Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.882 – 891

RESEARCH ARTICLE

A Framework for Using Cryptography for DNS Security

Naveen Kumar[1], Kamal Kumar Ranga[2]

[1]Research Scholar, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana
Email ID: naveengulia333@gmail.com

[2]**AP**, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana
Email ID: kamal.ranga@gmail.com

Abstract: DNS, Domain Name System is a protocol that resolves hostnames to IP Addresses over the Internet. DNS, being an open source, it is less secureand it has no means of determining whether domain name data comes from an authorised domain owner. So, these vulnerabilities lead to a number of attacks, such as, cache poisoning, cache spoofing etc. Hence, there is a need of securing DNS. Digital Signatures are a good way of authyenticating the domain owners. The digital signatures generated with public key algorithms have the advantage that anyone having the public key can verify them. Existing proposals include public key cryptographic algorithms (e.g., RSA, DSA etc.) for securing DNS. With the technology growing faster everyone accesses internet through mobile phones whether it is used to check E-Mails or visiting any secure sites, ECDSA involving ECC (Elliptic Curve Cryptography) concepts having less key sizes as compared to RSA can be implemented to provide security to DNS.

Keyword: DNS, IP, ECC, RSA

1. INTRODUCTION

The Domain Name System is a protocol for locating domain names and mapping them to IP addresses. DNS is a hierarchical, distributed database, which provides mapping between easy to remember hostnames, such as www.mdurohtak.ac.in, and IPv4 or IPv6 network addresses, for example, 117.211.115.134. Figure 1 shows a Domain Name System. Each node in the DNS tree represents a DNS name. Some examples of DNS names are DNS domains, computers, and services. A DNS domain is a branch under the node. For example, mdurohtak.ac.in is a DNS domain.

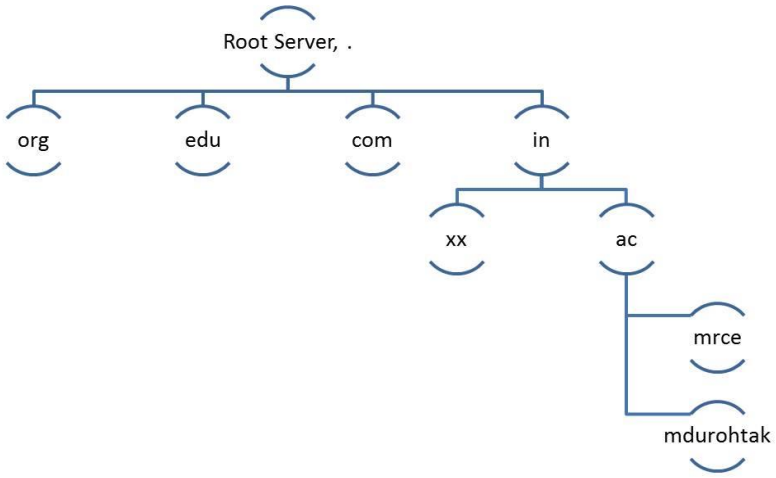

Figure 1.1 Domain Name System

Cryptography is the science of "Secret Writing". The process of changing the plain information called the plaintext into some sort of code called the cipher text is Encryption. The reverse of Encryption is Decryption. Cryptography includes various methods for providing security like Symmetric Key Cryptography, Public Key Cryptography and the Elliptic Curve Cryptography.

Cryptography has also been expanded to provide the following information security requirements:

- **Non-repudiation**: Preventing an entity from denying previous commitments or actions.
- **Integrity**: Ensuring no unauthorized alteration of data.
- **Authentication**: Verifying an entity's identity
- **Confidentiality**: Protecting the data from all but the intended receiver.

2. DNS Security

DNS, as originally designed, has no means of determining whether domain name data comes from the authorized domain owner or has been forged. This security weakness leaves the system vulnerable to a number of attacks, such as DNS cache poisoning, DNS spoofing etc.

Due to weak authentication between DNS servers exchanging updates an attacker may predict a DNS message ID and manage to reply before the legitimate DNS server, thus inserting a malicious record into DNS database. One of the recent vulnerabilities of a DNS server structure is the ability of an attacker to insert false information into caching servers. The exploit forces a compromised DNS server to send a request to an attacker's DNS server, which will supply the wrong host to IP mapping.

