GENERATION OF X.509V3 DIGITAL CERTIFICATES USING RSA FOR EMAIL APPLICATIONS

AN INDUSTRIAL INTERNSHIP REPORT

submitted by

SAKSHI GANERIWAL
(12BCE1054)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

VIT® UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore • Chennai
CHENNAI CAMPUS
Vandalur Kelambakkam Road, Chennai 600127

SEPTEMBER 2014

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore • Chennai
CHENNAI CAMPUS
Vandalur Kelambakkam Road, Chennai 600127

School of Computing Sciences and Engineering

DECLARATION BY THE CANDIDATE

I hereby declare that the Industrial Internship Report entitled "**GENERATION OF X.509V3 DIGITAL CERTIFICATES USING RSA FOR EMAIL APPLICATIONS**" submitted by me to VIT University, Chennai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bonafide industrial training undertaken by me under the supervision of **Kunal Abhishek at SETS.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Chennai: Signature of the Candidate

Date:

SOCIETY FOR ELECTRONIC TRANSACTIONS AND SECURITY (SETS)

(Registered under The Societies Registration Act XXI of 1860 Registration No. S.42605 of 2002)

Operational Headquarters
CIT Campus, MGR Knowledge City, Taramani, Chennai – 600 113
Phone : 91 44 66632520 website : www.setsindia.org

09th July, 2014

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Ms. Sakshi Ganeriwal bearing enrolment no. 12BCE1054 pursuing B.Tech (Computer Science and Engineering) at VIT University, Chennai has completed her internship at this establishment from 01st June 2014 to 04th July 2014 in the Information Security area with the topic titled "Generation of X.509v3 Digital Certificates using RSA for Email Applications".

During her internship she demonstrated very good technical skills with self-motivated attitude to learn new things.

We wish her all the best for her future endeavors.

Kunal Abhishek

(Kunal Abhishek)
Sr. Research Associate

SOCIETY FOR ELECTRONIC TRANSACTIONS AND SECURITY ★

VIT® UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore • Chennai
CHENNAI CAMPUS
Vandalur Kelambakkam Road, Chennai 600127

School of Computing Sciences and Engineering

<u>BONAFIDE CERTIFICATE</u>

This is to certify that the Industrial Internship Report entitled "**GENERATION OF X.509V3 DIGITAL CERTIFICATES USING RSA FOR EMAIL APPLICATIONS"** submitted by **SAKSHI GANERIWAL(12BCE1054)** to VIT University, Chennai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bonafide internship undertaken by him/her fulfills the requirements as per the regulations of this institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Dr.Ganesan R
Programme Chair
(B.Tech CSE)

Date:

Examiner (s) Signature
1. **2.**

ACKNOWLEDGEMENT

A summer project is a golden opportunity for learning and self development. I consider myself very lucky and honored to have so many wonderful people lead me through, in completion of this project.

I wish to express my indebted gratitude and special thanks to " **Mr. Kunal Abhishek**, Sr. Research Associate, **SETS**-Society for Electronic transaction and security, Chennai" who in spite of being extraordinarily busy with his duties, took time to hear, guide and keep me on the correct path and allowing me to carry out my industrial project work at their esteemed organization and extending during the training. I do not know where I would been without him.

I express my deepest thanks to **Sir Kunal Abhishek** for his guidance and support. He supported us by showing different methods of information collection about the company. He helped all time when we needed and he gave right direction toward completion of project.
I am also greatly thankful to all other employees of SETS for the kind cooperation and help. I also acknowledge their encouragement to me during the course of internship. Also, I would like to thank my fellow students who worked with me on this project and led to the completion of it.

SAKSHI GANERIWAL

Abstract

A digital certificate is a digital file that certifies the identity of an individual or an institution, or even a router seeking access to computer-based information. The most common use of a digital certificate is to verify that a user sending a message is who he or she claims to be, and to provide the receiver with the means to encode a reply. An individual wishing to send an encrypted message applies for a digital certificate from a Certificate Authority (CA).

In our project, in the first phase we understood the basic theoretical concepts of cryptography. We implemented, the RSA algorithm, an encryption algorithm used for generating digital certificates using C++ and JAVA.

In the second phase, we developed a digital certificate generation application with PHP and OpenSSL tools. The certificate authority manages the server and maintains a database of his/her clients who have applied for a digital certificate. Similarly, the clients have their own online secure portals through which they either generate their self-signed certificate or request for one from the certificate authority. For a self-signed certificate, the user needs to download the executable batch file provided to him on his portal, which enables him to generate his own certificate. For a CA-signed certificate, the client first downloads the executable batch file provided to him on his portal, which enables him to generate certificate signing request and sends it to the CA. The certificate authority sends the digital certificate to the client’s mail-box. The CA and the clients are connected through http server-client sessions.

The working of our project is portrayed through a user-friendly graphical interface with the look and format enhanced by Cascading style sheets.

TABLE OF CONTENTS

List of Figures:

List of Symbols, Abbreviations and Nomenclature:

SSL	Secure Sockets Layer
CA	certificate authority
PKI	public key infrastructure
Md5	Message Digest 5
DSA	Digital Signature Algorithm
MAC	Message Authentication Code
CSR	certificate signing request
SMTP	Simple Mail Transfer Protocol

CHAPTER 1-Introduction

1.1 About the Project

PKI SERVICES

PKI Services allow us to establish a PKI infrastructure and serve as a certificate authority for our internal and external users, issuing and administering digital certificates in accordance with our own organization's policies. Our users can use a PKI Services application to request and obtain certificates through their own Web browsers, while your authorized PKI administrators approve, modify, or reject these requests through their own Web browsers.

PKI Services supports Public Key Infrastructure for X.509.
It also supports the delivery of certificates through the Secure Sockets Layer (SSL) for use with applications that are accessed from a Web server.

Certificate Authority

A certificate authority, commonly called a CA, acts as a trusted third party to ensure that users who engage in e-business can trust each other. A certificate authority vouches for the identity of each party through the certificates it issues. In addition to proving the identity of the user, each certificate includes a public key that enables the user to verify and encrypt communications.

The trustworthiness of the parties depends on the trust that is placed in the CA that issued the certificates. To ensure the integrity of a certificate, the CA digitally signs the certificate as part of creating it, using its signing private key. Trying to alter a certificate invalidates the signature and renders it unusable.

Protecting the CA's private key is critical to the integrity of the CA. For this, the CA's private key is encrypted and stored with the CA itself. While creating a certificate, the private key is decrypted with the help of the passphrase and thus is to an extent protected.

As a CA using PKI Services, you can do the following:

- Track certificates you issue with an issued certificate list (ICL) that contains a copy of each certificate, indexed by serial number
- Track revoked certificates using certificate revocation lists (CRLs). When a certificate is revoked, PKI Services updates the CRL during the next periodic update.

PKI

The public key infrastructure (PKI) provides applications for performing the following types of security related activities:

- Authentication of different parties to engage in electronic transactions.
- Verifying the author of each message through its digital signature.
- Encryption of the content of each message.

