Dynamic DNS Update security, based on Cryptographically Generated Addresses and ID-Based Cryptography, in an IPv6 autoconfiguration context

Jean-Michel Combes
Orange Labs
France Telecom - Orange
Issy-Les-Moulineaux, France
Email: jeanmichel.combes@orange.com

Ghada Arfaoui*, Maryline Laurent†
CNRS Samovar UMR 5157
Institut TELECOM, TELECOM SudParis
Evry, France
*Email: ghada.arfaoui@it-sudparis.eu
†Email: maryline.laurent@it-sudparis.eu

***Abstract*—This paper proposes a new security method for protecting signalling for *Domain Name System* (DNS) architecture. That is, it makes secure DNS update messages for binding a *Fully Qualified Domain Name* (FQDN) of an IPv6 node and the IPv6 address of the node owning this FQDN. This method is based on the use of *Cryptographically Generated Addresses* (CGA) and *ID-Based Cryptography* (IBC). Combination of these two techniques allows DNS server to check the ownership of the IPv6 address and the FQDN, sent by the DNS client. In addition, this paper describes how this method has been implemented.**

***Index Terms*—IPv6; DNS update; Security; Cryptographically Generated Addresses; ID-Based Cryptography;**

I. Introduction

The **DNS** architecture, widely deployed in Internet, is used to store the binding between a **FQDN** of an IP node (e.g., `hostname.example.com`) and information related to this FQDN (e.g., the IP address of the node owning the FQDN). When the node's IP address changes, this information must be consequently updated in the DNS server: this may be done with the *dynamic DNS update* (**DNS UPDATE**) mechanism. To thwart masquerading threats, security mechanisms have been standardized, like *Secret Key Transaction Authentication for DNS* (**TSIG**) or *DNS Request and Transaction Signatures* (**SIG(0)**), but they still have limitations.

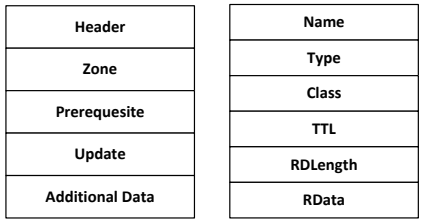
In this paper, we present a new method, securing the DNS UPDATE signalling, based on **CGA** and **IBC**. Our works focus on securing the DNS update of the binding between the name of an IP node and its IPv6 address: our solution brings the ownership proof of these two information elements.

Our article is organized as follows. First, the DNS architecture and related security threats are presented in section II with current countermeasures and their limitations. In section III, our solution is described including the two techniques CGA and IBC, and how the combination of them makes secure the DNS UPDATE signalling. Section IV gives implementation details of the solution, and section V an analysis of its advantages and limitations.

II. Domain Name System

A. Dynamic DNS update

The *dynamic DNS update* (**DNS UPDATE**) mechanism [1] allows an IP node to modify or delete information in the DNS server. This mechanism reuses the DNS message format, over UDP, and the structure of the information stored in the DNS server, known as *Resource Record* (**RR**), illustrated respectively in Figure 1 (a) and Figure 1 (b).



(a) (b)

Fig. 1. (a) DNS message format & (b) RR format

A DNS message is identified as a DNS update message thanks to Header field. Zone field indicates the domain zone including the FQDN to update. Prerequisite field contains the conditions for doing the update. Update field indicates actions to be performed: either deleting and/or adding the information, using RR(s) to encode these actions and the associated information. Finally, Additional Data field contains complementary information related to Update field.

Regarding RR structure, Name field includes the FQDN. Type field indicates the type of FQDN's information (e.g., `AAAA` for an IPv6 address). TTL field includes lifetime of the information. RData field contains the information (e.g., the IPv6 address) and RDLength indicates RData field's lenght.

B. Security considerations and current protection mechanisms

Allowing an IP node to update (or delete) data in the DNS server introduces security threats. There are mechanisms to

protect data against these threats: TSIG and SIG(0). The use of these mechanisms is strongly relying on DNS deployment architecture, and may suffer from limitations, especially in an environment where IPv6 autoconfiguration is deployed.

