International Journal of Innovations & Advancement in Computer Science
IJIACS
ISSN 2347 – 8616
Volume 3, Issue 8
October 2014

DNS Server Cryptography Using Symmetric Key Cryptography

Manish Shrivastava, Vineeta Singh
Institute of Technology, Guru Ghasidas Vishwavidyalaya

Abstract

The Domain Name System (DNS) is a hierarchical distributed database that facilitates the conversion of human readable host names into IP address and vice versa.
The security services are provided in DNS using DNSSEC system which provides security services through cryptography. Symmetric key cryptography can be applied to provide security services in DNSSEC. We have introduced several modifications in the process so as to extend the security services in cryptography.

Keywords: Domain Name System Security (DNSSEC), Authentication Protocols, Digital Signatures, Symmetric Encryption, Certificates

1 Introduction

Before discussing about DNS Security extension it is important to know the whole DNS System.

The Domain Name System (DNS) is a distributed computing system that enables access to internet resources by user-friendly domain names rather than IP address, by translating domain names to IP address and vice versa. This process is called name resolution. Name resolution is the mapping of domain names to IP address.

DNS System should have a data repository to store domain names and their associated IP address. This database must be distributed due to size, scalability and performance considerations. Also copies of database must be present for fault tolerance.

There is a need to manage the domain names repository and provide name resolution service too. This is provided by primary DNS component, name server. There are two main types of name server:

i) Authoritative name server which is authoritative for RRs for a particular zone(or zones) and answer for queries for resources for that zone using RRs in its own zone file. It can be a primary name server (stores the definitive version of all records in that zone) or a secondary name server (uses an automatic update mechanism to maintain an identical copy of primary server's database for a zone).

ii) Caching name server which stores DNS results for a period of time determined in the configuration (ttl) of each domain record. It provides responses through series of queries to authoritative name server in the hierarchy of domains found in name resolution query or from cache of responses of previous queries

To access the services provided by a DNS Server on the behalf of user processes there is another component of DNS called the resolver. There are two primary categories of resolvers:

i) Stub resolver which is basically a library that needs to be installed on every host that wants to access the DNS database. Every time a query needs to be sent functions of this library are called and process of retrieving the desired information is run.

ii) Resolving name server located on DNS server and serves a group of stub resolvers. When a recursive query is received the resolver sends iterative query to one of the root DNS servers serving the root domain.

2 Overview of DNSSEC

The main objective of DNSSEC is to ensure the authentication and integrity for the data received form DNS database via digital signatures scheme based on public key cryptography.

2.1 Need of DNSSEC

DNS system includes following entities:
i) The platform (hardware and OS) on which name servers and resolvers reside.
ii) Name servers and resolver software
iii) DNS Transactions
iv) DNS database (zone files)
v) Configuration files in the name server and resolver

DNS is expected to provide name resolution information for any publicly available resource on the internet. Hence, except for DNS data pertaining to internal resources (eg servers inside firewall) that is provided by internal DNS

name servers through secure channels, the DNS data does not require confidentiality.
Ensuring the authenticity of information and maintaining the integrity of information in transit is critical for efficient functioning of internet.
Hence integrity and source authentication are the primary DNS security goals.

2.2 Existing system

Each node in DNS tree is associated with public key. Each message from DNS Server is signed under the corresponding private key. One or more public keys of DNS root are publicly known. These public keys are used to generate the signature that binds the identity of each top level domain to corresponding public key.

Each parent signs the public keys of all its children in DNS tree. KEYRR is used to associate a domain name with a certain public key.

2.2.1 DNS query/response

A DNS query originates from a resolver and the destination is an authoritative or caching name server. The most common type of query is lookup for an RR based on owner name or RR Type. The security objective for query/response is to verify the integrity of each response received and that valid data ha s originated form right source called data origin authentication.

These services could be provided by establishing trust in source and verifying signature of data sent by the source. In DNSSSEC trust in public keys for signature verification is established by establishing chain of trust down to the current source of response through successive verification of signature of public key of child by its parent.

After authenticating source, response is to be authenticated. The response consists of digital signature of requested RRSet along with requested RRs encapsulated through RRSIG. The DNS client using trusted public key of source then verifies the digital signature to detect if the response is valid or bogus.

To ensure that RRs associated with query are really missing in zone file and have not been removed in transit, DNSSEC provides NEXTRR as a means for authenticating the nonexistence of RR.

2.2.2 Zone Transfer

A zone transfer refers to the way a secondary server refreshes the entire contents of its zone file form primary servers so that secondary server keep its zone file in synch with its primary name server.

A zone transaction starts as query from secondary name server to primary name server requesting all RRs from that zone. Since zone transfer places substantial demands on network resources so errant or malicious zone transfer can overload primary name server and denial of service (Dos) to legitimate user. Also there is possibility of tampering of zone messages.