PKI Services supports the following standards for public key cryptography:

- Secure Sockets Layer (SSL)
- Server certificates
- X.509v3 certificates
- RSA algorithms for encryption and signing
- DSA algorithms for signing
- MD5 hash algorithms

The problem faced is to find a user's public key and authenticate it. Trusting that a sender sending a message is who he or she claims to be may make the system vulnerable. Any person may pretend to be the sender and may misguide the receiver and hence the system.

There are several advantages of public key cryptography:

- **Asymmetry allows for easier key distribution**: We do not need a trusted third party to distribute keys.
- **Asymmetry provides proof of origin:** The secret key is something that only one entity knows.
- **Can be extended to form chains of trust:** Public Key certificate framework provides a natural way to model trust.

There are a few disadvantages as well:

- **Computation burden:** By its very nature, public key cryptography is not as fast or computationally efficient as symmetric cryptography.

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 this 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). The private key is given only to the requesting party and the public key is made publicly available (as part of a digital certificate) 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 (who can find out the public key from a public directory). In addition to encrypting messages (which ensures privacy), 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.

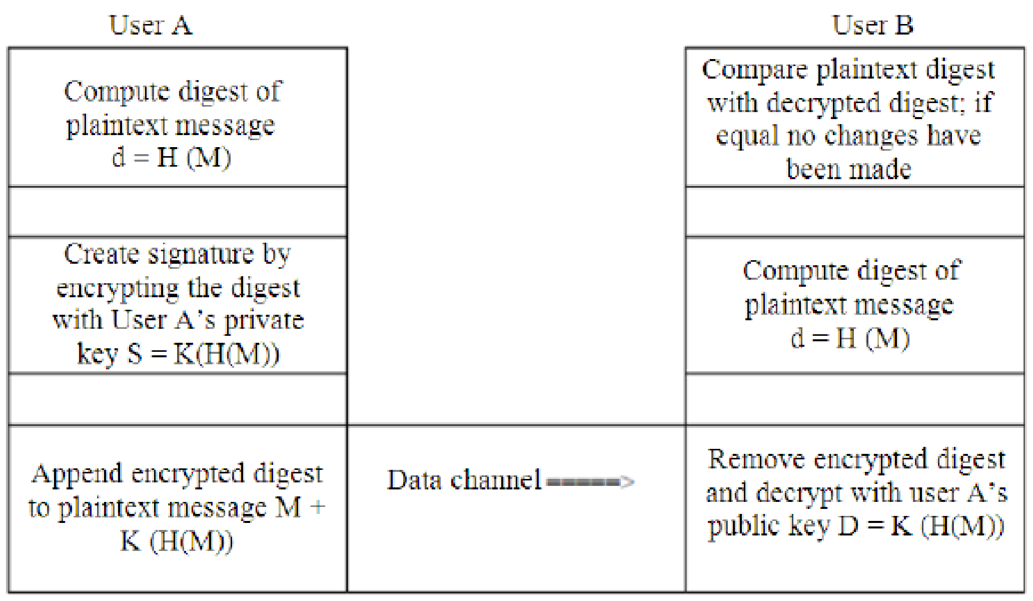

User A
User B
Compute digest of plaintext message d = H (M)
Create signature by encrypting the digest with User A's private key S = K(H(M))
Append encrypted digest to plaintext message M + K (H(M))
Data channel
Compare plaintext digest with decrypted digest; if equal no changes have been made
Compute digest of plaintext message d = H (M)
Remove encrypted digest and decrypt with user A's public key D = K (H(M))

Fig. 1. Generalized signature generation and authentication

The figure illustrates a generalized signature generation and verification process. The two users must agree to use the same hash function and have access to each other's public key.

It may also be useful to be able to encrypt the message. If that is the case, a digital signature is used to exchange a secret key with authentication, integrity, non-repudiation.

Following the exchange of the secret key, messages are encrypted with the secret key and exchanged. Each transmission can also contain a digest with signature to afford continued integrity and non-repudiation assurance. As indicated in Fig.1, a message digest produced by a hash function is used to confirm that the message was not changed in transit and that it truly represented the original message. 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 methods.)
- 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)

1.2 About the Company:

Society for Electronic Transactions & Security

SETS is a recognized Research & Development Institution established at the initiative of Office of Principal Scientific Advisor to the Government of India for Information and Communication Security Solutions as well as Cryptology. The Office of Principal Scientific Advisor to the Government of India through the Department of Science and Technology, Government of India is setting up a World class Advanced Facility for Information Security and Cryptology (AFISC) at SETS with "The State of The Art" facilities for conducting cutting edge R & D.

Advanced Facility for Information Security and Cryptology (AFISC) invites applications for the Post of Head (Network Security) from outstanding Research Scientists to build and lead a dedicated team of professionals.

CHAPTER 2-Overview of the Proposed System

2.1 System Preliminary Design

Normal web traffic is sent unencrypted over the Internet. That is, anyone with access to the right tools can snoop all of that traffic. This can lead to problems, especially where security and privacy is necessary. The Secure Socket Layer is used to encrypt the data stream between the web server and the web client (the browser).

SSL makes use of what is known as asymmetric cryptography, commonly referred to as public key cryptography (PKI). With public key cryptography, two keys are created, one public, one private. Anything encrypted with either key can only be decrypted with its corresponding key. Thus if a message or data stream were encrypted with the server's private key, it can be decrypted only using its corresponding public key, ensuring that the data only could have come from the server.

A certificate is not really necessary- the data is secure and cannot easily be decrypted by a third party. However, certificates do serve a crucial role in the communication process. The certificate, signed by a trusted Certificate Authority (CA), ensures that the certificate holder is really who he claims to be. Without a trusted signed certificate, your data may be encrypted; however, the party you are communicating with may not be whom you think. Without certificates, impersonation attacks would be much more common.

Following steps include the certificate generation process:

Step 1: Generation of Key Pairs

The **openssl** toolkit is used to generate an **RSA Private Key** and **CSR (Certificate Signing Request)**. It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.

The first step is to create a RSA Private Key. This key is a 1024 bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text.

Step 2: Generation of CSR (Certificate Signing Request)

Once the private key is generated a Certificate Signing Request can be generated. The CSR is then used in one of two ways. Ideally, the CSR will be sent to a Certificate Authority, who will verify the identity of the requestor and issue a signed certificate. The second option is to self-sign the CSR.

During the registration, the user will be prompted for several pieces of information for generation of the CSR. These are the X.509 attributes of the certificate. One of the prompts will be for "Common Name ". It is important that this field be filled in with the fully qualified domain name of the server to be protected by SSL. If the website to be protected will be https://something.com, then enter something.com at this prompt.

These are the attributes which the user will be asked for:

- Country Name (2 letter code) :e.g.-[GB]
- State or Province Name (full name) :e.g.- [Berkshire]
- Locality Name (city) :e.g.- [Newbury]
- Organization Name (company) :e.g.- [My Company Ltd]
- Organizational Unit Name (section)
- Common Name (your name or your server's hostname)
- Email Address
- A challenge password
- An optional company name

Step 3: Generation of Certificate

There are two ways certificate could be generated:

Self signed

A self-signed certificate is generated when either there is no plan on having a certificate signed by a CA, or to test new SSL implementation while the CA is signing client's certificate. This temporary certificate will generate an error in the client browser to the effect that the signing certificate authority is unknown and not trusted.