1) Threats: When an IP node updates data in the DNS server, two information elements are critical: the node's FQDN and IP address. Indeed, a malicious node could usurp either the FQDN or the IP address or both of them, when updating data in the DNS.

In the first case (i.e., spoofed FQDN), all the communications to the FQDN are redirected to the malicious node's IP address because the DNS UPDATE messagee includes a spoofed FQDN (e.g., www.mybanc.com) associated to its IP address. This attack looks like the well-known DNS cache poisoning attack but is more critical because all the DNS databases (i.e., DNS master/slave servers and DNS caches) are impacted. Moreover, the *DNS Security* (**DNSSEC**) mechanism [3] cannot protect against this attack: indeed, DNSSEC only guarantees the integrity and the authentication of information sent by a DNS server but, here, the information stored in the DNS server are already corrupted.

In the second case (i.e., spoofed IP address), all the communications to the FQDN are redirected to a wrong location because the DNS UPDATE messagee includes a spoofed IP address associated to its FQDN. This may be an IP address which is not assigned to an IP node and so, the victim cannot initiate a communication with the IP node associated to the FQDN. Or, this may be an IP address assigned to another node and so, this last one can be potentially the target of a *Denial of Service* (**DoS**) based on a flooding attack. Again, DNSSEC cannot protect against this threat: as explained previously, the information stored in the DNS server are already corrupted.

The DNS server may be updated either manually or dynamically. In the last case, the security mechanisms described in the two following sections may be used.

2) TSIG: Secret Key Transaction Authentication for DNS (**TSIG**) [4] is a security mechanism providing authentication for DNS UPDATE messages based on symmetric cryptography: DNS client, sending DNS UPDATE messages, the DNS server share a common secret key. TSIG uses *Hash-based Message Authentication Code* (**HMAC**) algorithms [5] for the authentication computation. TSIG security information are stored in a RR structure, TSIG RR.

3) SIG(0): DNS Request and Transaction Signatures (**SIG(0)**) [6] is a security mechanism providing authentication for DNS UPDATE messages too but it is based on asymmetric cryptography: DNS server knows DNS client's public key, stored in a KEY RR inside the domain zone configuration file. DNS client, when sending DNS UPDATE messages, protects them with its own secret key. SIG(0) can use different algorithms (e.g., Diffie-Hellman, RSA/SHA-1, DSA). As TSIG, SIG(0) security information are stored in a RR structure, SIG RR, where (cf. Figure 1 (b)), Type field value is 24, TTL field value is 0, RData mainly contains the signer's FQDN, algorithm used for the signature computation and its result. Once computation is done over DNS UPDATE message, SIG RR is included in Additionnal Data field of this message (cf. Figure 1 (a)).

4) Security limitations: Currently, in most of the enterprises and fixed cable/telecom operators IPv4 networks, the common scenario, as illustrated in Figure 2, is that a DHCP server [10] (acting as a DNS client too), (1) assigning an IP address to the IP node, (2) will also update the information (i.e., binding between the FQDN and the assigned IP address) in the DNS server. Regarding the security between the DHCP server and the DNS server, either none is set-up, especially when the servers are in the same administrative domain and so the security can be based only on layer 2 trust, or TSIG is used because TSIG is simpler to deploy (i.e., only a pre-shared key configured in the DHCP and DNS servers is needed) and less expensive than SIG(0) from a computation point of view (i.e., symmetric versus asymmetric cryptography). As such, currently, as far as the authors know, SIG(0) is not really deployed and TSIG is generally used.

