Scientific Journal of Impact Factor (SJIF): 3.134

ISSN (Online): 2348-4470
ISSN (Print) : 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2, Issue 3, March -2015

Generation of Digital Certificate x.509 v3 using Elliptic Curve Cryptography (ECC) & RSA : A Comparative Study & OpenSSL code Revisited

Akshay Yogesh Bahade[1]

[1]*Department of Information Technology, SRM University, akshaybahade@gmail.com*

Abstract *— A public key infrastructure enables users of a basically unsecure public network to securely and privately exchange data through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and when necessary revoke the certificates. The two major digital signature algorithms are Elliptic Curve Digital Signature Algorithm (ECDSA) and RSA algorithm. The two algorithms are used for generating the certificates exchanged between computer systems. The use of X.509v3 certificates to carry out authentication tasks is an approach to improve security. The main advantage of ECC versus RSA is that for the same level of security it requires a much shorter key length. The purpose of this work is to design and implement a free open-source Certification Authority able to issue X.509v3 certificates. The result of this research may assist organizations to increase their security level in wireless devices and networks. This study compares the performance of ECC based signature schemes and RSA schemes. It is observed that ECC based certificate authority schemes gives better speed and security.*

Keywords- Elliptic *Curve Cryptography, Digital Certificate, X.509v3, RSA, Security*.

I. INTRODUCTION

The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet and other applications for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For his reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet. The private key system is sometimes known as symmetric cryptography and the public key system as asymmetric cryptography. In public key cryptography, a public and private key are created simultaneously using the Same Algorithm (RSA) by a Certificate Authority (CA). RSA is the most common method employed in public key cryptography, for instance in X.509 digital certificates. These certificates are oriented to verify the identity of a person or an entity. The private key is given only to the requesting party and the public key is made publicly available in a directory that all parties can access. The private key is never shared with anyone or sent across the network. The private key is used to decrypt the text that has been encrypted with the public key by someone else. In addition to encrypting messages, a user can authenticate himself by using the private key to encrypt a digital certificate. A digital signature is a cryptographically secure method of establishing with a high degree of certainty that the person who electronically signs a message can be verified as the signer with the same confidence as that provided by a witness to a handwritten signature. A digital signature is similar to the Message Authentication Code (MAC) used with symmetric (secret) key systems. The signature is a cryptographic checksum computed as a function of a message and the user's private key. Because public-key systems tend to be slow, digital signatures are often used to sign a condensed version of a message, called a message digest, rather than the message itself. A message digest can be readily generated by a hashing function. Wireless networks have suffered a dramatic increase in recent years. Wireless technology is more and more present in our society and millions of wireless equipment are sold every year. However, one of the major concerns about wireless communications is security. Elliptic Curve Cryptography (ECC) is an innovative cryptographic technique. Its security resides in the same problem as RSA or Diffie-Hellman algorithms, but instead of using integers as symbols of the alphabet to be ciphered, it uses points in a mathematical object called elliptic curve. The real ECC potential is that, with a much smaller key length, it achieves the same security level as other proposals. Therefore, ECC presents some key attributes truly important in scenarios where the following resources are limited: processing power, storage space, bandwidth and power consumption. The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the DSA.

ECDSA was first proposed by Scott Vanstone in the year 1992 in response to National Institute of Standards and Technology (NIST's) request for public comments on their first proposal for DSS. It was accepted in 1998 as an International Standards Organization (ISO) standard (ISO 14888-3), accepted in 1999 as an American National Standards

Institute (ANSI) standard (ANSI X9.62) and accepted in 2000 as an Institute of Electrical and Electronics Engineers (IEEE) standard (IEEE 1363-2000) and a FIPS standard (FIPS 186-2) Digital signature schemes can be used to provide the following basic cryptographic services:

- Data integrity (the assurance that data has not been altered by unauthorized or unknown means)
- Data origin authentication (the assurance that the source of data is as claimed)
- Non-repudiation (the assurance that an entity cannot deny previous actions or commitments)

II. LITERATURE SURVEY

A. Digital Signature

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. The digital equivalent of a handwritten signature or stamped seal, but offering far more inherent security, a digital signature is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures can provide the added assurances of evidence to origin, identity and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer.