In cryptography and computer security, a **self-signed certificate** is an identity certificate that is signed by the same entity whose identity it certifies. This term has nothing to do with the identity of the person or organization that actually performed the signing procedure. In technical terms a self-signed certificate is one signed with its own private key.

A digital signature from a certificate authority (CA) attests that a particular public key certificate is valid . Users or their software on their behalf, check that the private key used to sign some certificate matches the public key in the CA's certificate. Since CA certificates are often signed by other, "higher-ranking," CAs, there must necessarily be a highest CA, which provides the ultimate in attestation authority in that particular PKI scheme.

Obviously, the highest-ranking CA's certificate can't be attested by some other higher CA and so that certificate can only be "self-signed." Such certificates are also termed **root certificates**. Clearly, the lack of mistakes or corruption in the issuance of such certificates is critical to the operation of its associated PKI; they should be issued with great care.

In a web of trust certificate scheme there is no central CA, and so identity certificates for each user can be self-signed. In this case, it has additional signatures from other users which are evaluated to determine whether a certificate should be accepted as correct.

CA signed

In cryptography, a **certificate authority** or **certification authority** (**CA**) is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. In this model of trust relationships, a CA is a trusted third party that is trusted by both the subject of the certificate and the party relying upon the certificate. CAs are characteristic of many PKI schemes.

Trusted certificates are typically used to make secure connections to a server over the Internet. A certificate is required in order to avoid the case that a malicious party which happens to be on the path to the target server pretends to be the target(man-in-the-middle attack). The client uses the CA certificate to verify the CA signature on the server certificate, as part of the checks before establishing a secure connection. Client software—for example, browsers— includes a set of trusted CA certificates.

A CA issues digital certificates that contain a public key and the identity of the owner. The matching private key is not made available publicly, but kept secret by the end user who generated the key pair. The certificate is also a confirmation or validation by the CA that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate. A CA's obligation in such schemes is to verify an applicant's credentials, so that users and relying parties can trust the information in the CA's certificates. The certificate authority is responsible for saying "yes, this person is who they say they are, and we, the CA, certify that". If the user trusts the CA and can verify the CA's signature, then (s) he can also assume that a certain public key does indeed belong to whoever is identified in the certificate.

RSA Implementation

RSA is a public key cryptographic algorithm. 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

RSA involves a **public key** and a **private key**. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

2. Compute n=pq.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

3. Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$, where φ is Euler's totient function.

4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are coprime.
   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65,537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

5. Determine d as $d \equiv e^{-1}$ (mod φ(n)); i.e., d is the multiplicative inverse of e (modulo φ(n)).

- This is more clearly stated as: solve for d given $d \cdot e \equiv 1$ (mod φ(n)).
- This is often computed using the extended Euclidean algorithm. Using the pseudo code in the Modular integers section, inputs a and n correspond to e and φ (n), respectively.
- d is kept as the private key exponent.

The *public key* consists of the modulus *n* and the public (or encryption) exponent *e*. The *private key* consists of the modulus *n* and the private (or decryption) exponent *d*, which must be kept secret. *p*, *q*, and ϕ(*n*) must also be kept secret because they can be used to calculate *d*.

Euler's totient function

Euler's totient or **phi function**, φ(n), is an arithmetic function that counts the totatives of n, that is, the positive integers less than or equal to n that are relatively prime to n.

if n is a positive integer, then φ(n) is the number of integers k in the range $1 \leq k \leq n$ for which gcd(n, k) = 1. The totient function is a multiplicative function, meaning that if two numbers m and n are relatively prime (to each other), then φ (mn) = φ (m) φ (n).

For example,

let $n = 9$. Then gcd (9, 3) = gcd (9, 6) = 3 and gcd (9, 9) = 9. The other six numbers in the range $1 \leq k \leq 9$, that is 1, 2, 4, 5, 7 and 8 are relatively prime to 9. Therefore, φ (9) = 6. As another example, φ (1) = 1 since gcd (1, 1) = 1.

Euler's product formula

It states

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right),$$

Where the product is over the distinct prime numbers dividing n.

The proof of Euler's product formula depends on two important facts:

Φ (*n*) is multiplicative

This means that if gcd $(m, n) = 1$, then $\varphi(mn) = \varphi(m)\,\varphi(n)$.

(Sketch of proof: let *A*, *B*, *C* be the sets of residue classes modulo-and-coprime to *m*, *n*, *mn* respectively; then there is a bijection between $A \times B$ and *C*, by the Chinese remainder theorem.)

$\boldsymbol{\varphi(p^k) = p^k - p^{k-1} = p^{k-1}(p - 1)}$

That is, if *p* is prime and $k \geq 1$ then

$$\varphi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1) = p^k\left(1 - \frac{1}{p}\right).$$

Example:

$$\varphi(36) = \varphi\left(2^2 3^2\right) = 36\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 36 \cdot \frac{1}{2} \cdot \frac{2}{3} = 12.$$

Encryption

A transmits a public key *(n, e)* to B and keeps the private key secret. B then wishes to send message *M* to A.

B first turns *M* into an integer *m*, such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the cipher text c corresponding to

$$c \equiv m^e \pmod{n}$$

This can be done quickly using the method of exponentiation by squaring. B then transmits *c* to A.

Note that at least nine values of *m* will yield a cipher text *c* equal to *m,* but this is very unlikely to occur in practice.

Decryption

A can recover *m* from *c* by using her private key exponent *d* via computing

$$m \equiv c^d \pmod{n}$$

Given *m*, A can recover the original message *M* by reversing the padding scheme.

(there are more efficient methods of calculating c^d using the precomputed values)

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 (mod 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 = se (mod 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.

Suppose A uses B's public key to send B an encrypted message. In the message, A can claim to be A but B has no way of verifying that the message was actually from A since anyone can use B’s public key to send an encrypted messages. In order to verify the origin of a message, RSA can also be used to sign a message.

Suppose A wishes to send a signed message to B. A can use its own private key to do so. A produces a hash value of the message, raises it to the power of *d* (modulo*n*) (as when decrypting a message), and attaches it as a "signature" to the message. When B receives the signed message,B uses the same hash algorithm in conjunction with A's public key. B raises the signature to the power of *e* (modulo *n*) (as when encrypting a message), and compares the resulting hash value with the message's actual hash value. If the two agree, B knows that the author of the message was in possession of A's private key, and that the message has not been tampered with since.

Mailing the Certificate

PHP does have a mail() function to send an email from the web server itself, but there are some drawbacks to doing this. Instead, using an external mail server to handle these emails.

Delivering automated emails reliably can be very difficult. Email providers and internet service providers (ISPs) take spam very seriously.
Email providers and ISPs take a number of factors into account when deciding whether or not to deliver a message. The **reputation of the server** sending the message is one of these factors. If the site runs on shared hosting, reputation is shared with every other website on that server. If any one of those sites gets identified as a spammer, the whole server can get blacklisted: any email sent from the server could go undelivered. Emails should be sent from a server with a really good reputation.

As the website increases in traffic, shared hosting will be switched off probably and scale out to use **multiple web servers** to handle the load. If the servers that generate web pages are kept separate from the servers that send emails, they can be scaled independently.

