Published as : Hosnieh Rafiee and Christoph Meinel. "A Secure, Flexible Framework for DNS Authentication in IPv6 Autoconfiguration". The 12th IEEE International Symposium on Network Computing and Applications (IEEE NCA13), IEEE, August 22 - 24, 2013 Cambridge, MA USA.

A Secure, Flexible Framework for DNS Authentication in IPv6 Autoconfiguration

Hosnieh Rafiee, Christoph Meinel
Hasso-Plattner-Institut, University of Potsdam
P.O. Box 900460, 14440 Potsdam, Germany
{Rafiee, Meinel}@hpi.uni-potsdam.de

Abstract— **The Domain Name System (DNS) is an essential part of the Internet on whose function many other protocols rely. One key DNS function is Dynamic Update, which allows hosts on the network to make updates to DNS records dynamically, without the need for restarting the DNS service. Unfortunately, this dynamic process does expose DNS servers to security issues. To address these issues two protocols were introduced: Transaction SIGnature (TSIG) and Domain Name System Security Extensions (DNSSEC). In Internet Protocol version 4 (IPv4) networks using these protocols eliminated security issues. In Internet Protocol version 6 (IPv6) however, there is an issue with the DNS authentication process when using the StateLess Address AutoConfiguration (SLAAC) mechanism (new to IPv6, nonexistent in IPv4). This authentication issue occurs when a node wants to update its resource records on a DNS server, during the DNS update process, or when a client wants to authenticate a DNS resolver to ensure that the DNS response does not contain a spoofed source address or message. In this paper we propose the use of a new mechanism which makes use of asymmetric cryptography to establish a trust relationship with the DNS server. We also consider the use of the current security parameters used to generate IPv6 addresses in a secure manner, i.e. Secure Neighbor Discovery (SeND), for assuring clients and DNS servers that the one they are communicating with is the real owner of this IP address. Since we are extending the RDATA field within the TSIG protocol to accommodate these new security parameters, we will call this new mechanism the CGA-TSIG algorithm.**

Keywords- DNS update, DNS, CGA, IPv6 autoconfiguration, TSIG, NDP,CGA-TSIG, Resolver authentication

I. Introduction

The Domain Name System (DNS) is a fundamental service of the Internet used by every application using IP addresses or IP devices. It is a distributed, hierarchical database which stores the mappings of IP addresses to hostnames. This hierarchy is referred to as the Domain Name System and is organized like an inverted tree radiating from a single root. The stored data is maintained in Resource Records (RRs). A RR is the basic data element that defines the structure and content of the domain name system. They are recognized by their type identifications. For example, type “AAAA” (IPv6 address record) is a RR that contains the IPv6 address of a host and type “NS” (Name Server record) is another RR used to identify the authoritative name servers for a zone. A zone is a portion of domain space that is authorized and administered by a primary name server and one or more secondary name servers. A name server can be a master or a slave. Master or primary name servers are the ones from which other name servers can transfer zone files. The authoritative name servers are the DNS servers that do not need to look for responses within their own cache, or to ask other name servers for the response, because they own this data and thus look for responses within their own RRs.

The main function of DNS servers or name servers is to respond to queries asked by different hosts. In order to provide hosts with up-to-date responses, the DNS RRs should be updated as soon as any changes are made to a host’s IP address, a host name, or any other data that is maintained within the RRs. Because manually changing the RRs in a zone file and then restarting the DNS service would have an adverse effect on the performance of the DNS servers, including the loss of many DNS queries during the restart process, dynamic update mechanisms have been put in place to facilitate this process. The latest versions of DNS implementations support Dynamic DNS (DDNS). DDNS is a mechanism used to update DNS RRs, on-the-fly, without the need for restarting the DNS service. This mechanism can be used in conjunction with other mechanisms, one such being Dynamic Host Configuration Protocol (DHCP). The main problem with DDNS is its security vulnerability. This is due to the fact that DDNS servers support the basic authentication mechanism which allows a host to update RR records that are based on a source IP address. This will allow a malicious host to spoof the authorized host’s IP address and then to modify the RRs on the DNS server which is the master of a zone. In IPv4 networks some mechanisms have been introduced in order to secure DNS operations in general and DDNS operations in particular: DNS Security extension (DNSSEC) (RFC 4033) and Transaction SIGnagure (TSIG) (RFC 2845). For example, it is possible to use both the active directory and DHCPv4 to give only the authorized host the ability to make updates to DNS records.