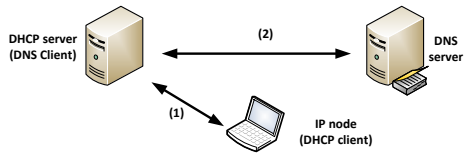


Fig. 2. DHCP use in IPv4 scenario

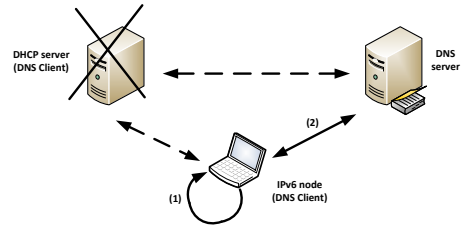


Fig. 3. IPv6 autoconfiguration scenario

In an IPv6 environment, a DHCPv6 server [11] can become useless when, as illustrated in Figure 3, (1) IPv6 nodes are able to autoconfigure their own IPv6 addresses [12]. As such, (2) these nodes will have to update by themselves the information into the DNS server. Consequently, the DNS server needs to check that information (i.e., FQDN and IPv6 address), sent by a IPv6 node, are correct and really owned by this node. The use of TSIG is not well adapted because, for each IPv6 node, the DNS server will have to configure a pre-shared key, and this is not really scalable when there are plenty of IPv6 nodes (e.g., in a fixed cable/telecom operator network). Regarding SIG(0), it doesn't solve the address ownership issue and needs to configure a public key in the DNS server for each IPv6 node.

III. Our solution

The goal of our solution is to solve the two following issues: the ownership of the IPv6 address and the ownership of

the FQDN. For scalability reasons, we based our solution on asymmetric cryptography and so we decided to reuse SIG(0), bringing two advantages: we use its authentication/integrity feature and it needs only minor modifications to integrate our solution.

A. IPv6 Address ownership

For this issue, the ownership property of the IPv6 Cryptographically Generated Addresses is used. A new RR is specified to transport the parameters related to these specific addresses and a minor modification is brought to the SIG RR.

1) Cryptographically Generated Addresses: An IPv6 address is the concatenation of two 64-bits parts where the first part is the network prefix and the second one is the *Interface Identifier* (**IID**). The IID is generally derived from the MAC address [13] but other IID generation methods are existing [14]. *Cryptographically Generated Addresses* (**CGA**) [15] are IPv6 addresses where the IID is the hash computation over the concatenation of a public key and specific parameters.

To generate a CGA, an IPv6 node needs first a RSA public/private key pair [16]. After performing the standardized algorithm [17], the IPv6 node should get an IID, associated with the key pair. This IID results from the first 64 bits of the SHA-1 hash function [18] applied over the data structure called *CGA Parameters*, illustrated in Figure 4. In this data structure, Modifier field and Collision Count field contain values used by the standardized algorithm for the CGA generation. Subnet Prefix field includes the IPv6 prefix that will be concatenated to the IID to finally build the CGA. Finally, the Public Key field contains the public key from the RSA public/private key pair.

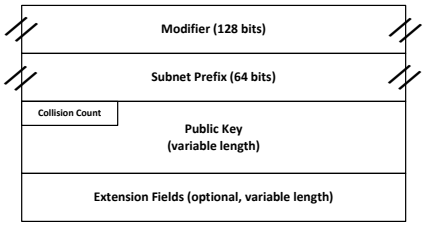


Fig. 4. CGA Parameters

To verify the ownership of a CGA, an IPv6 node needs first getting the associated CGA Parameters and data signed with the private key related to this CGA. The IPv6 node checks that it can regenerate the same CGA, following the standardized algorithm [17], and if so, then it checks the validity of the signature to confirm the node using the CGA is the real owner of the public key related to this address.

2) CGA use: The ownership feature of CGA, with the DNS UPDATE message, is used to certify that the message sender is the real owner of the address (i.e., the CGA) that will be updated in the DNS server. As illustrated in Figure 5, (1) the IPv6 node generates its CGA (and so a public/private key pair, K_{CGApub}/$K_{CGApriv}$) and (2) sends a DNS UPDATE message, including the *CGA Parameters*, which is signed with $K_{CGApriv}$. Finally, (3) the DNS server checks the ownership of the CGA, as explained before, and updates the information concerning the IPv6 node if the verification is correct.

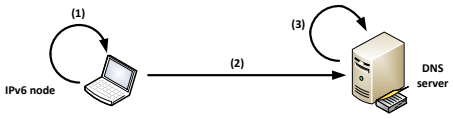


Fig. 5. CGA use scenario

3) CGAparams RR: A new RR is specified in a DNS UPDATE message to transport the CGA Parameters which are needed by the DNS server for the CGA ownership verification. The new RR is named *CGAparams* and its structure is strongly closed to the CGA Parameters structure, as illustrated in Figure 6.