Whether an own separate server is hosted or a third-party service is used, Simple Mail Transfer Protocol (SMTP) is the easiest method to have a web server interact with an email server. Most of the third-party services have their own unique APIs, but they typically support SMTP. We used SMTP. To integrate with them more tightly through an API to take advantage of some more advanced features keep using SMTP to make it easier to switch to a different service in the future.

To use SMTP in our PHP code, we used PHPMailer. This third-party library provides the settings and functions you need to send email using a variety of methods, including SMTP. SMTP example can be found in the test_smtp_basic.php file in the examples folder of the library.

Password Encryption

Hashing:

The **MD5** message-digest algorithm is a cryptographic hash function which produces a 128-bit (16-byte) hash value, expressed in text format as a 32 digit hexadecimal number. This algorithm has been used to encrypt the user's password.

2.2 Software Required

PHP – PHP Hypertext Protocols

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page – for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used in stand alone graphical applications.

PHP is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

PHP Mailer master

It is a full-featured email creation and transfer class for PHP.

Class Features

- Integrated SMTP support - send without a local mail server
- Send emails with multiple TOs, CCs, BCCs and REPLY-TOs
- Multipart/alternative emails for mail clients that do not read HTML email
- Support for UTF-8 content and 8bit, base64, binary, and quoted-printable encodings
- SMTP authentication with LOGIN, PLAIN, NTLM and CRAM-MD5 mechanisms over SSL and TLS transports

The only PHP function that supports this is the mail() function. However, it does not provide any assistance for making use of popular features such as HTML-based emails and attachments.

The PHP mail() function usually sends via a local mail server, typically fronted by a sendmail binary on Linux, BSD and OS X platforms, however, Windows usually doesn't include a local mail server; PHPMailer's integrated SMTP implementation allows email sending on Windows platforms without a local mail server.

SMTP - Simple Mail Transfer Protocol

SMTP a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP. In addition, SMTP is generally used to send messages from a mail client to a mail server. This is why you need to specify both the POP or IMAP server and the SMTP server when you configure your e-mail application.

OpenSSL

This module uses the functions of OpenSSL for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. See appendix.

CHAPTER3-Design of The System

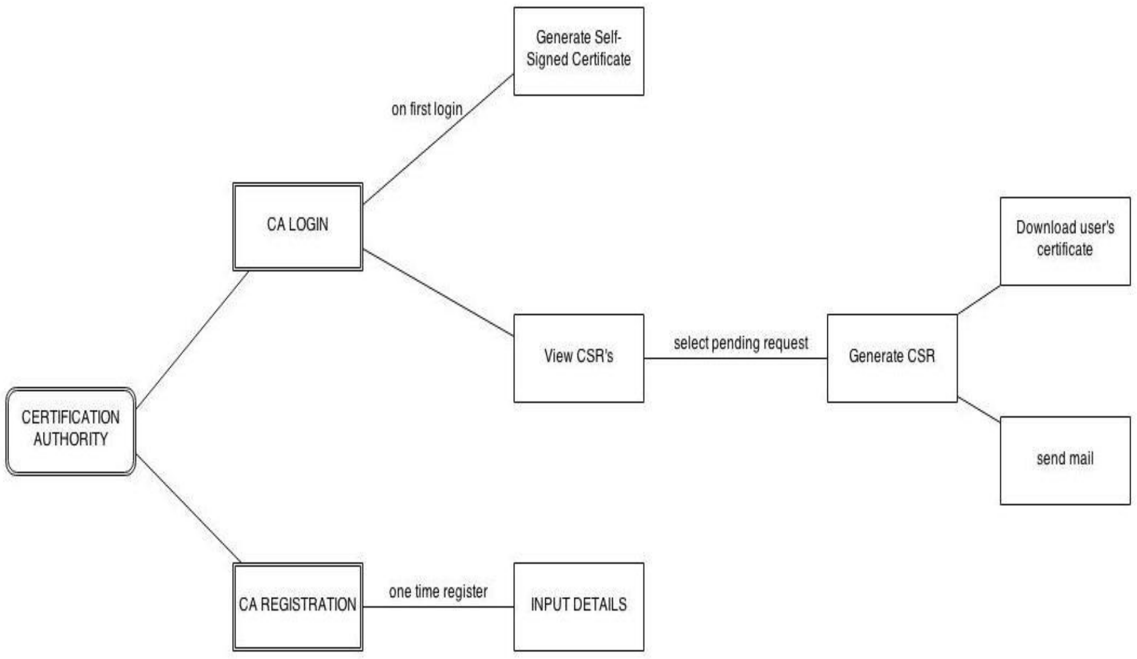
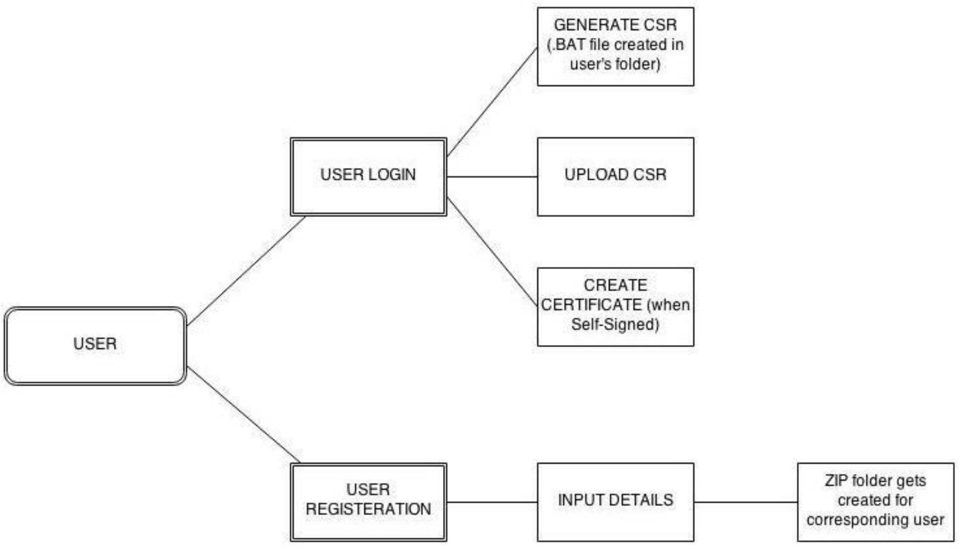