Although the mechanisms mentioned earlier work well in IPv4 networks they do not work well in IPv6 networks when Neighbor Discovery Protocol (NDP) is used. The NDP protocol requires no human intervention for the generation and assignment of addresses to the hosts on that network. But this does mean that a new host, who may be malicious, can join a network, can set its IP address, and then update the DNS record if the process is based merely on a source IP address. If any security mechanisms are running during DDNS, such as DNSSEC or TSIG, then problems could be encountered. The first problem relates to the manual configuration requirements needed for both of the previously described secure DDNS protocols. The second problem relates the lifetime of an IP address. In order to maintain privacy, the IP address should only be valid for a short period of time. Thus, to generate a

TSIG shared key, first, the key needs to be generated in the host and then placed on the server so that the authentication process can take place. Since this shared secret is only valid between two nodes (a host and a DNS server), this process will need to be repeated for all nodes on that network. This would again see the need for human intervention which is not really a viable solution. To address this issue, and to facilitate the authentication process when Secure NDP (SeND) is used in IPv6 networks, we propose to use the same parameters as are used in Cryptographically Generated Addresses (CGA) for verifying a host and for proving the address ownership of this host. We also propose the use of a flexible framework for the authentication process.

The remainder of this paper is organized as follows: Section 2 will briefly summarize the IPv6 autoconfiguration mechanisms, Section 3 will explain the DNS functions and the security in use in DNS and DDNS mechanisms, Section 4 will introduce our proposed extension to the TSIG protocol and the flexible framework for DNS authentication, Section 5 will evaluate using our proposed mechanisms, and Section 6 will summarize our conclusions.

II. IPv6 Autoconfiguration

IPv6 represents the next generation of Internet protocol that was proposed in RFC 2460. The main reason for its creation was to address the issue of the exhaustion of IP addresses that exists with IPv4. There are two mechanisms, stateless and stateful autoconfiguration, that can be used to configure a node's IP address. With autoconfiguration, the node is able to configure its IP address as soon as it has joined a new network, without the need for human intervention.

A. Stateful Autoconfiguration

This mechanism uses Dynamic Host Configuration Protocol (DHCPv6) [1] as the means for configuring a node’s IP address. This requires a certain amount of human intervention with respect to the installation and administration of the DHCPv6 servers.

B. Stateless Autoconfiguration

This mechanism specifically refers to the generation of a link local address, the generation of global addresses via StateLess Address AutoConfiguration (SLAAC), and the verification of the uniqueness of the addresses in IPv6 networks [2]. It is used in conjunction with other mechanisms, called Neighbor Discovery (ND) [3], to enable hosts to discover who their neighboring routers and hosts are and to present a means by which the host can obtain router information from them.

Two different mechanisms can be used by hosts to obtain DNS information: stateless DHCPv6 and Router Advertisement (RA)-based DNS configuration [4]. The RA message is sent by a router detailing information about the router prefixes in use in this network. It is needed to enable a node to generate its IP address. In Stateless DHCPv6, hosts configure their IP addresses using stateless IP address configuration and receive other information that is not contained in the RAs from DHCPv6 servers, such as DNS servers. As stated in section A, the configuration of DHCPv6 servers for this mechanism requires extra infrastructure and human intervention.

An alternative mechanism that may be used, when there is either no DHCPv6 infrastructure or clients do not support a DHCPv6 client, is a RA-based DNS configuration. Use of this mechanism enables hosts to obtain DNS information from RA messages.

C. SEcure Neighbor Discovery (SEND)

In order to make NDP more secure an extension to NDP, called Secure Neighbor Discovery (SeND) [5], is used. It provides NDP with security enhancements. SeND adds four new options to NDP messages. These options are Cryptographically Generated Addresses (CGA) [6], timestamp, nonce, and signature.

1) Cryptographically Generated Addresses (CGA)

CGA is an important option in SeND which provides nodes with the necessary proof of address ownership. It does this by providing a cryptographic binding between a host's public key and its IP address without the need for new infrastructure. A SeND-capable node relies on a CGA algorithm where the new dynamic IP address is automatically generated by use of the node’s public key and a one-way hashing algorithm generated from CGA parameters.

When a SeND-capable node wishes to generate a new IP address it uses a RSA algorithm [7] to generate key pairs (public/private keys) on-the-fly. It can also use an external application to generate the necessary key pairs that would then be made accessible to the CGA algorithm. A security level (Sec value) between 0 and 7 is selected. When a Sec level higher than 0 is selected, the strength of the generated IP address is higher thus providing greater protection against brute force attacks. Since we do use the CGA algorithm to prove IP address ownership, the following information is provided describing, briefly, how it works.

The node -

1. Generates a random modifier
2. Concatenates the modifier with a zero valued prefix (64 bits), a zero valued collision count (1 byte) and a RSA public key
3. Executes a Secure Hash Algorithm (SHA1) on the result of step 2 and takes the 112 bits of the digest and calls it Hash2
4. 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, goes to the next step.
5. Concatenates the modifier with the prefix, collision count, and public key. Executes SHA1 on the result and calls it Hash1. Takes 64 bits of Hash1 and sets the first 3 left-most bits to the Sec value. Sets bits u and g (bits 7 and 8) to one. The end result is the Interface ID (IID)
6. Concatenates the subnet prefix with the IID and executes Duplicate Address Detection (DAD) against the result to avoid possible address collisions on the network. It sends all CGA parameters (modifier, subnet prefix, collision

count, public key) along with the messages so that other nodes can verify its address ownership.