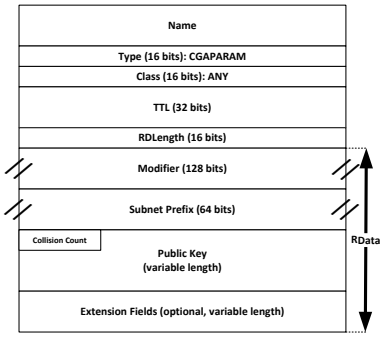


Fig. 6. CGAparams RR

4) SIG(0) modification: The SIG(0) is modified so a new value is defined in Algorithm Field of the SIG RR to indicate the signature is based on a CGA private key. When this new value is set, the SIG(0) process has to proceed with a CGA verification (i.e., CGA regeneration followed by the signature verification).

B. FQDN ownership

For this issue, we decided to use Identity-Based Cryptography. We had only to do a minor modification to the SIG RR. We also defined a mechanism, which was not the main goal of our works but needed for the proof of concept, to provide the necessary security material between the different entities.

1) Identity-Based Cryptography: **IBC** [19] deals with asymmetric cryptographic schemes (i.e., schemes involving a public/private key pair) which public keys are derived in a special way. While in non-IBC asymmetric schemes the public key is usually derived from a "randomly generated" private key, in IBC schemes the public key is deterministically derived from an identity chosen by the user (e.g., the email address `example@foo.com`). The associated private key is generated by an entity named the *Private Key Generator* (**PKG**). PKG is configured with a Master public/private key

pair to generate the private key associated to an identity. PKG needs to provide, to the sender or/and the receiver, the public PKG parameters used for cryptographic computations: Master public key, two hash functions and, with the *Elliptic Curve Cryptography* (**ECC**) [20] [21], a random point of the elliptic curve. This scheme can be applied either for encryption purposes, the most common use case, or signature purposes.

2) IBC use: To secure the DNS UPDATE message, we decided to use IBC, instead of the KEY RR [6], to certify message's sender is the real owner of the FQDN that will be updated in the DNS server. As illustrated in Figure 7, (1) the IPv6 node gets the public parameters from the PKG: PKG's Master public key K_{PKGpub}, two hash functions *Hash1()* and *Hash2()*, and a random point of the elliptic curve *ECC_Pt*. (2) The IPv6 node generates its identity (i.e., its public key $K_{FQDNpub}$), which is in fact the FQDN, and gets the associated private key $K_{FQDNpriv}$ from the PKG. (3) Using this private key, the IPv6 node signs the DNS UPDATE message and sends it to the DNS server. (4) DNS server gets the public parameters from the PKG and checks the signature's validity. If verification is correct, it updates information relative to the IPv6 node.

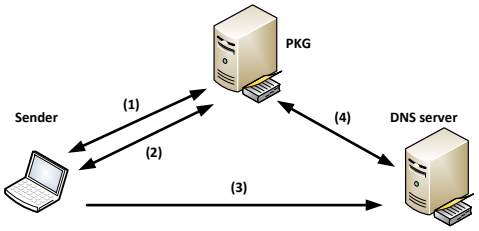


Fig. 7. IBC use scenario

3) SIG(0) modification: SIG(0) modification consists in defining a new value in the SIG RR, for Algorithm Field indicating that Signature Field includes a signature based on IBC scheme. If this new value is set, SIG(0) process has to get the PKG's public parameters, when the DNS server doesn't already know them, and check the signature with the identity (i.e., the FQDN).

4) Security material transport: For the proof of concept, we had to specify how PKG communicates with the different entities. We defined, over UDP [22], 3 types of message: "PKG's public parameters request" message, "Private key associated to an identity request" message and "PKG reply" message. We assume the message exchanges are secure and, especially, PKG is trusted when sending the PKG's public parameters.