In **DNS cache poisoning** attack, an intruder replaces a valid IP address cached in a DNS table with a rogue address. Requests for the valid address are redirected accordingly, and malware (e.g., worm, spyware, browser hijacker etc.) may be downloaded to the user's computer from the rogue location. DNSSEC employs cryptographic keys and digital signatures to ensure that lookup data is correct and that connections are to legitimate servers.

A scenario for **DNS Spoofing** is shown in figure 3 where, when the client sends a request to the DNS server to know the IP Address, it sends its ID with the Query. By identifying that ID, an Attacker can send malicious data to the client or redirect the client to an unwanted site.

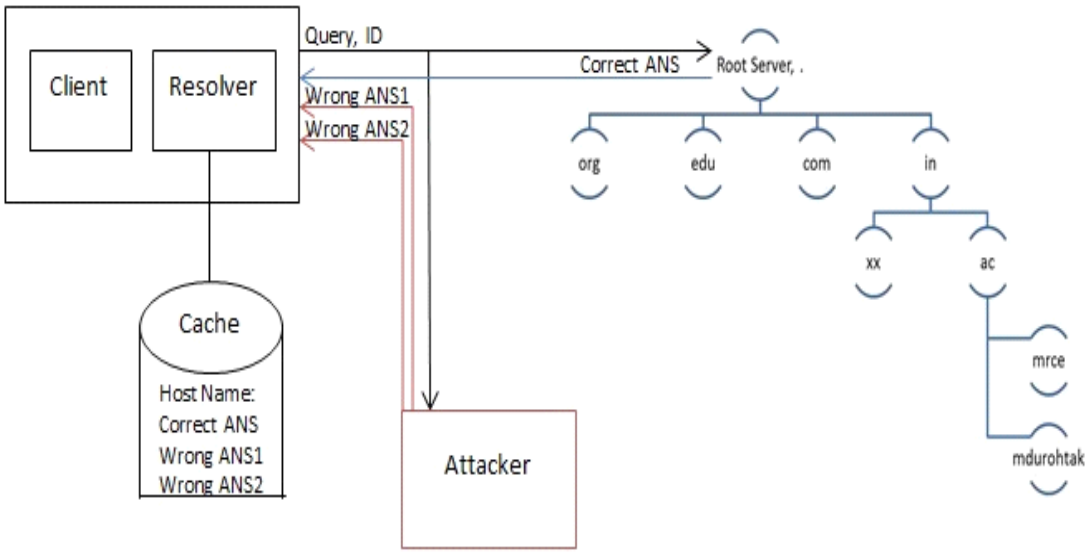

Figure 1.4 DNS Spoofing

Securing DNS means providing- Data Authentication (Origin is authentic) and Data Integrity (Information is not modified). Confidentiality is not required since DNS is public.

DNS Security Extensions (DNSSEC) is a set of Internet Engineering Task Force (IETF) standards created to address vulnerabilities in the Domain Name System (DNS) and protect it from online threats. The purpose of DNSSEC is to increase the security of the Internet as a whole by addressing DNS security weaknesses. Essentially, DNSSEC adds authentication to DNS to make the system more secure.

The **core elements** of DNSSEC were specified in three IETF Requests for Comments published in March 2005:

- **RFC 4033** - DNS Security Introduction and Requirements
- **RFC 4034** - Resource Records for the DNS Security Extensions
- **RFC 4035** - Protocol Modifications for the DNS Security Extensions

Existing proposals for securing DNS are mainly based on public-key cryptography. The public key algorithms used for authentication in DNSSEC are MD5/RSA (Rivest Shamir Adelman Algorithm) and DSA (Digital Signature Algorithm). Digital signatures generated with public key algorithms have the advantage that anyone having the public key can verify them.

The Idea behind it is that every node in Domain Name Space has a Public Key and each message from DNS Servers is signed using Private Key. Since DNS is Public, Authenticated DNS root Public Keys are known to all, which are used to generate Certificates/Signatures to combine the identity information of Top Level Domain. So, in Domain Name Space each parent signs the Public Keys of all its Children in the DNS tree.

Basic working is as follows:

It finds the IP Address of the Host Name "www.mrce.ac.in" in the same way as found above.