Figure 3.1

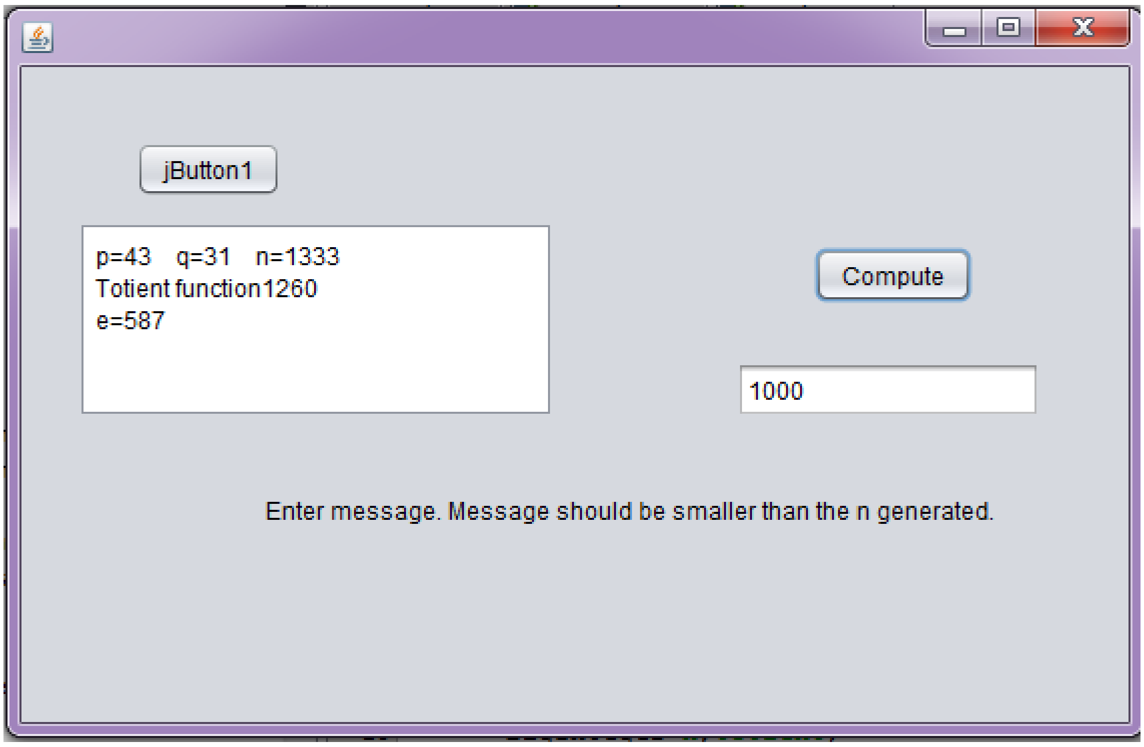


USER
USER LOGIN
GENERATE CSR (.BAT file created in user's folder)
UPLOAD CSR
CREATE CERTIFICATE (when Self-Signed)
USER REGISTERATION
INPUT DETAILS
ZIP folder gets created for corresponding user

Figure 3.2

CHAPTER 4-Implementation of the system

4.1 RSA



```
p=43   q=31   n=1333
Totient function1260
e=587
```

1000

Enter message. Message should be smaller than the n generated.

Figure 4.1

Codes:

4.1.1 Implementing RSA

4.1.1.1 Generates random prime numbers

```
public void randomgen(int u,int v)
{
    Random rand = new Random();
    a= rand.nextInt((v-u+1))+u;
if(a==p||a==0||a==1)
randomgen(u,v);
checkprime(a,u,v);
}
```

4.1.1.2 Primality test

```
public void checkprime(inta,intu,int v)
{
int m=0;
for(int i=2;i<=Math.sqrt(a);i++)
{
if(a%i==0)
 {
m++;
break;
 }
else
    m=0;
}
if(m>0)
randomgen(u,v);


}
```

4.1.2 Decryption key generation

```
   d=((y+totient)%totient);
```

4.1.2.1 Cipher text generation

```
publicBigDecimal power(BigDecimala,int b)
{
   c=new BigDecimal("1");
   m=new BigDecimal("1");
for(int i=1;i<=b;i++)
   {
     c=c.multiply(a);
   }
return c;
}
```

4.1.2.2 Retrieving message from textbox

```
public void msggenerate()
{
msg= Long.parseLong(jTextField1.getText());
}
```

4.3CSR Generation

4.3.1CA signed

```
@echo off
set path1=%~dp0
cd bin
openssl genrsa -out userkey.pem -passout pass:56789 -des3 2048
openssl req -new -key userkey.pem -passin pass:56789 -passout pass:56789 -out
csr.pem -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
copy userkey.pem %path1%
copy csr.pem %path1%@echo off
set path1=%~dp0
cd bin
openssl genrsa -out userkey.pem -passout pass:56789 -des3 2048
openssl req -new -key userkey.pem -passin pass:56789 -passout pass:56789 -out
csr.pem -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
copy userkey.pem %path1%
copy csr.pem %path1%@echo off
set path1=%~dp0
cd bin
openssl genrsa -out userkey.pem -passout pass:56789 -des3 2048
openssl req -new -key userkey.pem -passin pass:56789 -passout pass:56789 -out
csr.pem -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
copy userkey.pem %path1%
copy csr.pem %path1%@echo off
set path1=%~dp0
cd bin
openssl genrsa -out userkey.pem -passout pass:56789 -des3 2048
openssl req -new -key userkey.pem -passin pass:56789 -passout pass:56789 -out
csr.pem -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
copy userkey.pem %path1%
copy csr.pem %path1%
```

4.3.2 Self signed-

```
@echo off
set path1=%~dp0
cd bin
openssl genrsa -out userkey.pem -passout pass:56789 -des3 2048
openssl req -new -key userkey.pem -passin pass:56789 -passout pass:56789 -out
csr.pem -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
openssl req -new -x509 -key userkey.pem -passin pass:56789 -passout pass:56789 -
out selfcert.crt -days 365 -subj
/C=IN/ST=tamil/L=nadu/O=chennai/OU=kodai/CN=mudit/emailAddress=sg010@y
mail.com
copy userkey.pem %path1%
copy csr.pem %path1%
copy selfcert.crt %path1%
```

Zip file creation-

```
$query="select Common_name from user_details where Email= '$batname'";
$result=mysqli_query($con,$query);
$row = mysqli_fetch_array($result);
$batname=$row['Common_name'];
$zip = new ZipArchive;

$zipname=$batname."/"."download.zip";
$batname=$filename;
if ($zip->open($zipname) === TRUE)
 {
            $zip->addFile($batname, $batname);
            $zip->close();
            echo 'ok';
}
```

4.4 Upload file-

```
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Error: " . $_FILES["file"]["error"] . "<br>";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
     if (file_exists($path."/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
```

```
    move_uploaded_file($_FILES["file"]["tmp_name"],
    $path."/" . $_FILES["file"]["name"]);
    echo "Stored in: " . $path."/" . $_FILES["file"]["name"];
    }
```

4.5 Email-

```
require 'class.phpmailer.php';
$mail = new PHPMailer();
$mail->IsSMTP();
$mail->Mailer = 'smtp';
$mail->SMTPAuth = true;
$mail->Host = 'smtp.gmail.com'; $mail->Port = 587;
$mail->SMTPSecure = 'tls';
$mail->Username = "casets9@gmail.com";  //mail id used to send certificate to user
$mail->Password = "caset9";
$mail->IsHTML(true); // for HTML formatted emails
$mail->SingleTo = true; // to send a same email to multiple users. multiple emails
will be sent one-by-one.
$mail->From = "casets9@gmail.com";
$mail->FromName = "Certificate Authority";
$mail->addAddress($umail,"User 1");
//$mail->addAddress("user.2@gmail.com","User 2");
//$mail->addCC("user.3@ymail.com","User 3");
//$mail->addBCC("user.4@in.com","User 4");
$mail->Subject = "Testing PHPMailer with localhost";
$mail->Body = "Hi,<br /><br />This system is working perfectly.";
$path="C:/wamp/www/".$t."/".$filename;
$mail->AddAttachment($path);
if(!$mail->Send())
   echo "Message was not sent <br />PHPMailer Error: " . $mail->ErrorInfo;
else
   echo "Message has been sent";
```