How it works?
Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, signing software creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash along with other information, such as the hashing algorithm is the digital signature. The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves time since hashing is much faster than signing. The value of the hash is unique to the hashed data. Any change in the data, even changing or deleting a single character, results in a different value. This attribute enables others to validate the integrity of the data by using the signer's public key to decrypt the hash. If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't changed since it was signed. If the two hashes don't match, the data has either been tampered with in some way (integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer.

A digital signature can be used with any kind of message whether it is encrypted or not simply so the receiver can be sure of the sender's identity and that the message arrived intact. Digital signatures make it difficult for the signer to deny having signed something assuming their private key has not been compromised as the digital signature is unique to both the document and the signer, and it binds them together. A digital certificate, an electronic document that contains the digital signature of the certificate-issuing authority, binds together a public key with an identity and can be used to verify a public key belongs to a particular person or entity. Most modern email programs support the use of digital signatures and digital certificates, making it easy to sign any outgoing emails and validate digitally signed incoming messages. Digital signatures are also used extensively to provide proof of authenticity, data integrity and non-repudiation of communications and transactions conducted over the Internet.

B. Elliptic Curve Cryptography

An elliptic curve is a nonsingular projective algebraic curve over some field k with genus 1 and a specified point O (this will be the “point at infinity”). So long as k does not have characteristic 2 or 3, this will be a smooth plane cubic curve with the point at infinity, and we can describe the curve as points satisfying the equation

$$y2 = x3 + ax + b,$$

With a and b such that the discriminant,

$$\Delta = -16\,(4a3 + 27b2),$$

is nonzero (which will give the desired non-singularity). The group law on an elliptic curve. The operation exploited for key selection in elliptic curve cryptography comes from considering the elliptic curve as an abelian group with points as elements. The group law is point addition; to add two points P and Q, we will draw the line PQ through them (or use the tangent line at P to add it to itself), find the third point of intersection $-R$ of that line, and reflect it over the axis of symmetry of the curve. The resulting point, R, will be the sum of P and Q. For the purposes of this addition, note that the point at infinity O lies on any line through a point and it’s opposite. The formal properties of the addition law are described below.

Theorem - The addition law on elliptic curve C has the following properties (where O = −O is the point at infinity, and if P = (x0, y0), then −P = (x0, −y0)):
(i) For point P ∈ C, P + O = P,
(ii) For points P, Q ∈ C, P + Q = Q + P3
(iii) For point P ∈ C, there is some point −P such that P + (−P) = O
(iv) For P, Q, R ∈ C, (P + Q) + R = P + (Q + R).

In short, the addition law gives us the group properties that we desire. Additionally, we will note that the subset of points in this group whose both coordinates belong to a given field k, along with the point at infinity, will form a subgroup of the curve group C. This will be important, because the curves used in elliptic curve cryptography are defined over a finite field, and we need that set to be closed under point addition. Because our goal now is not to construct elliptic curve cryptography, but rather to understand how it works, we will omit the formal proof, but notice that most of the properties above follow directly from the geometric description of point addition.

C. ECDSA

Elliptic Curve Digital Signature Algorithm (ECDSA) is variant of the Digital Signature Algorithm (DSA) that operates with elliptic curves. Signature schemes are designed to be used when an entity A wants to send a message M to an entity B in an authenticated way, and B wants to verify the authenticity of M.

ECDSA acts as follows – First, an entity A should select what hash function to use (e.g. SHA). Moreover, A should establish the curve domain parameters (a, b, G, n, etc.) at the desire security level. Entity B should get in an authentic manner the selections made by A. Next, A and B should perform a key deployment procedure to be prepared to use ECDSA. A should set up an elliptic curve key pair associated with the elliptic curve domain parameters agreed on the setup procedure to use with ECDSA. Let's call K PA to the A's public key and K pA to the A's private key (randomly selected in [1, n-1]). B should obtain the elliptic curve public key selected by A, i.e. K PA.