The Dos can be minimized by mutual identification of servers using transaction signature (TSIG) based on shared secret key. TSIG specifies secret key to be used for mutual authentication as well as signing zone transfer requests and responses.

2.2.3 Dynamic Updates

The dynamic updates facility provides operations for addition and deletion of RRs in the zone file. It involves DNS clients making changes to zone data in an authoritative name server in real time. Clients typically performing dynamic updates are CA Server, DHCP Server or Multicast Address Server.

There may be possibility of unauthorized updates such as adding illegitimate resources, deleting legitimate resources, altering delegation information etc. Also data in dynamic update request could be tampered or update request messages could be captured and resubmitting later (replay attacks).

TSIG and SIG(0) meets all the security objectives by including timestamp field in request.

2.2.4 DNS Notify

Whenever changes occur in zone file of primary DNS server, secondary DNS severs must be notified of the changes. This is accomplished through DNS NOTIFY message, which signals secondary DNS server to initiate zone transfer.

The security risk in that is that secondary server may receive spurious DNS NOTIFY messages form sources other than primary name server that could increase its workload.

Secondary name server must be configured with enterprise's primary name server. Also TSIG must be used for communication.

3 Proposed System

3.1 Threats in existing system

The existing system has many shortcomings which are the reason for threats. An attacker could query for NEXT RR of a domain name to find the next domain name in canonical order and repeat the process to learn all the domain names in zone.

This system requires authentication via trusted certification authority (CAs) or trusted third party (TTPs) which can operate offline. So there might be a possibility

International Journal of Innovations & Advancement in Computer Science
IJIACS
ISSN 2347 – 8616
Volume 3, Issue 8
October 2014

where the CA or TTP could be imposter by someone else so as to generate fake certificate and read encrypted messages or sign arbitrary message under stolen identity.

DNS usually runs over UDP. In existing system the authenticated queries and responses do not fit into 512 byte UDP datagram. If a name server with SIG RR is queried for n different types of RRs it would return n public key signatures (1 for each RR). So queries need to be truncated so as to send in another datagram. Also Signature may be exposed to replay attacks unless inception and expiration date are very close.

However the system proposed can tackle these problems very efficiently. In our method we are using symmetric key cryptography technique for the communication between the resolver and name servers. For each transaction new key is generated by the name server so there is no need for full trust on CAs and TTPs. Also symmetric key algorithms are very fast and made of small keys .So authenticated messages fit into 512 byte UDP. In this method signatures cannot be reused providing protection against replay attacks. There is no need for NEXT RRs.

3.2 Proposed approach

Let us suppose local name server or resolver say U queries for IP address of uv.zyz.com.

U must have an authentic copy of root's public key. This key can be communicated from trusted sources or any other out of band technique.

When U contacts root server for the first time it sends a request containing root's header (PH), DNS query and two random keys K1, K2 encrypted under root's public key R_{0u}.

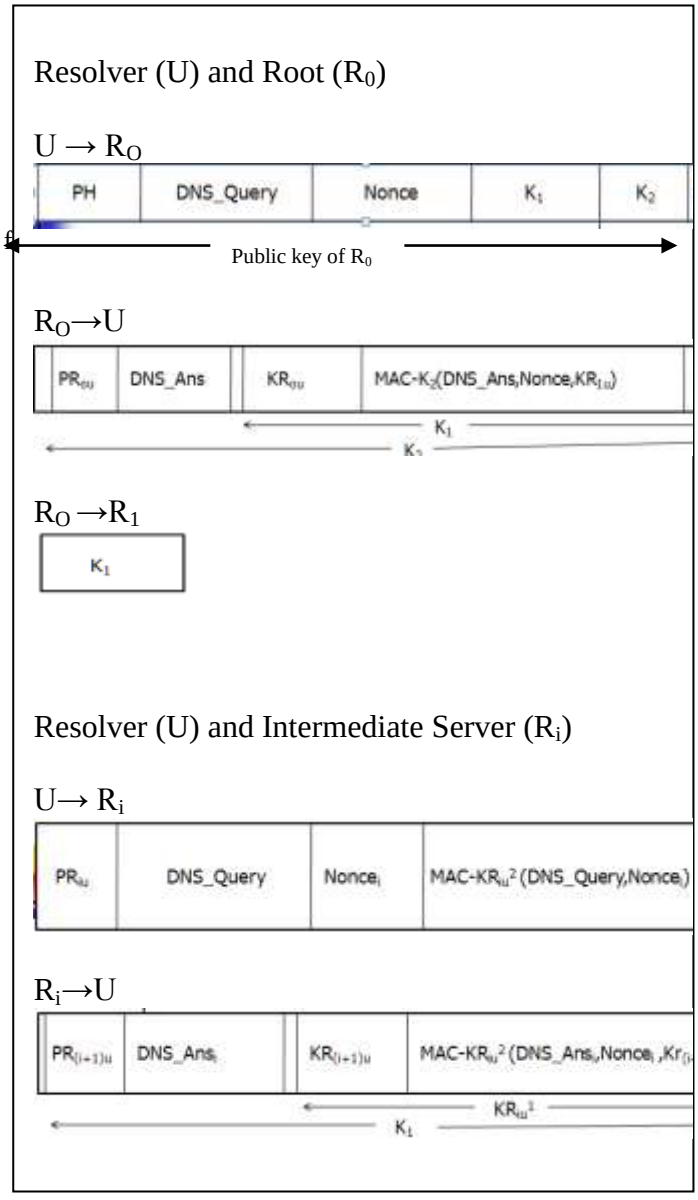
The root decrypts the message with its private key R_{0P} and then matches its address with the u_addr in PH. After verifying the resolver to be authorized root finds the RR for com in its database and sends the certificate for com server R_1, a key pair (KR_{1U}^1, KR_{1U}^2) that is to be shared between resolver U and R1 along with the digest encrypted under K1. The root also shares key K_1 with com server.

