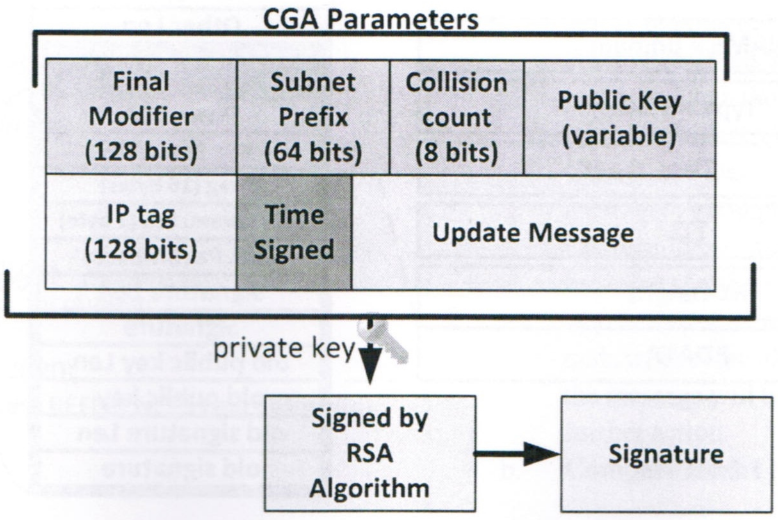
   b. **Manually Exchange the Public/Private Keys:**
   A DNS server administrator will need to manually save the public/private keys, of a master DNS server, in the slave DNS server. This approach does not have the disadvantage of the first approach because any time any DNS server wants to change its IP address it will use the public/private keys which will be readily available for authentication purposes.

   c. **Retrieve Public/Private Keys from a Third Party Trusted Authority (TA):**
   The message exchange option of SEND can be used for the retrieval of the certificate. This may be done automatically, from the TA, by using the Certificate Path Solicitation and the Certificate Path Advertisement messages. However, in practice, it is still not clear just how the hierarchical certificate will be processed. A second option has a DNS server administrator

retrieving the certificate from a TA manually. Then, like in scenario 2, saving the certificate to the DNS server for use in the generation of its address, or in the DNS Update process. In this case, whenever any of these servers wants to generate a new IP address, the DNS update process can still be accomplished automatically, without the need for human intervention.

2. **Generate Signature:**
Before a node starts exchanging a shared secret, it first sends the DNS Transaction Key (TKEY) (Eastlake, 2000) query to the server in order to ask for the establishment of a shared secret session. In order to reduce the number of message exchanges needed between a DNS server and a host, a modified CGA signature is added to the first message sent by a host (a client). This will contain the required information needed for the DNS Update Request.
For the generation of this signature, all CGA parameters (modifier, public key, collision count and subnet prefix), that are concatenated with the IP tag, the DNS update message, and the Time Signed field, are then signed by using a RSA algorithm, or other algorithms that depend on what has been chosen for the CGA generation, and the private key, which was generated in the initial step of IP address generation and was cached for use in this step. This signature can be added as an extended option to the TSIG RDATA field. Figure 3 shows the format of the data in this signature. Time Signed is the same timestamp as that used in RDATA. This value is the UTC date and time value obtained from the signature generator. This approach will prevent replay attacks by changing the content of the signature each time a node wants to send a DNS Update Request. The Update Message contains the entire DNS update message, with the exclusion of the TSIG RR. A DNS update message consists of a header, a zone, a prerequisite, an update and additional data. The header contains the control information, the zone identifies the zones to which this update should be applied (Mockapetris, 1987b), the prerequisite prescribes the RRs that must be in the DNS database, the update contains the