To sign a message M, A should proceed according to the following steps:

- A applies the hash function to the message, and derives an integer e from the obtained hash.
- A selects a random integer k in the range [1, n-1], and calculates $K = k \cdot G = (x_A , y_A)$.
- A assesses $r = x_A \pmod n$.
- A calculates $s = k - 1 (e + K_{pA} \cdot r) \pmod n$.
- The signature is the pair (r, s).

To verify the signature, the entity B should proceed as follows:

- B checks if r and s are integers, otherwise the signature is not valid.
- B applies the hash function to the message, and derives an integer e from the obtained hash.
- B calculates

$u_1 = e \cdot s - 1 \pmod n$ and $- 1$

$u_2 = r \cdot s \pmod n$.

- B computes $K = (x_A, y_A) = u_1 \cdot G + u_2 \cdot K_{PA}$.
- The signature is valid if $x_A = r \pmod n$.

In this work, the Certification Authority uses ECDSA to sign an ECC X.509v3 digital certificate containing an ECIES public key, thus verifying the authenticity of the public key and its owner.

D. RSA

RSA is one of the oldest and most widely used public key cryptographic algorithms. The algorithm was invented in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman. The RSA cryptosystem is based on the assumption that factoring is a computationally hard task. This means that given sufficient computational resources and time, an adversary should not be able to "break" RSA (obtain a private key) by factoring. This does not mean that factoring is the only way to "break" RSA. In fact, breaking RSA may be easier than factoring.

RSA Key Generation -

A RSA public and private key pair can be generated using the algorithm below:

- Choose two random prime numbers p and q
- Compute n such that n = p * q

- Compute φ (n) such that φ (n) = (p-1)*(q-1).
- Choose a random integer e such that 1< e < φ (n) and gcd (e, φ (n)) = 1, then compute the integer d such that: e*d ≡ 1 mod φ (n) (e, n) is the public key and (d, n) is the private key.

RSA Signature Generation and Verification -

Signature of a message m is a straightforward modular exponentiation using the hash of the message and the private key. The signature s can be obtained by:

$$s = hash(m)^d \pmod{n}$$

A common hash algorithm used is SHA-1.To verify a signature s for message m, the signature must first be decrypted using the author's public key (e, n). The hash h is thus obtained by:

$$h = s^e \pmod{n}$$

If h matches hash (m), then the signature is valid. The message was signed by the author and the message has not been modified since signing.

E. X.509 STANDARD

The International Telecommunication Union X.509 specification delivers a set of standards for the implementation of a public key infrastructure and among them; one is being used for the structure of a digital public key certificate. The X.509 certificate standard has evolved many years ago. Version 1 was introduced in 1988 and assumed that by using the issuer distinguished name of a certificate, it would be possible to build a certificate chain going back to the root certificate. Version 2 which was introduced in 1993 presented the concept of unique identifiers to allow for the re-use of issuer distinguished names. The Version 3 was introduced in 1996 and allows anyone to define an extension and include it within their certificate. Version 1 certificates are generally used as root or self-signed certificates, version 2 certificates that have been outdated by version 3 certificates are now used for most applications. Version 4 type certificates are also known as Extended Validation (EV).

III. ECC CERTIFICATION AUTHORITY DESIGN

Our work can be divided into three blocks: to create the CA, to create the certificate request by the client and the CA signing the certificate request. In addition, we define three classes that are shared by all blocks:

- Key Generation is in charge of generating an elliptic curve key pair. It uses the ECIES scheme specified in ANSIX9.63 and IEEE P1363.
- X509Subject takes the component of the client data, splits it into its minimum units, and composes it again to eliminate possible misspellings.
- Certificate Utils generates .cer certificates (certificates signed by the CA, i.e., the identity of the user has been verified), and .der certificates (certificates that have not been signed yet, i.e. a client certificate request).