The resolver after receiving the message decrypts it with K_1 and extracts the certificate for com server along with the answer, key pair (KR_{1U}^1, KR_{1U}^2) and digest. The resolver verifies the digest to check for integrity and the proceeds to server R_1 to send its certificate along with the request, nonce and digest of request and nonce with key KR_{1U}^1.

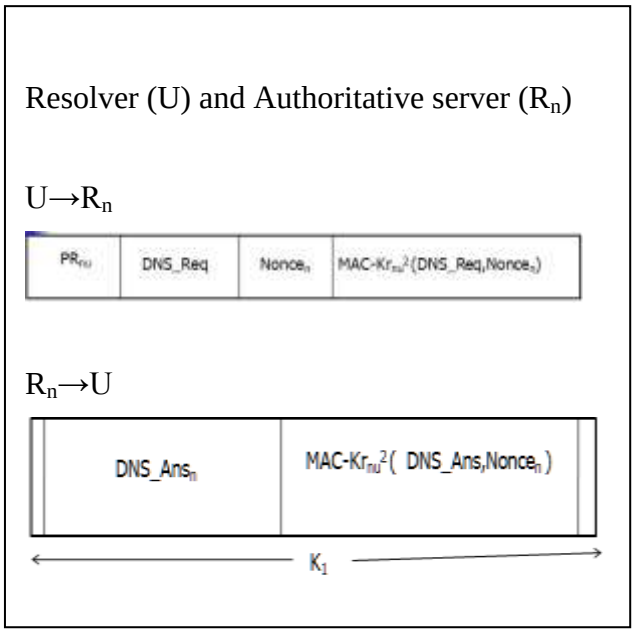
The R_1 server has got the key K_1 from root R_0 so it would decrypt the message and then extracts key pair (KR_{1U}^1, KR_{1U}^2) from certificate. It calculates the MAC of request and nonce with key KR_{1U}^1 AND verifies the digest to check for integrity. It calculates MAC of answer, nonce and key pair for the next server with key KR_{1U}^1, combines it with key pair (KR_{2U}^1,$KR2U^1$) and encrypts it with key KR_{1U}^2. It responds with this message along with certificate of next name server and answer, the whole message encrypted with key K_1.

For the rest of the query the process continues until an authoritative name server's certificate is obtained.

The response from the authoritative name server R_n will contain answer and the MAC of the answer and nonce with key (KR_{nu}^1,KR_{nu}^2) the whole message encrypted with key K_1. The resolver after decrypting the message and verifying the digest will contact the web server corresponding to the host name it queried for.



International Journal of Innovations & Advancement in Computer Science
IJIACS
ISSN 2347 – 8616
Volume 3, Issue 8
October 2014



3.2.1 Algorithm

1. Set Query= enter the query
2. Response= call Resolver_Operation(Query)
3. Print Response

Resolver_Operation(query)