III. Domain Name System (DNS) Updates

A. *DNS and its functions*

The DNS consists of a distributed tree structure of databases which contain individual records called Resource Records (RRs) -- such as AAAA, PTR, etc. Each RR describes the characteristics of a zone (domain) and has a binary or wire-format, which is used in queries and responses, and a text format used in zone files. A detailed description of these RRs, as well as their message format, can be found in RFCs 1034 and 1035.

There are two categories of name servers: Authoritative and Recursive.

- Authoritative:
  An authoritative name server is one that gives original and authoritative answers to DNS queries.
- Recursive:
  A recursive name server responds to queries about any domain. It first checks its own records and cache for the answer to the query and then, if it cannot find an answer, it queries other servers before passing the response back to the originator of the query.

1) *The mechanisms used to Update DNS*

DNS update [8] is the process of adding, changing, or removing a RR record in a zone's master file. Dynamic DNS (DDNS) (RFC 2136) is a mechanism used to enable real-time, dynamic updates to entries in the DNS database. The clients or servers can automatically send updates to the authoritative name servers to modify the records they want to change.

B. *DNS Threats*

Originally DNS did not contain robust security features because scalability was an issue. The basic security mechanism of this protocol is to check whether or not the source and destination IP address and the query ID are the same as that which was sent by the resolver. If so, the query answer will be accepted. The use of this process makes it easy for an attacker to spoof this data and then send it to the client's DNS resolver, or to update the node's records on the DNS server, in order to have the traffic forwarded to his desired nodes for the purpose of his gaining network access. This illustrates just how vulnerable this protocol is to several types of attack. These vulnerabilities can be classified into three categories; bugs in DNS implementations or other services, information leakage within the DNS configuration, and other attacks such as cache poisoning, man in the middle, DoS, and DDNS vulnerabilities [9, 10]. An example of how information leakage could occur would be when a zone transfer from a master to a slave server takes place. An attacker can sniff to obtain a copy of the entire DNS zone for a domain. He would thus not need to scan the entire network as he would have already obtained a complete listing of all the hosts in that domain. Moreover, when using DNS for a dynamic update in conjunction with other protocols, such as DHCP, the DNS server may become vulnerable to several other types of attack, such as IP spoofing, record deletion, redirection, and DoS. This could occur because the server is usually the master of a zone and the authentication for such updates is based solely on the source IP address

C. *Existing Security Mechanisms in DNS*

Mechanisms have been introduced in an attempt to make DNS more secure. . The problem with these mechanisms is that they only provided partial solutions to the vulnerability issues mentioned earlier. Therefore some extensions, such as the DNSSEC and TSIG, were implemented in the DNS protocol to try to further reduce its vulnerability.

1) *DNS Security Extension (DNSSEC)*

DNSSEC, introduced by the Internet Engineering Task Force (IETF), is an extension to DNS (RFC 4033) used to validate DNS query operations. It verifies the authenticity and integrity of query results from a signed zone. It uses asymmetrical cryptography meaning that separate keys are used to encrypt and decrypt data to provide security for certain name servers with their respective administrative domains. When DNSSEC is used, all responses include a digital signature. This prevents 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 a problem with using DNSSEC is that the signatures are not created on-the-fly because the DNS, itself, does not have access to the keys which would enable it to sign its own responses. Thus the administrator of that zone needs to sign each domain and sub domain manually, ahead of time, and then store those signatures in the SIG RRs of the DNS server. Also, the zone private key should be stored offline. This is the reason that Dynamic Update cannot be fully supported. It cannot generate the signature, on-the-fly, in order to respond to real-time queries. Also, the use of DNSSEC cannot guarantee the data's confidentiality because it does not encrypt the data but just signs it [9].

2) *Transaction SIGnature (TSIG)*

TSIG (RFC 2845) is a protocol that provides endpoint authentication and data integrity using one-way hashing and shared secret keys to establish a trust relationship between two hosts that may 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 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 in a DDNS update request. Some fields contained in the TSIG RR are: Name, Class, Type, Time To Live Resource Data (TTL RDATA), etc. The RDATA field is used to specify the type of algorithm used in a one-way hashing function along with the other information normally included.

IV. Proposed Flexible Framework

The current solution for automating the manual configuration process makes use of current security protocols, like TSIG or DNSSEC, which allows the use Active Directory (AD) or GSS-TSIG (RFC 3645). This means that the node has already been authenticated and thus will be able to update the DNS records. To address this problem, and to offer a more general solution for minimizing the need for human intervention during the DNS authentication process, we propose a framework where asymmetric cryptography is used to provide nodes with a high level of security. There are two scenarios in play here. The authentication of a node (a client or another DNS server) to a DNS server during a DNS update, and the authentication of a DNS resolver with a client. In both scenarios we assume that the node is aware of the IP address of the DNS server and that of the DNS resolver. This is because they can get this address from the DHCPv6 server (that will not be secure) or from a router advertisement message after authenticating the router via a trusted authority. The IP addresses can be generated using CGA, Simple Secure Addressing Scheme (SASS) [11] or other mechanisms. In this scheme the lifetime of an IP address is not short.