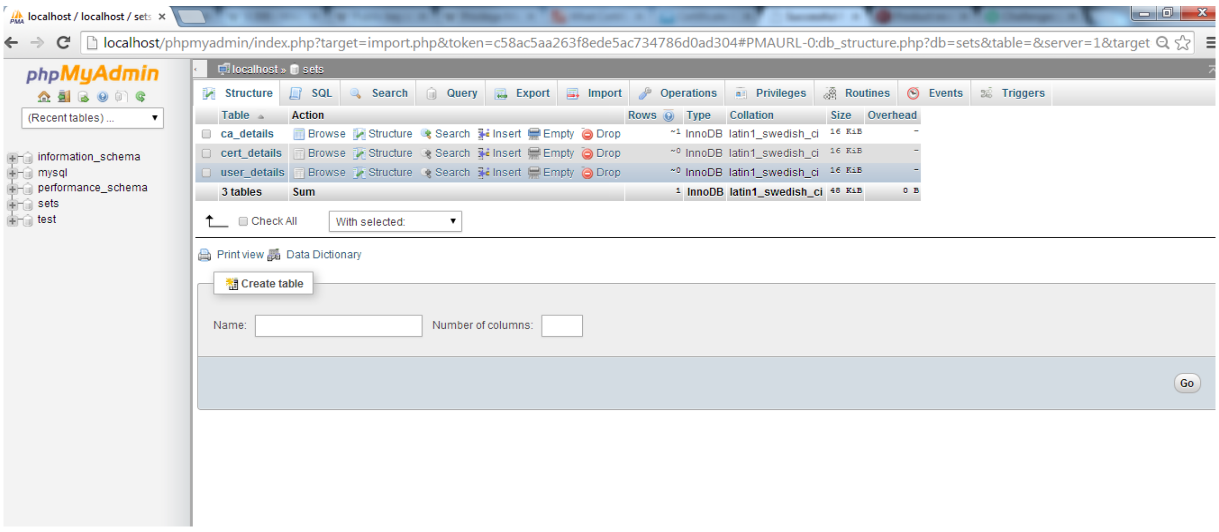
4.6 Snapshots-

Database consisted of 3 tables:

1.ca_details(information required for ca login)

2.user_details(information required for user login)

3. cert_details(information for pending csr)



localhost/phpmyadmin/index.php?target=import.php&token=c58ac5aa263f8ede5ac734786d0ad304#PMAURL-0:db_structure.php?db=sets&table=&server=1&target

localhost » sets

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	ca_details	Browse Structure Search Insert Empty Drop	~1	InnoDB	latin1_swedish_ci	16 KiB	-
☐	cert_details	Browse Structure Search Insert Empty Drop	~0	InnoDB	latin1_swedish_ci	16 KiB	-
☐	user_details	Browse Structure Search Insert Empty Drop	~0	InnoDB	latin1_swedish_ci	16 KiB	-
	3 tables	Sum	1	InnoDB	latin1_swedish_ci	48 KiB	0 B

Figure 4.2

Index

Page:

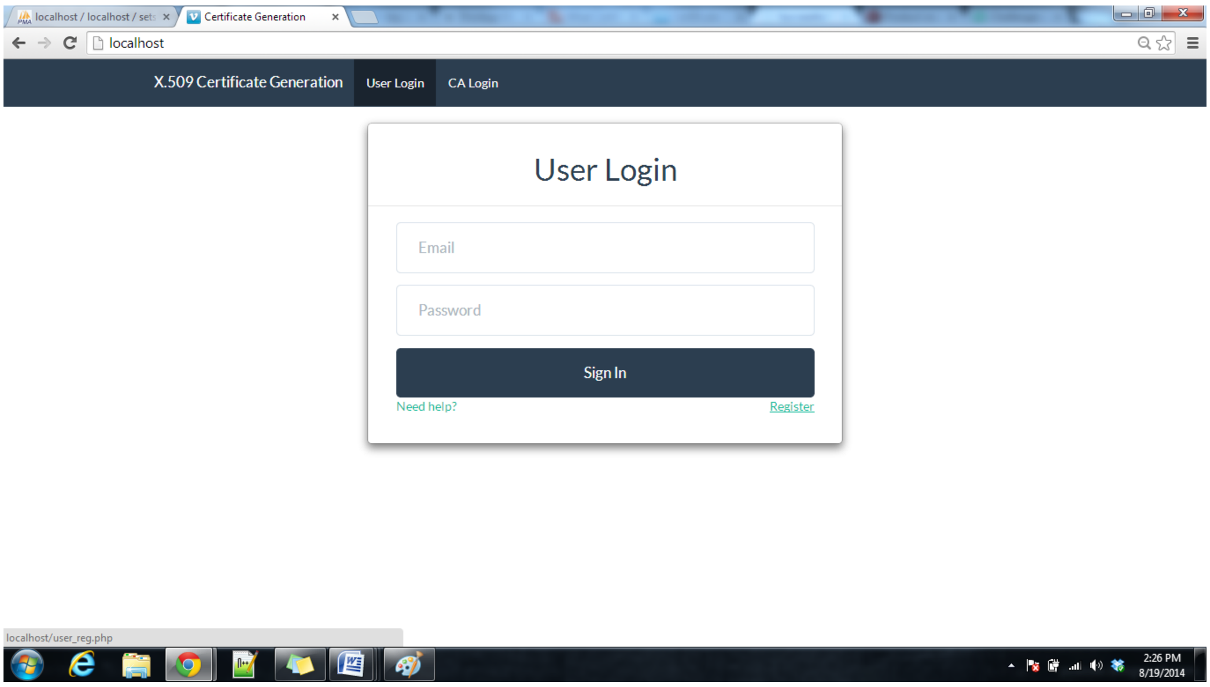
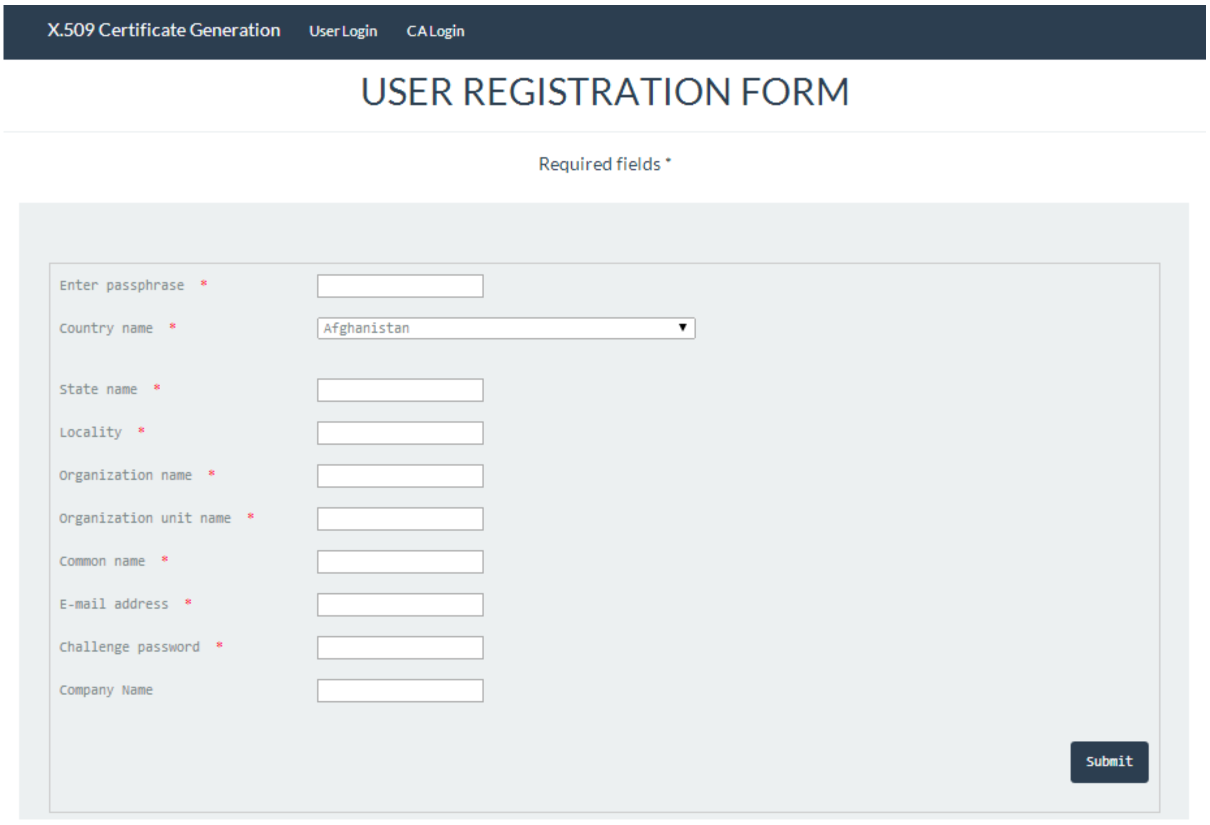