Figure 3. Modified TSIG signature content



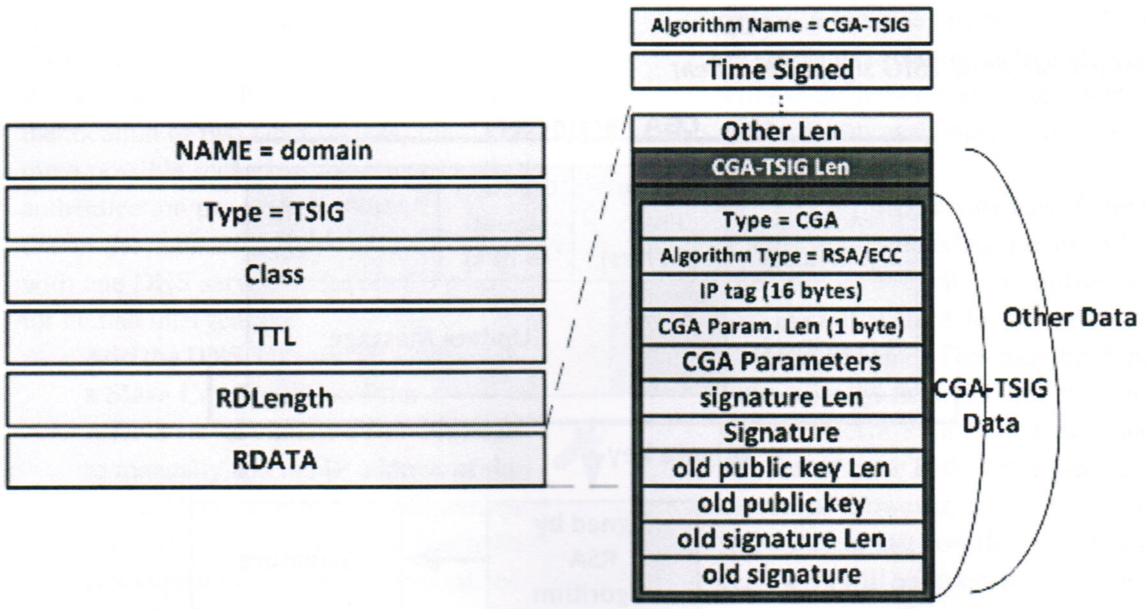
RR that needs to be modified or added, and the additional data is the data that is not a part of the DNS update, but is necessary in order to process this update.

3. **Generate Old Signature:**
If the nodes generated new key pairs, then they need to add the old public key and message, signed by the old private key, to the CGA-TSIG Data. A node will retrieve the timestamp from Time Signed, will use the old private key to sign it, and then will add the content of this signature to the old signature field of the CGA-TSIG DATA. This step will be skipped when the node does not generate new key pairs. In this case, the length of the old signature field is set to zero.

As explained earlier, the TSIG RR contains fields such as Name, Class, etc. The TSIG RDATA field is extended to accommodate the addition of a CGA-TSIG Len and CGA-TSIG Data, as shown in Figure 4. The algorithm name is the same algorithm as that used for generating an IP address in IPv6 networks, i.e., the CGA algorithm which is referred to as CGA-TSIG. The Other Len field contains the overall length of the Other Data field, which will contain the CGA-TSIG Len, the CGA-TSIG Data and other options. CGA-TSIG Data contains the IP tag, the tag used to identify the node's old IP address, the Type used as the name of the algorithm used in SEND, i.e., CGA, the algorithm type used to generate key pairs and sign the message which, by default, would be the RSA, the signature (the format of this signature was explained in step 2 for the CGA-TSIG generation), the old public key, the old signature, and the length of each of them. The length of the CGA parameters will be variable and is dependent on the size of public key.

A client's public key can be associated with several IP addresses on a server. A DNS server keeps a client's public key and IP addresses in a data field formatted as shown in figure 4. This allows the client to update his own RRs using multiple IP addresses, while at the same time, allowing him to change IP addresses. If a client

Figure 4. Modified TSIG RR format



wants to add RRs to the server by using a new IP address, then the IP tag field will be set to binary zeroes and the server will add the new IP address being passed to it to the CGATSIGIPs table in the database. If the client wants to replace an existing IP address in the CGATSIGIPs table (see Figure 5) on the server with a new one, then the IP tag field will be populated with the IP address which is to be replaced. The server will then look for the IP address referenced by the IP tag in the CGATSIGIPs Table (or file) and replace that IP address with the new one.

When a host sends a DNS Update message to a DNS server for the first time, the DNS server must save the public key for this client in CGATSIGkeys.