A. *Authentication during the DNS Update*

There are two different scenarios in play here also. One pertains to the authentication of a node with a DNS server in order to update the DNS records, and the other pertains to the authentication of two DNS servers, such as a slave and a master. Our solution focuses primarily on the first scenario but it can also be used to resolve issues stated in the second scenario.

When a client joins a new SeND-enabled network, it first generates its IP address and then must update its DNS RRs. To accomplish this a DNS request message is sent requesting the public key of the DNS server. The DNS server’s response to this client will include the public key contained in DNSKEY RR (RFC 3757) with SEP flag set to zero. This is because it is not the zone key. The client then verifies the public key of the DNS server. If the DNS server's IP address was set using SeND, then there will be a binding between its public key and its IP address. The client can then use the verification steps explained in [7, 11]. If the DNS server did not use SeND to set its IP address, it will need to provide the node with the name of the third party Trusted Authority (TA) where the node can verify the DNS server’s public key. In this case the public key verification process is the same as that used in Secure Socket Layer (SSL) processing, which is explained in RFC 6101. When the SSL protocol is used, the clients are provided with a list of TAs from which they can obtain the public keys of authorized nodes. After a successful verification, using either approach, the client saves the DNS server's public key in its memory, encrypts its hostname and other data using th DNS server’s public key and the same algorithm as is used by the DNS server for key pair generation, and signs this encrypted data, along with the DNS update message, using its own private key. A DNS update message is then sent to the DNS server. For sending this data to the DNS server, we propose an extension be added to the current TSIG RDATA field (this will be explained in more detail in the next sections). We chose to use the TSIG RDATA field because it has an *Other Data* section that can be used to insert the parameters necessary for our verification purposes. In this case we do not need to introduce any new RRs.

In the case of multiple DNS servers (authentication of two DNS servers) there are again two possible scenarios with regard to the authentication process. The authentication process may differ from that of a node (client) with two DNS servers because of the need for human intervention.

1. Manually exchange the public/private keys
   A DNS server administrator needs to manually save the public/private keys of a master DNS server within the slave DNS server. Any time any DNS server wants to change its IP address it needs to use these public/private keys for the authentication.
2. Retrieve the public/private keys from a third party TA by using the SSL key verification process (explained earlier).

1) Generation of a modified TSIG

The public key and other required parameters used to generate a new IP address for a node can be used to create the TSIG RR. These values should thus be cached in the node's memory for later use.

The following steps outline our proposed solution to the Update Request vulnerability issue.

- Step 1. Retrieve the public/private keys and other parameters from cache

The key pairs are generated using a RSA algorithm, or other CGA/SSAS supported algorithms, during IP address generation using SeND. In this step all required CGA/SSAS parameters are obtained from cache. If the node cannot find these values in cache, it will generate key pairs using ECC [12] or RSA algorithms.

- Step 2. Encrypt the data using the DNS server’s public key

A DNS update message consists of a header, a zone, a prerequisite, an update, and additional data. The header contains the control information (RFC 2136). The zone identifies the zones to which this update should be applied (Section 4.1.2 RFC 1035). The prerequisite prescribes the RRs that must be in the DNS database. The update contains the RR that needs to be modified or added. When our framework is used, the update and prerequisite sections should be encrypted using the DNS server’s public key and should not be sent unencrypted. The additional data is that data which is not a part of the DNS update, but is necessary in order to process this update. The node first encrypts the prerequisite data and the update section containing RRs separately using the DNS server’s public key and the same algorithm that the DNS server uses for its keypair generation, which can be RSA, ECC or any other future algorithm. Then it places them in the update and prerequisite sections of the DNS message. To improve this, it is possible to encrypt the update and prerequisite sections using a symmetric algorithm and encrypt the shared secret using the DNS server’s public key. In this case the overhead for using public/private key encryption will be mitigated.

- Step 3. Generate the signature

To generate the signature, all CGA parameters (modifier, collision count and subnet prefix excluding the public key) that were concatenated with the encrypted DNS update message (such as the prerequisite and the update sections) and the *Time*

Signed field, are signed using an ECC algorithm and the private key which was generated in the initial step for the IP address generation. This signature can be added, as an option, to the *Other Data* section of TSIG RDATA field. Figure 1 shows the format of the data in this signature. *Time Signed* is the same timestamp as is used in RDATA. This value is the number of seconds since 1 January 1970, in UTC date and time format, obtained from the signature's generator system. This approach will prevent replay attacks by changing the content of the signature each time a node wants to send a DNS Update Request.