Figure 4.3

User registration form:


X.509 Certificate Generation User Login CA Login

USER REGISTRATION FORM

Required fields *

Enter passphrase *	
Country name *	Afghanistan
State name *	
Locality *	
Organization name *	
Organization unit name *	
Common name *	
E-mail address *	
Challenge password *	
Company Name	

Submit

Figure 4.4

CA login:

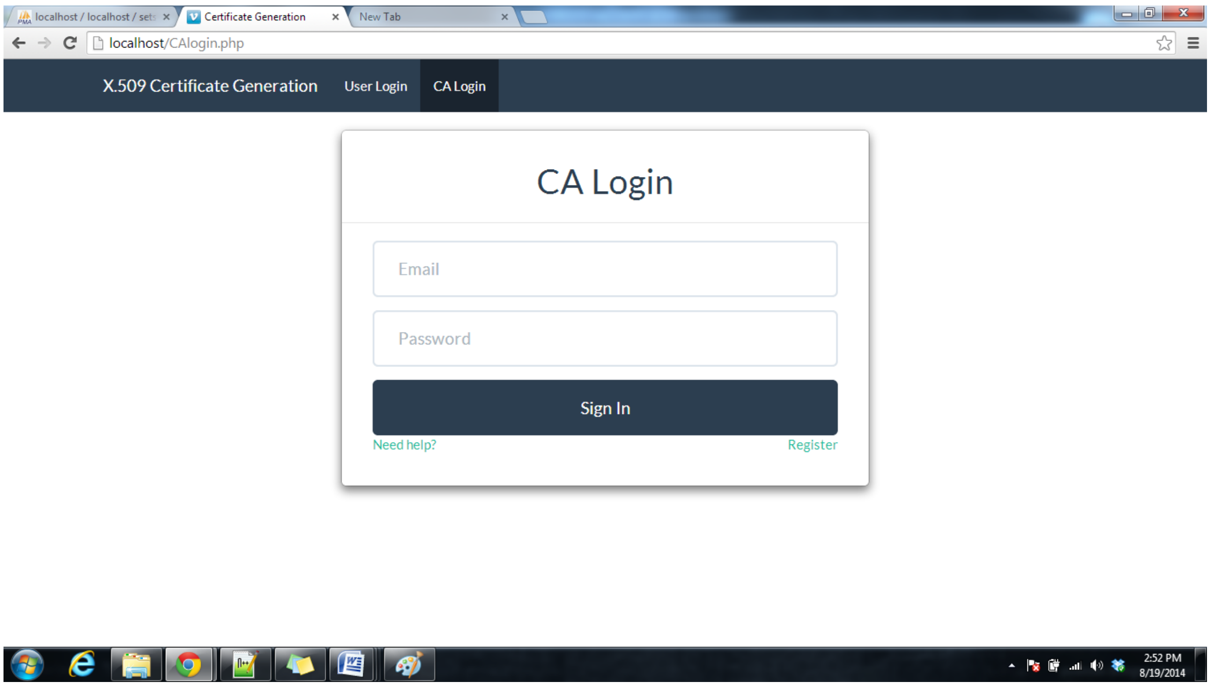
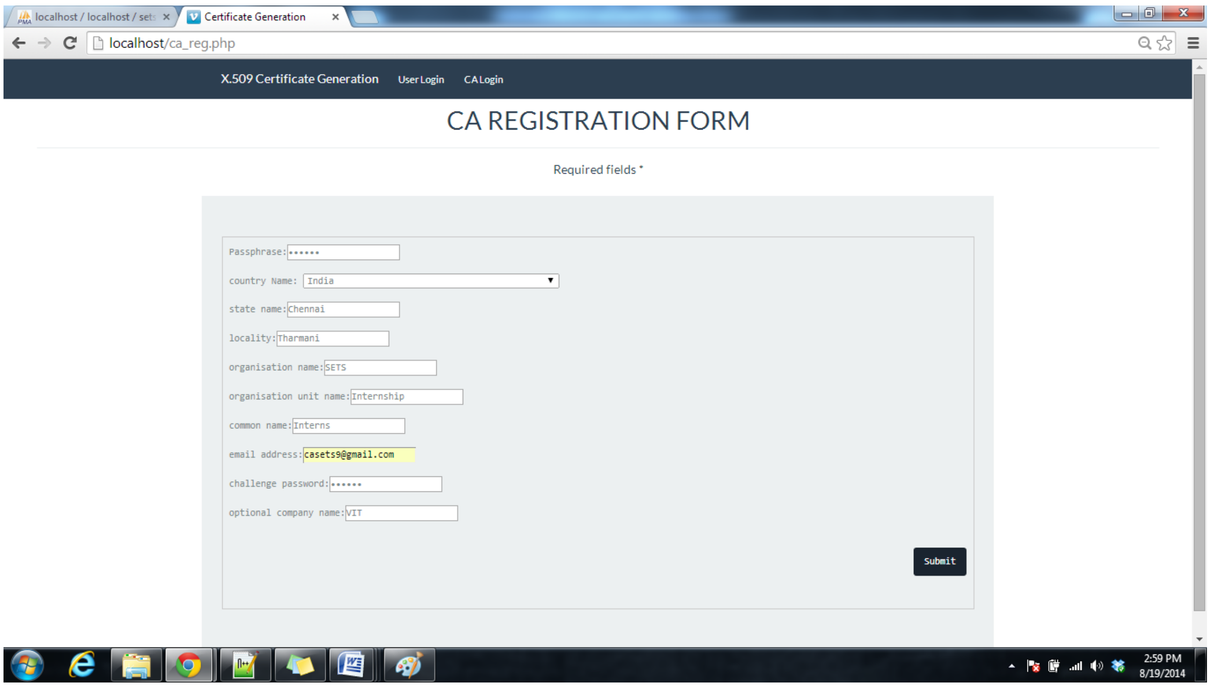