All DNS update requests/responses sent to the DNS server, or vice versa, should contain the modified TSIG RR in order to give other communicating nodes the ability to validate the sender. These update requests/responses will contain all the required information needed to process the DNS Update Request. Whenever a client, or a DNS server, generates a DNS update request, and uses either TCP or UDP as the transport layer to send this Update Request message to one DNS server, the DNS server should verify this message and, according to the verification result, discard it without further action or process the message. When the process is successful, the DNS server will send a DNS response message back to the sender informing the sender that the update process was completed successfully.

Figure 5. CGA-TSIG tables on MySQL backend database

```
create table cgatsigkeys (
id         INT auto_increment,
pubkey VARCHAR(300),
primary key(id)
);

create table cgatsigips (
id         INT auto_increment,
idkey        INT,
IP         VARCHAR(20),
FOREIGN KEY (idkey) REFERENCES cgatsigkeys(id)
primary key(id)
);
```

The query response sent back to the client from a resolver should also contain the modified TSIG RR. But the clients' query requests do not need to contain this option because a resolver responds to anonymous queries sent from any host. This enables a client to authenticate the resolver and to discard the responses that contain spoofed source IP addresses.

Modified TSIG Verification

To prevent attackers from making an unauthorized DNS update modification, authentication of the sender is very important. The verification process is as follows:

1. **Execute the CGA Verification:**
   In order to verify CGA a node needs to process the following steps:
   a. **Check the Subnet Prefix:**
      The 64 leftmost bits of IPv6 addresses constitute the subnet prefix. The receiver obtains the subnet prefix from the source IP address of the sender's message. Then the subnet prefix is obtained from the CGA parameters in the CGA-TSIG Data field of the received message. A comparison is then made between these two subnet prefixes. If the subnet prefixes match, then execute step 2. If there is not a match, then the node will be considered an attacker and the message will be discarded without further action.
   b. **Compare Hash1 to the Interface ID:**

The receiver will obtain all of the CGA parameters from the CGA-TSIG Data field. Then Hash1 will be calculated by executing SHA1 against these CGA parameters in order to obtain the 64 leftmost bits from the result. Hash1 is then compared to the 64 rightmost bits of the sender's IP address known as the Interface ID (IID). It will ignore any difference in the first three leftmost bits of the IID (Sec value) and the u and the g bits (see Figure 1). u and g are bits 7 and 8 of the first leftmost byte of the IID. If there is a match, execute step 3. If they do not match, then the source will be considered a spoofed source IP address and the message will be discarded without further action.

c. **Evaluate Hash2 with CGA Parameters:**
The receiver obtains the CGA parameters. It sets the collision count and the subnet prefix to zero, and then execute SHA1, on the resulting data, in order to obtain a result. The 112 leftmost bits of the result is called Hash2. The 16×sec leftmost bits of Hash2 are compared to zero. If the condition is met, execute the next step in the CGA-TSIG verification process. If the condition is not met, then the CGA parameters will be consider as spoofed CGA parameters and the message will be discarded without further action.

2. **Check the Time Signed:**
The Time Signed value is obtained from the CGA-TSIG Data field and is called t1. The current system time is then obtained and converted to UTC time, and is called t2. If t1 is in the range of t2 and t2 minus x minutes (see formula 1: x minutes may vary according to the transmission lag time), execute step 3. If t1 is not in the range of t2, then the source will be considered a spoofed message, and the message will be discarded without further action. The range of x minutes is used because the update message may experience a delay during the transmission over TCP or UDP. Both times will use UTC time in order to avoid differences based on geographical location.
$t2-x \leq t1 \leq t2$ (1)

3. **Verify the Signature:**
The signature, contained in the CGA-TSIG Data field of the DNS message, should be verified. This can be done by retrieving the public key from the CGA-TSIG Data field and using it to verify the signature. If the verification process is successful, and the node does not want to update another node's RR, then the Update Message will be processed. If the signature verification is successful and the node wants to update another node's RR(s), then the process will execute step 4. If the verification was not successful, the message will be discarded without further action.

4. **Verify the Source IP address:**
If a node wants to update a/many RR(s) on another DNS server, like a master DNS server wanting to update RRs on the slave DNS server, the requester's source IP address must be checked against the one in the DNS configuration file. If it is the same, the Update Message will be processed. If it is not the same, then step 5 will be executed.