As explained in section III (Part C), the TSIG RR contains fields such as Name, Class, etc. We added our option to *Other Data* section of the TSIG RDATA field to accommodate the addition of a signature and public keys. Figure 2 shows our proposed options to the TSIG RDATA field. The algorithm type refers to CGA-TSIG. *Other Len* defines the overall length of the *Other Data* which contains the algorithm type used to generate key pairs and sign the message which, by default, would be ECC. *Type* indicates the Interface ID generation algorithm that was used by SeND. This field allows for the use of future algorithms in place of CGA. The assigned default value for CGA is 1. Other algorithms would be assigned numerical values sequentially. For example, SSAS could be assigned a value of 2. If the node does not use SeND, and it generates its public/private key by itself, with no association to its IP address, then this value will be set to 0. Other fields in *Other Data* are the *parameters,* public keys and the signature (the format of this signature was explained in the prior section), and fields for the length of each of them. The length of the parameters is variable and depends on the *Type*. If the node generates a new public/private key, it needs to include the old public key, signed by the old private key, and add it to the old signature section of the CGA-TSIG data structure (see figure 2). The old pubkey len field contains the length of the old public key. It is set to zero when the public key of the node does not exist in the DNS server. If the node only wants to change its hostname, and the DNS server already has its public key, then the node will set both the old public key len and new public key len to zero. A client's public key can be associated with several IP addresses on a server. This allows the client to update his own RRs using multiple IP addresses, while at the same time, allowing him to change IP addresses. When a host sends a DNS Update message to a DNS server for the first time, the DNS server must save the public key and hostname of this node in the CGATSIGkeys table. The DNS server assigns the validation time to the public key and stores in CGATSIGkeys. If it does not receive an update request from the node, using this public key, during this allotted time, which depends on the privacy policy of the network, the DNS server sets a status flag, in its database for this public key, to inactive. These inactive records can be removed automatically or by the DNS administrators.

DNS update requests/responses sent to the DNS server, or vice versa, should contain our modified TSIG RR to give the 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 (it should include our proposed TSIG RR), 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.

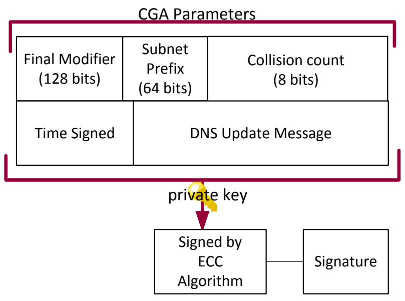

Figure 1. Modified TSIG Signature Content

2) *Modified TSIG Verification*

It is very important to authenticate senders to prevent attackers from making unauthorized DNS update modifications. Since we propose to use the CGA or SSAS algorithm in our approach, the first steps of the verification process are almost identical to those used in the CGA standard RFC [6] or SSAS, i.e., there will be only a few modifications. This is because, in CGA, there is no need to add the signature as it already exists in the SeND [5]. This is why the signature verification is considered as a part of SeND and not as part of the current CGA verification process. In our proposed approach, when a receiver (DNS server or a client) receives a DNS update message, it executes the following verification steps in sequence to authenticate the sender:

- Step 1. Process the CGA/SSAS verifications

The receiver will obtain all the CGA parameters from the TSIG RDATA field. Then Hash1 is calculated by executing SHA1 against these CGA parameters to obtain the 64 leftmost bits of the result. Hash1 is then compared to the 64 rightmost bits of the sender's IP address known as the Interface ID (IID). Any difference in the first three leftmost bits of the IID (Sec value) is ignored along with the u and the g bits. u and g are bits 7 and 8 of the first leftmost byte of the IID. If there is no match, the source is considered a spoofed source IP address and the message is discarded without further action.

When they match, the receiver obtains the CGA parameters. It sets the collision count and the subnet prefix to zero and executes SHA1 on the resulting data. The 112 leftmost bits of the result is called Hash2. The 16×sec leftmost bits of Hash2 are compared to zero. When the condition is met, execute the next step. When the condition is not met, the CGA parameters are considered spoofed CGA parameters and the message is discarded without further action. When SSAS is used, the node follows the SSAS verification process as explained in [11]. If the node generates its public/private key itself, and sets its *Type* in the modified TSIG RR (*Other Data* field) to zero (see figure 2), then skip this step.

- Step 2. Check the *Time Signed*

The *Time Signed* value is obtained from TSIG RDATA and is called t1. The current system time is obtained and converted to UTC time in seconds and this value is called t2. If t1 is in the range of t2 and t2 minus x seconds (see formula 1), then go to step 3, otherwise the source is considered a spoofed message and the message is discarded without further action. The range of x seconds is used because the update message may experience a delay during the transmission over TCP or UDP. This time value is dependent on network policy and transmission delay, so both times will use UTC time to avoid any differences in time based on different geographical locations.

$$t2 - x \leq \mathrm{t1} \leq \mathrm{t2} \qquad (1)$$

- Step 3. Verify the signature

The signature contained in the TSIG RDATA field of the DNS update message needs verification. This can be done by retrieving the public key from the DNS server's database or from the *Other Data* in TSIG RDATA and using this 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 is processed. If the signature verification is successful and the node wants to update another node's RRs, then the process continues with step 4. If the verification is not successful, the message is discarded without further action.