In the first block, classes to make the CA, we define the class CA cert EC. Its main task is to generate the X.509v3 root certificate and the PKCS#12 (Public Key Cryptography Standard, PKCS) with the corresponding CA's private key. A root certificate is a certificate that contains the public key of a CA. Clients can trust a CA only if a copy of the CA root certificate is in its trusted root certificate store. Moreover, the CA public key included in the CA root certificate is needed to verify the validity of any certificate that the CA issues. PKCS is a set of standard protocols to exchange secure information on the Internet using a public key infrastructure. PKCS#12 is a standard that specifies a portable format for storing a user's private key.

In the second block, classes to create the certificate request by the client. They involve the following tasks:

- Graphic Client launches an applet that the client uses to fill in the data necessary for the certificate request.
- Manager captures the applet events.

User Cert EC_DER generates a .der certificate (a certificate request). This is a certificate without a signature, hence, not valid yet. The .der certificate should be sent to the CA for signing. This class also generates a PKCS#12 to store the private key. Minimum Client takes the client .der certificate and sends it to the CA.

In the last block, classes so that the CA signs the certificate request. The goals of these classes are: Minimum Server, the CA is listening, waiting for a client request.

When a client connects to the CA, the CA checks if the client is authorized to demand the service. If the client is authorized then the CA signs the client certificate request using the DER2CER class. Afterwards, the CA returns the

signed certificate (.cer), ready for use, back to the client. The CA keeps waiting for new client requests DER2CER, this class firstly edits the client certificate request (PKCS#10), and adds new data such as key length, certificate serial number, period of validity and signature algorithm. In our case, the CA uses the ECDSA with SHA-1 (Elliptic Curve Digital Signature Algorithm with Secure Hash Algorithm 1) to sign certificates. DER2CER needs to know the key to access the CA secret key, which is located in the file server.pfx. It is necessary to know the CA secret key otherwise the CA would not be able to sign the client certificate. Afterwards, the client certificate (.cer) is created with Certificate Utils.

IV. RELATED WORKING AND DESIGN

The design consists of writing an OpenSSL code for creating the ECC based X.509v3 Digital certificate, configuring the digital certificate with the Thunderbird mail and establishing a trust for the created certificate and finally digitally signing and encrypting the e-mails. First we have to create a certification authority (CA). Then certificates need to be generated signed by this CA. For this a code need to written which retrieves the user's information and verify it. After verification, the CA signs and issues the certificate to the defined entity/person. The certificate generated will be of *cert.pfx* format.

The created certificate need to be imported to the Mozilla thunderbird and once it's imported and accepted by the trust. Once the certificate and trust get installed properly, the user can use it for sending e-mail from thunderbird digitally signed. The recipient of the mail will be able to get the public key of sender and with this he/she can send the encrypted e-mail with the support of S/MIME. This is just a proof of concept which we have illustrated in Mozilla Thunderbird. This certificate can be integrated with many applications which require the security and can be imported to any web browsers.

Internet Key Exchange (IKE) protocol is the most common usable mechanism to exchange keying materials and negotiate security associations between two distant entities. This study proposes a new flexible approach for complexity reduction and security improvement of the IKE implementation. In this study, an initial secret key negotiation based on Elliptic Curve Cryptography (ECC) for phase 1 of IKE has been proposed, which instead of RSA, uses ECC-based public key certificate for authentication of the entities. The use of X.509v3 certificates to carry out authentication tasks is an approach to improve security. The main advantage of ECC versus RSA is that for the same level of security it requires a much shorter key length. The purpose of this study is to design and implement a free open-source Certification Authority able to issue X.509v3 certificates using ECC. The result of this research may assist organizations to increase their security level in wireless devices and networks, in a costless way, by including authentication techniques based on ECC digital certificates. Elliptic Curve Cryptography (ECC) is emerging as an attractive alternative to traditional public-key cryptosystems (RSA, DSA, DH).