1. Set label= call DNS_Req_last(Query)
2. Modify DNS_Req after removing label from query
3. Generate two random numbers K1 and K2 and $Nonce_0$.
4. Set PH = u_addr+ root_addr + $Nonce_0$
5. Set U_Query= PH + label + K1 + K2. Encrypt U_Query with R_{0U}
6. Response=Resolver_Root_Req(U_query, MAC_{KRou}(U_Query))
7. Response=Decrypt Response with K1
8. Separate Cert, DNS_Ans from Response
9. Decrypt the Response with K1 and separate Key from Response
10. Calculate the Digest of DNS_ans+$Nonce_0$+Key. If this Digest is not equal to Response then flag error
11. Set Resolver_Response=DNS_Ans
12. Repeat steps 12 to 20 until DNS_Req is the last label
13. Set label= call DNS_Req_Last(DNS_req). Modify DNS_Req after removing label.
14. Generate Nonce as any random number.
15. Response= call Resolver_Domain_Req (cert, label,Key, Nonce, MAC_{Key} (label+Nonce))
16. Decrypt Reponse with K1
17. Separate Cert, DNS_Ans from Response
18. Decrypt the Response with Key K_{R1U}^2 and separate new Key pair from Response
19. Calculate the MAC_{KR1U}^1(DNS_ans+$Nonce_0$+Key). If this Digest is not equal to Response then flag error.
20. Resolver_Response=Resolver_Response +DNS_Ans
    (end of while)
21. Return Resolver_Response.

Resolver_Root_request(Query, Digest)

1. Decrypt query with K_{R0P}
2. Calculate digest of Query with K_{Rou} and if it is not equal to Digest then flag error
3. Separate PH from Query. If root_addr≠address of root then flag error
4. Separate Nonce, DNS_Query, K1 and K2 from Query
5. Set DNS_Ans, Cert and Key from root's database corresponding to the query label
6. Set Response= MAC_{K2}(DNS_Ans+ Nonce + Key)
7. Encrypt Response with K1
8. Add Cert, DNS_Ans to the Response and encrypt it with K1
9. Call Domain_Domain(K1)
10. Return Response

Resolver_Domain_Request(Cert,label,Key, Nonce,digest)

1. Set k1 =Call Get_Domain()
2. Calculate MAC_{Key}(label+Nonce) and if it is not equal to digest then flag error
3. Set DNS_Ans, Cert and Key pair from cert's database corresponding to the query label
4. Set Response=MAC_{KR1U}^1(DNS_Ans+ Nonce + Key)
5. Encrypt Response with Key K_{R1U}^2
6. Add Cert, DNS_Ans to the Response and encrypt it with K1
7. Call Domain_Domain(K1)
8. Return Response

Domain_Domain(K1)

1. Set interkey= K1
2. Return

Get_Domain()

1. Return interkey

4 ANALYSIS

In this method we have used short symmetric key certificates. Therefore the size of messages is suited to the 512 byte UDP datagram. Also due to small size the larger

number of responses can be stored in cache as compared to that in public key cryptography.

Also there is no use of NEXT RRs in this method therefore zone files could be more manageable and smaller.

This method provides protection against replay attacks because the signatures cannot be reused since for every course of communication new certificates are generated.

It is well suited to the demands of private domain spaces of corporations which have legitimate need to hide certain part of their name spaces ensuring confidentiality.

Use of symmetric key encryption algorithm reduces the time to a large extent as compared to the public key cryptography. So the speed of communication and name resolution is increased greatly. Managing DNS symmetric certificates is easier.

5 Conclusion

In this paper, we presented a proposal for DNSSEC that implements symmetric key certificates which provides higher level of security and low network traffic as compared to the current one using public key cryptography. This method also helps in reducing storage requirements and ensuring confidentiality.

6 References

[1] Ramaswamy Chandramouli., Scott Rose., NIST Special Publication 800-81-2, "Secure Domain Name System (DNS) Deployment Guide".

[2] IETF DNSSEC WG., "DNS Security (DNSSEC) Charter", IETF, 1994.

[3] Paul Albitz and Cricket Liu, DNS and BIND, 4th Edition O'Reilly, 2001

[4] Steven M. Bellovin, "Using the Domain Name Systemfor System Break-Ins", Proceedings of the Fifth Usenix Unix Security Symposium, pp. 199-208, June 1995

[5] Mockapetris P., RFC 1034, "Domain Names-concepts and Facilities", November 1987. (URL:ftp://ftp.isi.edu/in-notes/rfc1035.txt)

[6] Mockapetris P., RFC 1035, "Domain Names - Implementation and Specification", 1987. (URL:ftp://ftp.isi.edu/in-notes/rfc1035.txt)

[7] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993

[8] Eastlake D., "Domain Name System Security Extensions", RFC 2535, March 1999.

(URL: ftp://ftp.isi.edu/in-notes/rfc2538.txt)

[9] Eastlake D., "Dsa Keys and SIGs in the Domain Name System ", RFC 2536, March 1999.

[10] Eastlake D., Gudmundsson O., RFC 2538, "*Storing Certificates in the Domain Name System*", March 1999.

(URL: ftp://ftp.isi.edu/in-notes/rfc2538.txt)

[11] Eastlake D., "Secret Key Establishment for DNS (TKEYRR)", RFC 2930, September 2000

[12] Eastlake D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000