C. Global solution

Finally, we integrated together the 2 previously described components, CGA and IBC parts, to reach our goal. As illustrated in Figure 8, (1) the IPv6 node generates its CGA (related to a public/private key pair, K_{CGApub} and $K_{CGApriv}$). (2) The IPv6 node gets PKG's public parameters: PKG's Master public key K_{PKGpub}, two hash functions *Hash1()* and *Hash2()*, and a random point of the elliptic curve *ECC_Pt*. (3) The IPv6 node generates its identity (i.e., its public key $K_{FQDNpub}$, which is in fact the FQDN, and gets the associated private key $K_{FQDNpriv}$ from the PKG. (4) The IPv6 node generates the DNS UPDATE message, includes the CGA RR, computes the two signatures of the DNS UPDATE message, respectively with $K_{FQDNpriv}$ and $K_{CGApriv}$, and includes them (i.e., one SIG RR per signature) in the DNS UPDATE message before

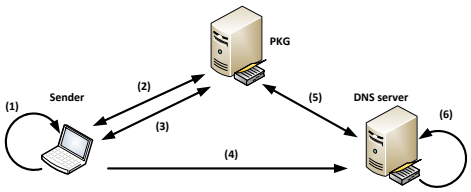


Fig. 8. Global solution

(5) When receiving the message, the DNS server gets the public parameters from the PKG, checks the validity of the signature based on the FQDN. (6) The DNS server checks the ownership of the CGA and, if the two verifications are correct, updates the information concerning IPv6 node.

IV. Proof of concept

To demonstrate the validity of our solution, a proof of concept was implemented and deployed on a testbed.

A. Testbed

The testbed, as illustrated in Figure 9, is composed of 3 entities: an IPv6 node which acts as DNS client, a router which acts as PKG and a DNS server. The operating system on these entities is Debian 2.6.32. All the testbed is only running over IPv6. DNS server is based on BIND 9.7.3[1] from ISC. PKG implementation is based on *Pairing-Based Cryptography* (**PBC**) library[2] from Stanford university.

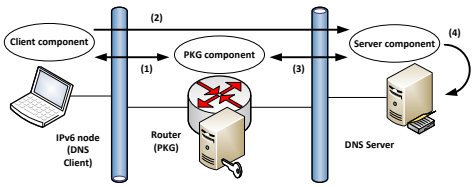


Fig. 9. PoC Testbed and Implementation architecture

B. Implementation

We decided to implement our solution into 3 components: a "Client" component, a "Server" component and a "PKG" component. As illustrated in Figure 9, "Server" component, located on the DNS server, (4) has only to send locally a `nsupdate` command to BIND daemon when (3) the security of the DNS UPDATE message is correct. This avoids doing

[1]http://www.isc.org/software/bind
[2]http://crypto.stanford.edu/pbc/

any modification to BIND. (1) "Client" component has to generate the CGA, secure the DNS UPDATE message based on our solution and (2) send it. "PKG" component provides the needed security material to "Client" and "Server" components.

For the CGA part of our solution, we reused NDProtector[3], a Python based SEND/CGA implementation and especially CGA generation/validation module which is based on scapy6[4], a fork of the well-known packet manipulation program scapy[5].

For the IBC part of our solution, we reused PBC library and for signature model, we used the Hess method [23].

All functions of our solution, illustrated in Figure 10, are described in the following sections.

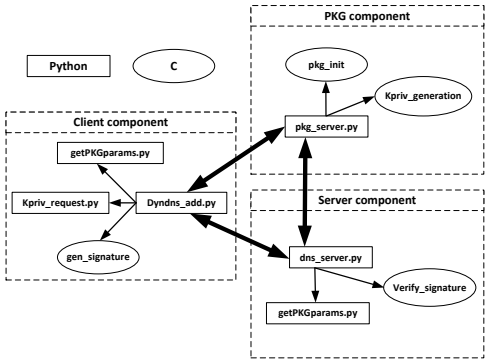


Fig. 10. Functions of our solution

1) "PKG" component: It has to provide needed security material to "Client" and "Server" components: the private key associated to a FQDN and the PKG's public parameters. This component is composed of 3 functions.