- Step 4. Verify the public key

If a node's public key is the same as that present in the CGATSIGKeys table, then process this update message. If the node's public key and hostname do not exist in the CGATSIGKeys table, and the node does not want to update other nodes' RRs (that exists in DNS database or on other DNS servers on the Internet), add this public key and hostname to the CGATSIGKeys table and process this update message. This is done because it is a new node that is joining this network. 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, then the DNS server checks whether or not the public key retrieved from the TSIG RDATA is the same as what was saved manually by the administrator. If it is the same, then the update message is processed. Otherwise the message is discarded without further action.

B. Authentication during query resolving

The query response that is sent by the resolver back to the client needs to include the modified TSIG RR. However, the client does not need to request the resolver's public key in a separate message because the resolver can include its public key in the same message that is sent to the client in the CGA/SSAS parameters field of the modified TSIG RR. Because a resolver responds to anonymous queries sent from any host, client query requests need not contain this option. Clients can thus authenticate resolvers and can discard responses that contain spoofed source IP addresses. In this case, when the resolver wants to generate the modified TSIG

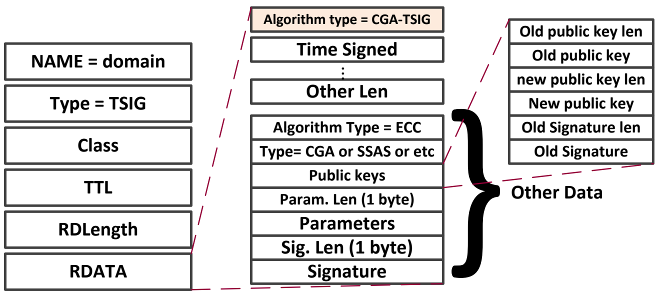

Figure 2. Modified TSIG RR Format

RR, it skips step 2 of the modified TSIG generation and does not include the *DNS update message* in the signature. The verification steps are the same except for step 4.

There are two scenarios in play here. In the first, the resolver generates its public/private key, itself, and does not associate this with its IP address. In this case, when the client first receives a message from the resolver, after successful verification, it stores the resolver's public key in a file. For all further DNS queries, the client will accept the DNS response from the resolver with this public key.

In the second scenario, the resolver generates its IP address using SeND, which makes use of the SSAS or CGA algorithm. This approach is more secure because the client is able to check the address ownership of the resolver the first time it receives a message from him. This way an attacker does not have a chance to spoof the resolver's IP address and then send its own public key to the client.

In both scenarios the client can obtain the public key from the CGA/SSAS parameters.

I. Evaluation

A. Security Analysis

1) Analyzing the RSA algorithm

The security of our proposed approach relies on the degree of difficulty needed to break the chosen asymmetric algorithms. To start generating key pairs, in a RSA algorithm, two prime numbers called p and q are chosen [13]. Then $n = pq$ is calculated. n is used as a module for public/private keys and the size of that module is usually the size of public/private keys. The public/private keys used in the RSA algorithm are chosen to be the same length for security reasons. Then the Euler Totient Function is calculated where the φ function is the number of prime numbers smaller than value n. This value is calculated using formula 1.

$$\varphi(n) = \varphi(q)\varphi(p) = (p-1)(q-1), if\ p\ and\ q\ are\ prime\ (1$$

The Public key consists of two values, an exponential called e and a module called n. These two values are sent to the receiver of the message as a public key. The private key consists of n, as a module, and a secret exponential called d, where $ed = 1 mod (p-1)(q-1)$. There is an attack used against RSA that is related to the size of the module. If the length of the module, known as the key size, is not enough, then an attacker can easily break the RSA by using a brute force attack in the hope of finding the secret module. He does this by using the public key and the encrypted message as an input to the brute force function.

There is another attack used against a factor large integer (factorization). A sieve prime algorithm is an efficient algorithm that is used to find the prime numbers smaller than a certain value x. The attacker can use this function to find all possible prime numbers less than n. Then he tries them, looking for a value of x to use in the brute force equation.

2) *Possible attacks against our approach*

When a node generates its public/private key by itself (not using SSAS/CGA), then that key is not bound to his IP address. During the authentication of the resolver for the first time, with a probability of 0.5, the attacker will be able to send a message with the spoofed source IP address and sign it with his own public key. As the node accepts the first response to its query from the legitimate or illegitimate resolver, this gives the attacker a chance to poison the client’s cache. To prevent this attack we propose to use a monitoring system which will sniff all messages sent from the resolver’s source IP address. If the monitoring system finds two messages at the same time, having two different public keys, then log the event and notify the network administrator. This attack will not be possible if the resolver generates its IP address using CGA or SSAS. During the DNS update, this attack cannot occur because the important values will be encrypted using the DNS server’s public key. Thus, the attacker will not know the content of the data sent from the node to the DNS server thereby preventing him from proceeding with his attacks.