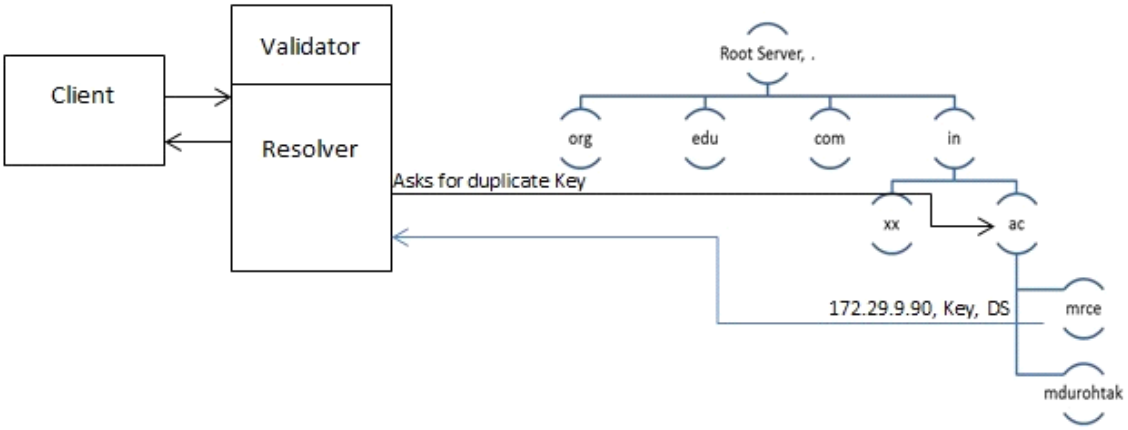

Figure 1.5 Basic Idea of Security

When the response is received, a Key and a DS (Digital Signature) is sent along with it. Then, the resolver asks the top level domain for a duplicate key. Validator, then checks whether the two keys match. If yes, then it is assured that data is not modified. It also checks the DS and its validity. If it is correct, then the data is authentic. Same procedure is repeated for the above domains and finally, the data is verified and send back to the client.

Deploying DNSSEC is a two way process. It must be deployed at both the authoritative side (DNS servers) and the client side (resolvers, browsers, applications).

At the **client side** the security is achieved if the DNSSEC validation is done by the end-user applications rather than by external resolvers at the ISP, e.g., browsers (Chrome, Firefox etc.) are soon to have built-in DNSSEC validators.

Deploying DNSSEC at the **authoritative side** is a bit difficult. Since signing of DNS zones needs tools that assist with zone signing procedure. Taking into account that a zone needs to be resigned every time its content changes and when the signing keys expire, the tool should also be as automated as possible. There are a number of options for signing and maintaining zones:-

- **Open DNSSEC** is a standalone DNS zone signer that works with file based zones.
- **Secure64** is a complete DNS and DNSSEC appliance with an emphasis on security and automation.
- **BIND** version 9.7 and newer for zone signing.

With the technology growing faster everyone accesses internet through mobile phones whether it is used to check E-Mails or visiting any secure sites, ECC (Elliptic Curve Cryptography) can be implemented. ECC provides same level of Security as RSA with benefits of small key sizes, faster computation, and memory and energy savings.

3. Techniques Used

3.1 Symmetric Key Cryptography

This type of cryptography has a common shared key for both encryption and decryption. The key must be shared between the two parties in a secure manner before starting exchanging the data. This can be easily implemented and is faster. The concept is well explained by the following diagram:-

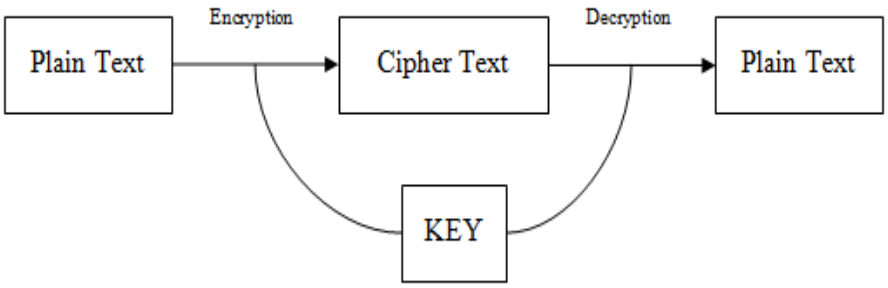

Fig 2.1 Symmetric Key Cryptography

3.2 Public Key Cryptography

This requires a key pair to share the data. The public key (known to all) is used for encryption and the private key (only with the user) is used for decryption. It requires larger resources to produce the key pair, so has slower algorithms. The advantage here is that no key has to be shared before starting the conversation. The concept is well explained by the following diagram:-