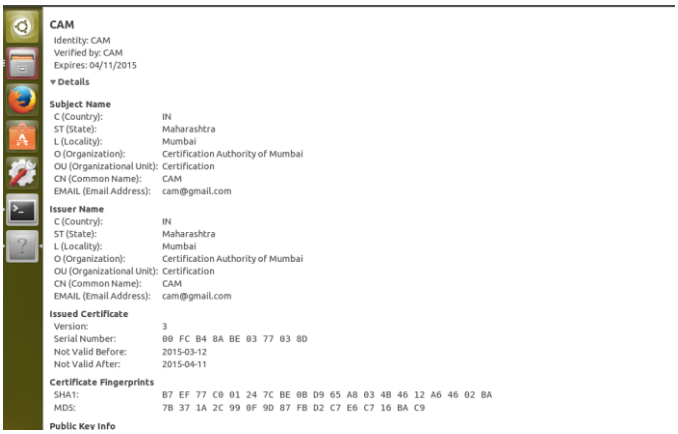

Fig.1. X.509 format certificate fields

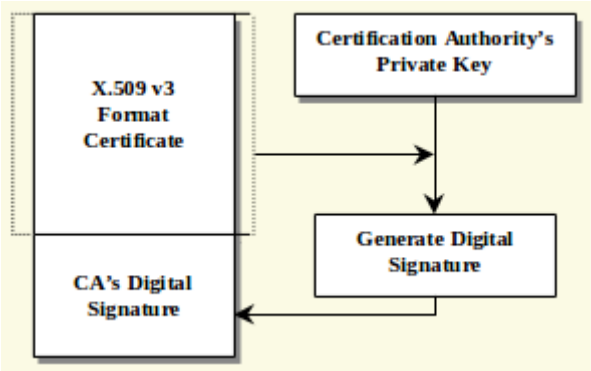

Fig.2. Certificate Authority

V. CONCLUSION

In this paper, we propose, design, and implement free open-source Certification Authority that generates X.509v3 certificates by using elliptic curve cryptography. As discussed, ECC provide security with less key size compared to any other algorithms present now and hence can be easily integrated in scenarios where, the resources like processing power, storage space, power consumption and bandwidth are limited. We believe this drift promises well for the future of Elliptic Curve Cryptography and not just for digital certificates. We have implemented ECC based certificates in OpenSSL and hence webservers and browsers can now use our variant to communicate securely.

REFERENCES

[1] Certicom, "Standards for efficient cryptography. Sec2: Recommended Elliptic Curve Domain Parameters", Released Standard Version 1.0, 2000. Available online <http://www.secg.org> . Last accessed March 3 rd ,2006.

[2] S. Vanstone, "Next generation security for wireless: elliptic curve cryptography", Computers & Security, Vol. 22, No. 5, pp. 412-415, 2003

[3] A Certification Authority for Elliptic Curve X.509v3 Certificates by Maria-Dolores Cano, Ruben Toledo-Valera, Fernando Cerdan

[4] Certificate Authority Schemes Using Elliptic Curve Cryptography, Rsa And Their Variants Simulation Using Ns2 by Shivkumar, S. and 2 G. Umamaheswari

[5] N. R. Potlapally, S. Ravi, A. Raghunathan, N. K. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols", IEEE Transactions on Mobile Computing, Vol. 5, No. 2, pp.128-143, 2005.

[6] W. Rao, Q. Gan, "The performance analysis of two digital signatures schemes based on secure charging protocol", Proc. International Conference on Wireless Communications, Networking, and Mobile Computing, Vol. 2, pp. 1180-1182, September 2005.

[7] ITU-T. Rec. X.509 (revised) the Directory Authentication Framework, International Telecommunication Union, Geneva, 1993

[8] ANSI X9.62, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", American Bankers Association, 1999.

[9] ANSI X9.63, "Elliptic Curve Key Agreement and Key Transport Protocols", American Bankers Association, 1999.

[10] NIST, "Recommended Elliptic Curves for Federal Government Use", July 1999, see http://csrc.nist.gov/csrc/fedstandards.html. R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev.,inpress