There is another attack that can be perpetrated against DNS servers. Attackers can send thousands of DNS update message using different hostnames which will deplete the DNS server’s resources by making it perform countless verification processes along with adding the public key to the CGATSIGKeys. Denial of Service attacks are the type of attacks that cannot be easily prevented unless a monitoring system is used. Another possibility for preventing these attacks would be to not allow the DNS server to accept any requests from nodes with unknown subnet prefixes. This configuration can be added to the firewall or also to the DNS server itself.

3) *Implementation and Testing*

The data that we evaluated after our implementation consisted of generation and verification times and the packet from our proposed TSIG modification. For example, the average time needed to generate a key pair using a RSA key of 1024-bits for 10 samples on a computer with a 2.6 GHz CPU processor and 2 GB RAM was less than 200 milliseconds. This value constitutes about 10% of the total process time for the CGA generation. The main problem with using the CGA algorithm is the effect that the computational process will have on performance, i.e., the computational cost involved in creating the CGA. SSAS is the solution to this problem. Its compute times are much shorter than those for CGA and it also provides the node with proof of IP address ownership. Another solution is to improve the CGA algorithm. In spite of the sequential nature of CGA, it is possible to improve the performance time for CGA generation by applying parallelization techniques [14]. This speeds up the process, thus reducing the total time spent, by a node, in generating its own IP address and then sending the DNS Update Message.

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 was explained in previous sections, it can cache that value and fetch it from memory whenever it is needed. This means that once it is generated, CGA will be available for different uses until it is time for the generation of another IP address.

The use of the ECC algorithm, as the default algorithm, is preferable in solution to sign the message, but it is not practicle for message encryption. This is because currently the ECC algorithm is only available for digitally signing the message and for symmetric encryption using a shared secret. This shared secret should be exchanged manually or it should be encrypted using an assymetric encryption like RSA. Elliptic Curve Integrated Encryption Scheme (ECIES) [15] is an assymetric encryption based on Diffie-Hellman Integrated Encryption Scheme (DHIES) proposed by Victor Shoup in 2001, but, It is not widely used for encryption purposes. Table 1 shows that an ECC with a 192 bit key size can be used for digitally sign the data. This is equivalent to a 7680 bit RSA key size. In this case, the packet size would be decreased by a factor 11 times smaller than when using RSA. RSA key generation and signature generation and verification times, using a higher key size than what is shown in this table, are really slower than those for ECC. But using the current key sizes, i.e., 1280 key size, RSA performs better than ECC. We also evaluated the encryption and decryption times of messages of 50 bytes and 100 bytes. Our results showed that decryption consumed more time than encryption, but with this size of message, the total time was a small value, less than a 15000 microseconds.

A. *Threat Analysis*

Allowing more flexibility in the authentication process, i.e., letting the host generate the public/private key itself, alleviates the host's dependency on other network services, such as SeND. Even though this high flexibility will work well for the authentication of a node during a DNS update, it might allow an attacker to spoof the IP address of the resolver the first time the node asks for a DNS query from a resolver during the DNS resolver to client authentication process. There are both advantages and disadvantages concerning the generation of key pairs using the DNS service or of the use of cached data. For example, CGA-TSIG uses the cached values available in the node after the generation of the IP address in a secure manner. But if the node does not use SeND, the key pairs must be generated the first time the node wants to send a DNS update message. However, it is possible to use the same key pairs for a certain period of time, which is dependent on network policy. On the other hand, CGA-TSIG does not depend on SeND.

There are several types of attacks that our proposed approach may prevent. Here we evaluate some of those attacks.

1) *IP Spoofing*

During the DNS Update process it is important that both communicating parties know the one they are communicating with is the real owner of that IP address and that messages have

Published as : Hosnieh Rafiee and Christoph Meinel. "A Secure, Flexible Framework for DNS Authentication in IPv6 Autoconfiguration". The 12th IEEE International Symposium on Network Computing and Applications (IEEE NCA13), IEEE, August 22 - 24, 2013 Cambridge, MA USA.

Table 1. Comparison of ECC and RSA

Algorithm type	Key size	Average Key Generation (microseconds)	Average Signature Generation (microseconds)	Average Signature Verification (microseconds)	Message Encryption <= 100 bytes data (microseconds)	Message Decryption <= 100 bytes data (microseconds)
RSA	1280 bits	350651	45527	34347	1000	12701
ECC	192 bits	95544	66877	104332	-	-

not been sent from a spoofed IP address. In the CGA-TSIG approach, this can be fulfilled by the use of the CGA/SSAS algorithm which utilizes the node for the IP address ownership verification. In this case the node generates a public/private key itself, and then uses the signature and the public key to prevent this type of attack. However, it is recommended that the DNS servers (a slave, a master or a DHCP server that wants to update DNS records on behalf of other nodes) do a manual exchange of the public keys or use Third Parties in order to retrieve the proper certificates from them.