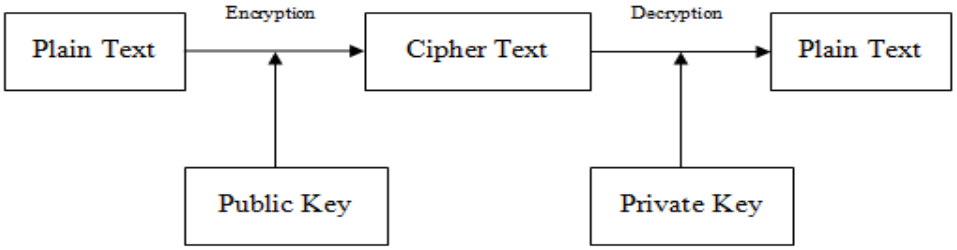

Figure 2.2 Public Key Cryptography

3.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a kind of public key cryptography, based on the concept of elliptic curves. Elliptic curves are basically cubic equations of two variables, with coefficients. ECC uses only those elliptic curves, wherein the variables and coefficients are restricted to elements of a finite field.

3.4 Arithmetic in elliptic Curve over Fp

There are two basic operations as described below:

- **Adding distinct points P and Q:** The negative of the point $P = (x_P, y_P)$ is the point $-P = (x_P , - y_P \bmod p)$. If P and Q are distinct points such that P is not -Q, then P + Q = R where,
  $s = (y_P - y_Q) / (x_P - x_Q) \bmod p$
  $x_R = s^2 - x_P - x_Q \bmod p$
  $y_R = - y_P + s(x_P - x_R) \bmod p$
- **Doubling the point P:** Given y_P is not 0, 2P = R where,
  $s = (3 x_P^2 + a) / (2 y_P) \bmod p$
  $x_R = s2 - 2 x_P \bmod p$
  $y_R = - y_P + s(x_P - x_R) \bmod p$

3.5 Elliptic Curve Discrete Logarithm Problem (ECDLP)

The ECDLP is the basis for the security and is based on the intractability of Scalar Multiplication products. Given points P and Q in the elliptic curve group, then find k such that, P.k = Q. k is a discrete logarithm of Q to the base P.

E.g.: Consider the elliptic curve $y^2 = x^3+5x+7$ defined over F_{19} and let P=(6,4) and Q=(5,0). Then k.P = Q can be computed as:

P = (6, 4)

2P = (14, 16)

3P = (0, 2)

4P= (5, 8)

5P = (5, 0) = Q

So, k = 5

In reality the value of k would be large, making it infeasible to determine k.

Given a point R = k.P, where R and P are known, then there is no way to find out what the value of 'k'. Since, there is no point subtraction or point division, to resolve k = R/P. Also, computing k requires roughly 2n/2 operations. If the key size is 192 bits, then 296 operations are to be done which would take millions of years.

This thing where the multiplicand can't be found even when the original and destination points are known is the whole basis of the security behind the ECDSA algorithm, and the principle is called a trap door function or ECDLP.

3.6 RSA

The RSA scheme is a block cipher in which the plaintext and cipher text are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.

The computational steps for key generation are

1. Generate two different prime p and q
2. Calculate the modulus $n = p \times q$
3. Calculate totient $\varphi(n) = (p - 1) \times (q - 1)$
4. Select for public exponent an integer e such that $1 < e < \varphi(n)$ and $\gcd(\varphi(n), e) = 1$
5. Calculate for the private exponent a value for d such that $d = e - 1 \bmod \varphi(n)$
6. Public Key = [e, n]
7. Private Key = [d, n]

RSA Encryption

1. A message, m which is to be sent.
2. A cipher text, c created by exponentiating: $c = m^e \bmod n$, where e and n are receiver's public key.
3. Sending c to receiver.

RSA Decryption

To decrypt, receiver also exponentiate: $m = c^d \bmod n$, where d is sender's public key. Since only d is known, only the intended receiver can decrypt this message.

RSA Digital Signature

Suppose sender A wants to send a message m to receiver B in such a way that B is assured that the message is both authentic and has not been tampered.

Signature Generation

1. Sender A, creates a digital signature s by exponentiating: $s = m^d \bmod n$, where d and n are A's private key.
2. A sends m and s to Bob.

Signature Verification

To verify the signature, B exponentiates and checks that the message m is recovered: $m = s^e \bmod n$, where e and n are A's public key.

Thus, encryption and authentication take place without any sharing of private keys: each person uses only another's public key or their own private key. Anyone can send an encrypted message or verify a signed message, but only someone in possession of the correct private key can decrypt or sign a message

3.7 ECDSA

The elliptic curve digital signature algorithm is the elliptic curve analogue of DSA and serves the same purposes of key generation, signature generation, and signature verification.