The C function `pkg_init` initializes the PKG: generation of the PKG's public parameters (i.e., K_{PKGpub}, *Hash1()*, *Hash2()* and *ECC_Pt*) and the PKG's Master private key. The C function `Kpriv_generation` allows the generation of the private key (i.e., $K_{FQDNpriv}$) associated to an identity (i.e., the FQDN in our context). Finally, the Python function `pkg_server.py` is the main part of the PKG component: at first, this one launches the function `pkg_init` and then, listens for any request (i.e., either PKG's public parameters or private key associated to an identity using the function `Kpriv_generation`) and replies to them.

2) "Client" component: It has to generate the CGA, get the public paramaters and the private key associated to the FQDN from the PKG, generate and sign the DNS UPDATE message, and finally send it. This component is composed of 4 functions.

The Python function `getPKGparams.py` allows the "Client" component to request and get the PKG's public parameters. The Python function `Kpriv_request.py` is used by the "Client" component to request and get the private key (i.e., $K_{FQDNpriv}$) associated to an identity (i.e., the FQDN in our context). The C function `gen_signature` allows the signature of the DNS UPDATE message based on the private key (i.e., $K_{FQDNpriv}$) associated to the FQDN. Finally, the Python function `Dyndns_add.py` is the main part of the "Client" component: it generates the CGA, requests and gets the PKG's public parameters using the function `getPKGparams.py`, requests and gets the private key associated to a FQDN using the function `Kpriv_request.py`, generates the DNS UPDATE message including the CGA RR, signs this message, respectively with $K_{FQDNpriv}$ and $K_{CGApriv}$, using the function `gen_signature`, and sends it to the DNS server.

3) "Server" component: When receiving a DNS UPDATE message, "Server" component has to get the public paramaters from the PKG, check the security validity of the DNS UPDATE message, and finally, if the verification is correct, notify the DNS server daemon about the corresponding update. This component is composed of 3 functions.

The Python function `getPKGparams.py` allows the "Server" component to request and get the PKG's public parameters. The C function `Verify_signature` allows the signature's verification generated with the private key (i.e., $K_{FQDNpriv}$) associated to the FQDN. The Python function `dns_server.py` is the main part of the "Server" component: when a DNS UPDATE message is received, this function requests and gets the PKG's public parameters using the function `getPKGparams.py`, checks the FQDN ownership using the function `Verify_signature` and the CGA ownership (i.e., CGA regeneration and signature verification with K_{CGApub}). Finally, if verification is correct, this function generates and sends locally a DNS UPDATE message (i.e., binding between the FQDN and the CGA) to the DNS server daemon.

V. Advantages and limitations

Advantages and limitations of our solution are given below, and, for each limitation, possible solutions, when any, are given.

A. Advantages

First of all, upon receiving a DNS update, server is able to check the FQDN's ownership like TSIG and SIG(0). Moreover it allows to check the IPv6 address's ownership,included in the DNS UPDATE message and that will be associated to the FQDN in the DNS server. To the best of our knowledge, there is no alternative solution today for such a security feature.

Unlike SIG(0), our solution doesn't require anymore the storage of the public key for each IPv6 node, included in a KEY RR, in the DNS server. Thus, when DNSSEC is deployed too, as the domain zone configuration file is smaller, the time to sign the zone should be reduced too.

Our solution is more scalable than the use of TSIG which requires to provide a pre-shared key for each IPv6 node willing to update information in the DNS server.

[3]http://amnesiak.org/NDprotector/
[4]http://natisbad.org/scapy/
[5]https://secdev.org/projects/scapy/

B. Limitations and potential solutions

The main limitation of our solution is the revocation. Indeed, a CGA provides a strong relationship between an address and its owner but nothing prevents a malicious host to use it in the future when the CGA has been compromised by a collision. Unlike X.509 certificates using a *Public Key Infrastructure* (*PKI*), where either a *Certificate Revocation List* (CRL) [24] or *Online Certificate Status Protocol* (OCSP) [25] can be deployed, a CGA is infrastructureless and so cannot be revoked. Concerning the IBC, revoking the public key would mean revoking the identity and so the FQDN in our context. A solution could be to concatenate a specific data (e.g., a date) to the identity, as already proposed in previous studies [26].