5. **Verify the Public Key:**
The DNS server checks whether or not the public key retrieved from the CGA-TSIG Data field is the same as what was saved manually by the administrator or is in storage where the public keys and IP addresses are saved automatically after the first DNS update. If it is the same, then the Update Message will be processed. If it is not the same, then the message will be discarded without further action.

6. **Verify the Old Public Key:**
   If the old public key length is zero, then skip this step and discard the DNS update message without further action. If the old public key length is not zero, then the DNS server will retrieve the old public key from the CGA-TSIG DATA and will check whether or not it is the same as what was saved in the DNS server's storage where the public keys and IP addresses are saved. If it is the same, then step 7 will be executed. If they are not the same, then the message should be discarded without further action.
7. **Verify the Old Signature:**
   The old signature contained in the CGA-TSIG DATA should be verified. This can be done by retrieving the old public key and old signature from the CGA-TSIG DATA and using this old public key to verify the old signature. If the verification is successful, then the update message should be processed and the new public key should be replaced with the old public key in the DNS server. If the verification process fails, then the message should be discarded without further action.

The verification process for the resolver in a client also follows the same steps as above. With the use of the approach proposed in this chapter, the probability of the attacks, described earlier, being successful will be greatly diminished. This is because the DNS Update requestor's source IP address can be verified by the DNS server's use of the CGA algorithm. The CGA signature also prevents the spoofing of DNS update messages.

Modified TSIG Evaluation

We evaluated different algorithms as part of our proposed TSIG modification. For example, the average time to generate a key pair with a RSA key of 1024 bits using over 1000 samples on a computer with a 2.6 GHz CPU processor and 2 GB of RAM was less than 27.8 milliseconds. This value constitutes about 10% of the total CGA generation process time. The main problem with our proposed approach, thus, is the affect that the CGA algorithm computational process time will have on performance, i.e., the computational cost involved in creating the CGA. This will adversely impact both the address generator and the process used by the attacker. For the attacker, the cost of doing a brute force attack against a (16xSec + 59)-bit hash value would have an algorithm processing time value estimated to be $O(2^{16\times Sec + 59})$. But, in spite of the sequential nature of CGA, it is possible to improve the CGA generation performance with the application of parallelization techniques. To evaluate this approach, we did several experiments using two 64 bit VM operating systems. The first VM was a quad core CPU and the second was a single-core CPU. Both machines had 2 GB of RAM. All the measurements are done for a RSA key size equal to 1024 bits and Sec value of 1. The CGA creation process was called 1000 times to ensure sufficient sampling. The parallel approach sped up the CGA computation time by 70.1%, when using 4 cores. These results show that the CGA process does benefit by the use of a multicore processor utilizing parallelization techniques.

Moreover, when a node once generates a CGA, it does not need to re-generate it in order to send the DNS update message. As explained in prior sections, it can cache that value and fetch it from memory whenever it is needed. This means that, once it is generated, the CGA can then be available for different uses until it is time for the generation of another IP address. Another improvement results from the reduction in network traffic. Generally, to establish a secure DNS update, a minimum of four messages are exchanged between a DNS server and a client. With our approach, just two messages are needed. One is the DNS update request, the other is the DNS update response, both of which contain our proposed TSIG RR. This therefore lends itself to speeding up the DNS Update process.