Algorithm

Key Pair Generation

Let A be the signatory for a message M. Entity A performs the following steps to generate a public and private key:

1. Select an elliptic curve E defined over a finite field Fp such that the number of points in E (Fp) is divisible by a large prime n.
2. Select a base point, P, of order n such that $P \in E(Fp)$
3. Select a unique and unpredictable integer, d, in the interval [1, n-1]
4. Compute Q = dP
5. Sender A's private key is d
6. Sender A's public key is the combination (E, P, n, Q)

Signature Generation

Using A's private key, A generates the signature for message M using the following steps:

1. Select a random number k to be used only once, that is, for every new signature generation of a message, a new k is selected, such that $1 < k < n-1$
2. Generate (r, s) component of signature such that
   2.1. k.G = (x, y)
   2.2. r = x modulo n
   2.3. if r = 0 then repeat 2 again
3. Calculate hash of message (M) whose signature is to be generated, i.e., e = h(M)
4. $s = k^{-1}(e + (d*r))$ modulo n

Signature Verification

The receiver B can verify the authenticity of A's signature (r, s) for message M by performing the following:

1. Obtain signatory A's public key (E, P, n, Q)
2. Verify that values r and s are in the interval [1,n-1]
3. Calculate u1 = $e*s^{-1}$ modulo n
4. Calculate u2 = $r*s^{-1}$ modulo n
5. Calculate T = u1.G + u2.Q = (x1, y1), where '.' is point multiplication and '+' is point addition and can be calculated using elliptic curve arithmetic.
6. Calculate v = x1 modulo n
7. The signature for message M is verified only if v = r

4. Conclusion

There are various security measures adopted in DNS using public key cryptography, which includes RSA and DSA. With the technology growing day by day, there is a need of same level of security with smaller key sizes. Now, everyone uses mobile to retrieve data from internet and mobile being small and portable device needs security with less power consumption. This can be done with the help of ECC by implementing ECDSA in DNS.

References

1. William Stallings, Cryptography and Network Security, 4th Edition, Pearson Education, Inc., 2011
2. Neetesh Saxena, Narendra S. Chaudhari, "Secure Encryption with Digital Signature Approach for Short Message Service", IEEE, 2012.
3. Suranjith Ariyapperuma and Chris J. Mitchell, "Security vulnerabilities in DNS and DNSSEC", IEEE Computer Society
4. Aqeel Khalique Kuldip Singh Sandeep Sood, "Implementation of Elliptic Curve Digital Signature Algorithm", International Journal of Computer Applications (0975 – 8887), Volume 2 – No.2, May 2010.
5. Vivek Kapoor, Vivek Sonny Abraham, Ramesh Singh, "Elliptic Curve Cryptography", May 20-26, 2008. ACM Ubiquity, Volume 9, Issue 20.
6. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, Sheueling Chang Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs".
7. Ramaswamy Chandramouli and Scott Rose, "Challenges in securing the domain name system", US National Institute of Standards and Technology, The IEEE Computer Society, 2006.
8. Giuseppe Ateniese , Stefan Mangard , "A New Approach to DNS Security (DNSSEC)".
9. D. Sravana Kumar, CH. Suneetha, A. Chandrasekhar, "Encryption of Data using Elliptic Curve over Finite Fields", International Journal of Distributed and Parallel Systems (IJDPS) Vol.3 (January 2012), No.1.

10. Ghanmy Nabil, Khlif Naziha,, Fourati Lamia, Kamoun Lotfi, "Hardware implementation of Elliptic Curve Digital Signature Algorithm (ECDSA) on Koblitz Curves", 8th IEEE, IET International Symposium on Communication Systems, Networks and Digital Signal Processing.
11. Kadjo Tanon Lambert, Oumtanaga Souleymane, Kone Tiemoman, Abba Brice, and Tety Pierre, "Deployment of DNSSEC: Problems and Outlines".
12. Muhammad Yasir Malik, "Efficient Implementation of Elliptic Curve Cryptography Using Low-power Digital Signal Processor".
13. Sachin Kumar Sinha, Avinash Kant Singh, Amaresh Sharma, "Security System for DNS using Cryptography".
14. Tarun Narayan Shankar, G. Sahoo(April / May 2009), "Cryptography with Elliptic Curves", International Journal Of Computer Science And Applications Vol. 2, No. 1.