2) *DNS Dynamic Update Spoofing*

Because the signature contains both CGA/SSAS parameters and the DNS update message, proof is offered of the data integrity of the message and the validity of the update message.

3) *Resolver Configuration Attack*

Regardless of whether TSIG or DNSSEC is used, when our proposed extension is implemented onto a DNS server and into a client application, the DNS server or the client will not need further configuration. This reduces the possibility for the introduction of human errors in the DNS configuration file. Since this type of attack is predicated on human error, it will be minimized with the use of our proposed extension. For clients, DNS clients only need to support CGA-TSIG Data fields so the update is completely automatic. In servers, only for first time configuration is human intervention required. In this case a Third Party Trusted Authority is used to obtain the public key without a need for human intervention.

4) *Replay attack*

Using *Time Signed* in the signature modifies the contents of the signature each time the node generates it and sends it to the DNS server. This value is the current time of the node, in UTC. As explained in prior sections, this prevents the attacker from copying the content from the original message.

II. CONCLUSION

DNS Update gives nodes the ability to update their DNS records dynamically. Unfortunately, security issues exist for DNS servers trying to authenticate nodes whose Resource Records (RRs) need updating. Two different protocols were introduced to secure DNS Updates: TSIG and DNSSEC. In IPv6, when stateless autoconfiguration is used, these secure protocols fail because in stateless autoconfiguration there is no control over the nodes that join the network. The secure DNS Update will thus fail the authentication process. Moreover, when using TSIG or DNSSEC, not all processing is done automatically. We thus propose a flexible solution where we offer the use of asymmetric cryptography for the DNS Update authentication process of a node within a DNS server (extension CGA-TSIG). We also offer the same solution for the authentication of a DNS resolver with a client. We showed how these processes improve and automate the authentication process. Our evaluation showed that our approaches could prevent several types of attacks -- DNS Update spoofing, etc. We also explained how to authenticate without needing to use a TA, except when we want to eliminate the manual step necessary for key exchange when authenticating two DNS servers,

REFERENCES

[1] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "SEcure Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, Internet Engineering Task Force, July 2003, http://tools.ietf.org/html/rfc3315

[2] S. Thomson, T. Narten, T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, Internet Engineering Task Force, September 2007, http://tools.ietf.org/html/rfc4862

[3] T. Narten, W. Nordmark, W. Simpson, H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, Internet Engineering Task Force, September 2007, http://tools.ietf.org/html/rfc4861

[4] J. Jeong, S. Park, L. Beloeil, S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, Internet Engineering Task Force, November 2010, http://tools.ietf.org/html/rfc6106

[5] J. Arkko, J. Kempf, B. Zill, P. Nikander, "Secure Neighbor Discovery (SeND)", RFC 3971, Internet Engineering Task Force, March 2005, http://tools.ietf.org/html/rfc3971

[6] T. Aura, "Cryptographically Generated Addresses (CGA)", RFC 3972, Internet Engineering Task Force, March 2005, http://tools.ietf.org/html/rfc3972

[7] J. Jonsson, B. Kaliski, "Secure Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Secifications Version 2.1", RFC 3447, Internet Engineering Task Force, February 2003, tools.ietf.org/html/rfc3447

[8] B. Wellington, "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, Internet Engineering Task Force, November 2000, http://tools.ietf.org/html/rfc3007

[9] R. Austein, M. Larson, D. Massey, S. Rose, " DNS Security Introduction and Requirements", RFC 4033, Internet Engineering Task Force, March 2005, http://tools.ietf.org/html/rfc4033

[10] S. Ariyapperuma, C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC", The Second International Conference on Availability, Reliability and Security (ARES'07), pp,:335-342, April 2007

[11] H. Rafiee, C. Meinel, "A Simple Secure Addressing Scheme for IPv6 AutoConfiguration", IEEE, The 11th Annual Conference on Privacy, Security and Trust, July 2013

[12] D. L. R. Brown, "SEC 1: Elliptic Curve Cryptography", Certicom Research, http://www.secg.org/download/aid-80/sec1-v2.pdf , 2009

[13] J. Hoffstein, J. Pipher, J. H. Silverman, "An Introduction to Mathematical Cryptagraphy", Springer, ISBN: 978-0-387-77993-5, 2000

[14] H. Rafiee, A. AlSa'deh, C. Meinel, "Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating Systems", IEEE 26th of the International Conference on Information Networking (ICOIN 2012), February 2012

[15] Victor Shoup, "Elliptic Curve Integrated Encryption Scheme", http://www.cryptopp.com/wiki/Elliptic_Curve_Integrated_Encryption_Scheme , 2001