Finally, the last limitation is that the PKG becomes a point of failure: if this one is compromised, all the public/private key pairs used by the IPv6 nodes are compromised. To solve this, previous studies [27] recommend that the PKG generates only one part of the private key related to an identity and the final user finalizes the private key's generation.

VI. Conclusions and future works

In this paper, we presented a new security method, based on CGA and IBC, allowing a DNS server to check the ownership of the received information (i.e., the IPv6 address and the FQDN), and we described how this mechanism was implemented. Currently, for an environment where the IPv6 autoconfiguration is deployed, as far as we know, no alternative solutions exist with the same security properties as the mechanism presented in this paper.

First, the next step regarding our solution would be to secure the exchanges between the PKG and the other entities, for example, with classical methods like TLS. The analysis of the performances in a real DNS environment should be useful to determine the right cryptographic algorithms (e.g., ECC, RSA) for the different parts of our solution. Finally, our solution could be submitted to the IETF as a standard solution.

Acknowledgment

The authors would like to thank Johan Clier (Telecom SudParis), Matthieu Coudron (Telecom SudParis) and Henri Gilbert (ANSSI) for their help.

References

[1] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)," Internet Engineering Task Force, RFC 2136, Apr. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2136.txt

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," Internet Engineering Task Force, RFC 4033, Mar. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4033.txt

[3] P. Vixie, O. Gudmundsson, D. E. 3rd, and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)," Internet Engineering Task Force, RFC 2845, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2845.txt

[4] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Engineering Task Force, RFC 2104, Feb. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2104.txt

[5] D. Eastlake, "DNS Request and Transaction Signatures (SIG(0)s)," Internet Engineering Task Force, RFC 2931, Sep. 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2931.txt

[6] ——, "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)," Internet Engineering Task Force, RFC 2539, Mar. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2539.txt

[7] ——, "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)," Internet Engineering Task Force, RFC 3110, May 2001. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3110.txt

[8] ——, "DSA KEYs and SIGs in the Domain Name System (DNS)," Internet Engineering Task Force, RFC 2536, Mar. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2536.txt

[9] R. Droms, "Dynamic Host Configuration Protocol," Internet Engineering Task Force, RFC 2131, Mar. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2131.txt

[10] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," Internet Engineering Task Force, RFC 3315, Jul. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3315.txt

[11] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," Internet Engineering Task Force, RFC 4862, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4862.txt

[12] M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks," Internet Engineering Task Force, RFC 2464, Dec. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2464.txt

[13] T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," Internet Engineering Task Force, RFC 4941, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4941.txt

[14] T. Aura, "Cryptographically generated addresses (cga)," in *Information Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, vol. 2851, pp. 29–43.

[15] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, February 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[16] T. Aura, "Cryptographically Generated Addresses (CGA)," Internet Engineering Task Force, RFC 3972, Mar. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3972.txt

[17] N. institute of standards and technology, "FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-2," DEPARTMENT OF COMMERCE, Tech. Rep., August 2002. [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[18] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53. [Online]. Available: http://dl.acm.org/citation.cfm?id=19478.19483

[19] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987. [Online]. Available: http://www.jstor.org/stable/2007884

[20] V. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology CRYPTO 85 Proceedings*, ser. Lecture Notes in Computer Science, H. Williams, Ed. Springer Berlin / Heidelberg, vol. 218, pp. 417–426.

[21] J. Postel, "User Datagram Protocol," Internet Engineering Task Force, RFC 0768, Aug. 1980. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[22] F. Hess, "Efficient identity based signature schemes based on pairings," in *SAC 2002, LNCS 2595*. Springer-Verlag, 2002, pp. 310–324.

[23] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Internet Engineering Task Force, RFC 5280, May 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5280.txt

[24] M. Myers and H. Tschofenig, "Online Certificate Status Protocol (OCSP) Extensions to IKEv2," Internet Engineering Task Force, RFC 4806, Feb. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4806.txt

[25] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing." Springer-Verlag, 2001, pp. 213–229.

[26] S. S. Al-riyami, K. G. Paterson, and R. Holloway, "Certificateless public key cryptography." Springer-Verlag, 2003, pp. 452–473.