Modified SEND

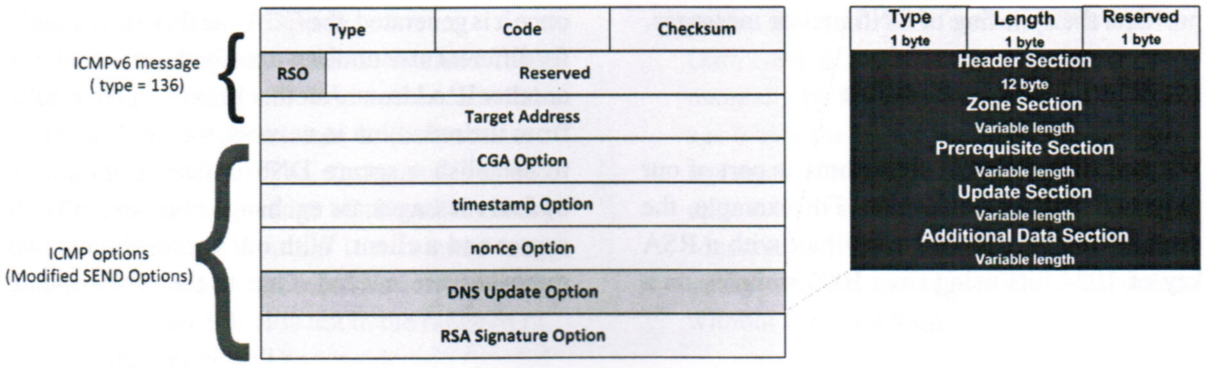
In this approach, the DNS update message is added to the SEND Neighbor Advertisement (NA) message (Rafiee et al., 2013) which means that the DNS server process, using this SEND implementation, will initiate the update process. The DNS service no longer needs to listen to extra ports because, with this implementation of SEND, after a successful verification, another intermediate service is called to add/modify/remove RRs on the DNS database. The NA message is the message that is sent by the node when the node generates its IP address and wants to advertise it. In this case, the node sets the S flag in the NA message to zero. This simplifies the DNS update mechanism for local networks and utilizes a secure authentication mechanism, i.e., the SEND verification process. The NA message format, with modified SEND options, is depicted in Figure 6. As the figure shows, the DNS update is a new message option. This is also included in the RSA signature so that the DNS server will be sure of the integrity of this data. It will also assure the DNS server that this node, with this IP address, actually owns this hostname. The checksum calculation for NA messages is also included in the DNS Update option.

When the DNS server is on another network, for transparency, a Controller Node (CN) is used. The task of the CN is to listen to NA messages sent by nodes and, after successful verification, generating DNS updates on behalf of these nodes. In this case the administrator of that network only needs to configure the CN node by using CGA-TSIG or the other currently available mechanisms instead of configuring all new nodes. In this approach, the CN node needs to maintain the public key, IP address, and domain names, of the nodes, in his database.

CONCLUSION

In this chapter DNS functions are classified into DNS read functions and DNS write functions. Attacks that might be initiated against each of these functions are then described. DNS Update is one of the essential functions used by DNS servers. It affords nodes the ability to update their DNS records dynamically. Unfortunately, it also creates new security issues for DNS servers as to how to authenticate the nodes who's RRs it wants to update. Another problem with this update process lies in maintaining data integrity and confidentiality during DNS queries. There have been three different protocols introduced for securing DNS

Figure 6. Modified NA message with DNS update option



Updates and queries: TSIG, DNSSEC and IPSEC. But in IPv6, if StateLess Address AutoConfiguration (SLAAC) is used, these secure protocols will fail because, in SLAAC, there is no human intervention used to control what nodes join a network. Moreover, in TSIG or DNSSEC, not all processing is done automatically, and a few steps may need to be done offline. To address this problem two solutions were proposed. An extension to the TSIG protocol (CGA-TSIG) and an extension to SEND, which takes advantage of the use of CGA for the DNS authentication process of a node within a DNS server. Both approaches will also decrease the number of messages needed to be exchanged between the DNS server and the DNS client. This will result in enhanced performance for the DNS update process. These approaches might prevent several types of attack such as DNS Update spoofing, etc. These approaches also automate the authentication process between the communicating nodes (a client and a DNS server) since CGA does not need a Public Key Infrastructure (PKI) framework for the verification of the node's address ownership. CGA-TSIG can also be used for securing the DNS read process. The CGA-TSIG data structure can be added, as an extension, to DNS query responses sent by resolvers. This will thus prevent the types of attacks described in earlier sections.

REFERENCES

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005). DNS security introduction and requirements. *RFC*. Retrieved March 2005, from http://www.ietf.org/rfc/rfc4033.txt

Atkins, D., & Austein, R. (2004). Threat analysis of the domain name system (DNS). *RFC*. Retrieved August 2004, from http://www.ietf.org/rfc/rfc3833.txt

Brown, D. L. (n.d.). SEC 1: Elliptic curve cryptography. *Certicom Research*. Retrieved May 21, 2009, from www.secg.org/download/aid-780/sec1-v2.pdf

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *RFC*. Retrieved May 2008, from http://www.ietf.org/rfc/rfc5280.txt

Deering, S., & Hinden, R. (1998). Internet protocol, version 6 (IPv6) specification. *RFC*. Retrieved December 1998, from http://www.ietf.org/rfc/rfc2460.txt

Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., & Carney, M. (2003). Dynamic host configuration protocol for IPv6 (DHCPv6). *RFC*. Retrieved July 2003, from http://www.ietf.org/rfc/rfc3315.txt

Eastlake, D. (2000). Secret key establishment for DNS (TKEY RR). *RFC*. Retrieved September 2000, from http://www.ietf.org/rfc/rfc2930.txt

Kent, S., & Seo, K. (2005). Security architecture for the internet protocol. *RFC*. Retrieved December 2005, from http://www.ietf.org/rfc/rfc4301.txt

Merino, P. J., Martínez, A. G., Organero, M. M., & Kloos, C. D. (2006). Enabling practical IPsec authentication for the internet. *Lecture Notes in Computer Science, 4277*, 392-403. Retrieved from http://rd.springer.com/chapter/10.1007/11915034_63

Microsoft Library. (n.d.). *Recursive and iterative queries*. Retrieved 2013, from http://technet.microsoft.com/en-us/library/cc961401.aspx

Mockapetris, P. (1987a). Domain names - Concepts and facilities. *RFC*. Retrieved November 1987, from http://www.ietf.org/rfc/rfc1034.txt

Mockapetris, P. (1987b). Domain names - Implementation and specification. *RFC*. Retrieved November 1987, from http://www.ietf.org/rfc/rfc1035.txt

Narten, T., Nordmark, E., Simpson, W., & Soliman, H. (2007). Neighbor discovery for IP version 6 (IPv6). *RFC*. Retrieved September 2007, from http://www.ietf.org/rfc/rfc4861.txt

Rafiee, H., Loewis, M. V., & Meinel, C. (2013). DNS update extension to IPv6 secure addressing. In *Proceeding of FINA Conference*. IEEE.

Rooney, T. (2011). Secure DNS (part I). In *IP address management: Principles and practice* (pp. 256–258). Hoboken, NJ: IEEE Press/Wiley.

Stapp, M., & Volz, B. (2006). Resolution of fully qualified domain name (FQDN) conflicts among dynamic host configuration protocol (DHCP) clients. *RFC*. Retrieved October 2006, from http://www.ietf.org/rfc/rfc4703.txt

Thomson, S., Narten, T., & Jinmei, T. (2007). IPv6 stateless address autoconfiguration. *RFC*. Retrieved September 2007, from http://www.ietf.org/rfc/rfc4862.txt

Troan, O., & Droms, R. (2003). IPv6 prefix options for dynamic host configuration protocol (DHCP) version 6. *RFC*. Retrieved December 2003, from http://www.ietf.org/rfc/rfc3633.txt

Vixie, P., Gudmundsson, O., Eastlake, D., III, & Wellington, B. (2000). Secret key transaction authentication for DNS (TSIG). *RFC*. Retrieved May 2000, from http://www.ietf.org/rfc/rfc2845.txt

Wellington, B. (2000). Secure domain name system (DNS) dynamic update. *RFC*. Retrieved November 2000, from http://www.ietf.org/rfc/rfc3007.txt

KEY TERMS AND DEFINITIONS

CGA-TSIG: A combination of CGA with TSIG. This modification of TSIG utilizes CGA as a means of authenticating a node with a DNS server.

DHCPv6: A protocol that can be used to allow a DHCP server to automatically assign an IP address to a host from a defined range of IP addresses configured for that network.

DNS Update: DNS update is a process of adding, removing, or modifying one or several Resource Records (RRs) on DNS servers.

DNSSEC: An extension to DNS which secures the DNS functions and verifies the authenticity and integrity of query results from a signed zone.

IPsec: Provides access control, data authentication, integrity, and confidentiality for the data that is sent between communication nodes across IP networks.

IPv6 Autoconfiguration: A node generates its IP address as soon as it connects to the network by using Stateless or Statefull mechanisms.

NDP: Neighbor Discovery (ND) and StateLess Address Autoconfiguration (SLAAC) mechanism, together, are called NDP. This mechanism allows hosts to discover their neighboring routers and hosts and obtain router information, and to generate and to set their IP address.

TSIG: TSIG is a protocol that provides endpoint authentication and data integrity by using one-way hashing and shared secret keys to establish a trust relationship between two hosts.