Figure 4.5

CA registration form:



Passphrase:••••••

country Name: India

state name:Chennai

locality:Tharmani

organisation name:SETS

organisation unit name:Internship

common name:Interns

email address:casets9@gmail.com

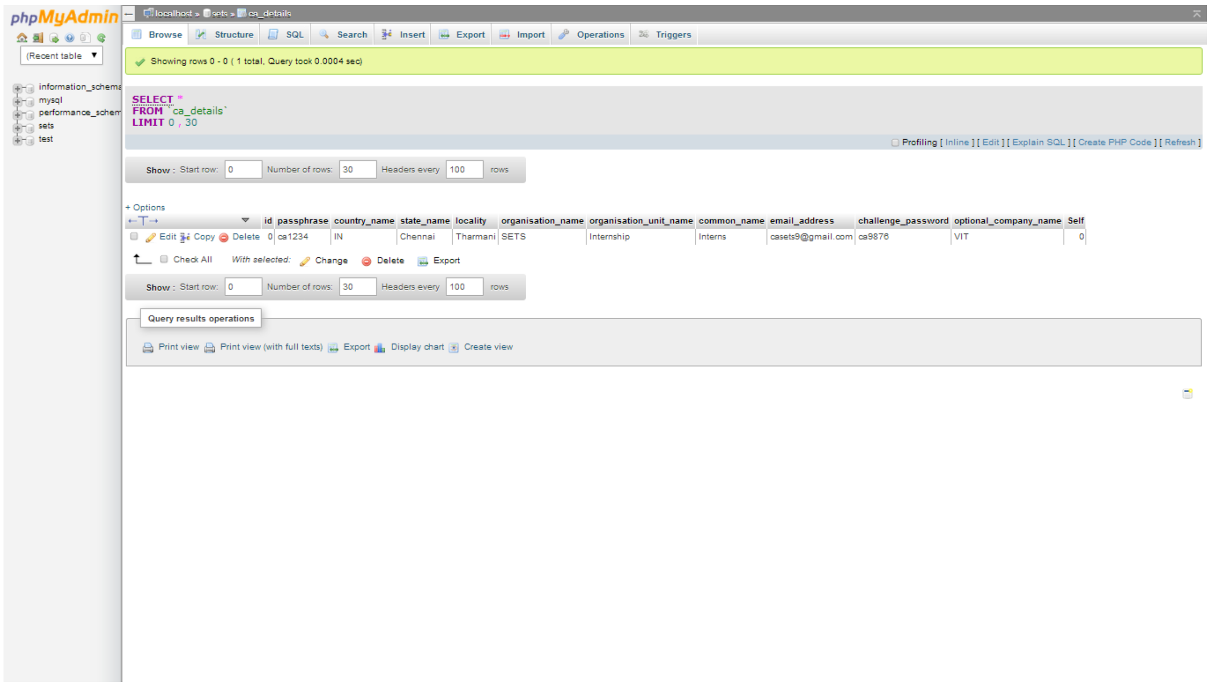
challenge password:••••••

optional company name:VIT

Figure 4.6

Corresponding entries in the database:

For ca:



Showing rows 0 - 0 (1 total, Query took 0.0004 sec)

```
SELECT *
FROM `ca_details`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

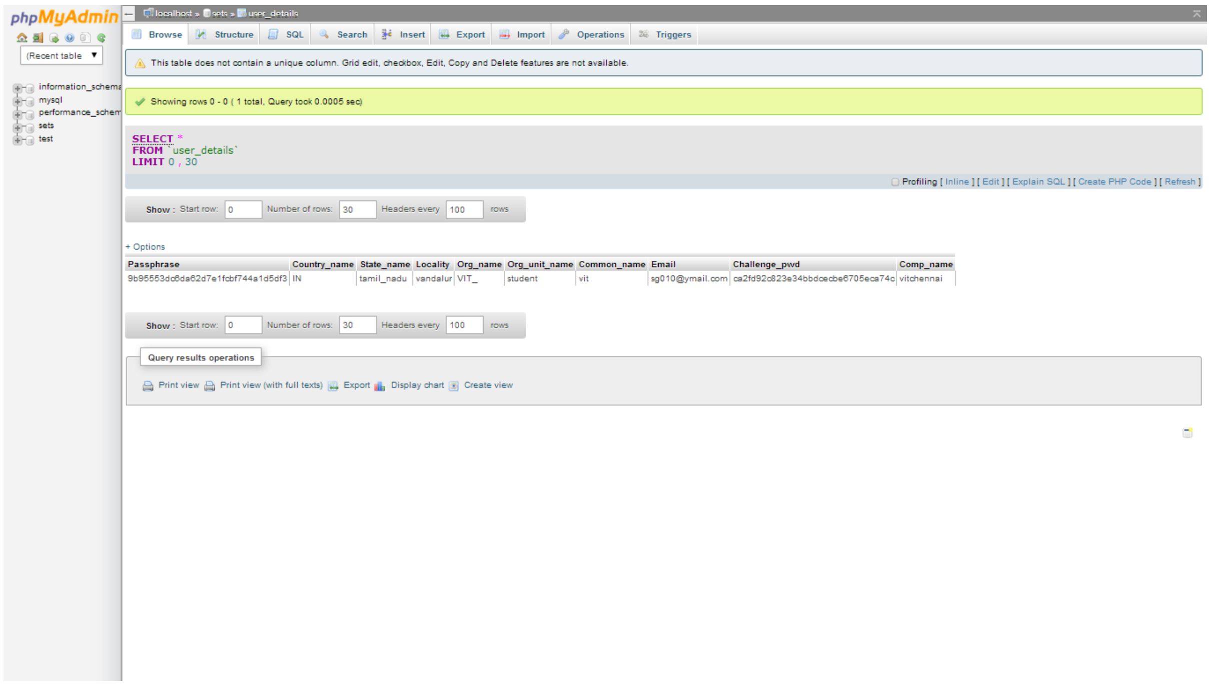
id	passphrase	country_name	state_name	locality	organisation_name	organisation_unit_name	common_name	email_address	challenge_password	optional_company_name	Self
0	ca1234	IN	Chennai	Tharmani	SETS	Internship	Interns	casets9@gmail.com	ca9876	VIT	0

Check All With selected: Change Delete Export

Query results operations: Print view, Print view (with full texts), Export, Display chart, Create view

Figure 4.7

For user:



This table does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0005 sec)

```
SELECT *
FROM `user_details`
LIMIT 0 , 30
```

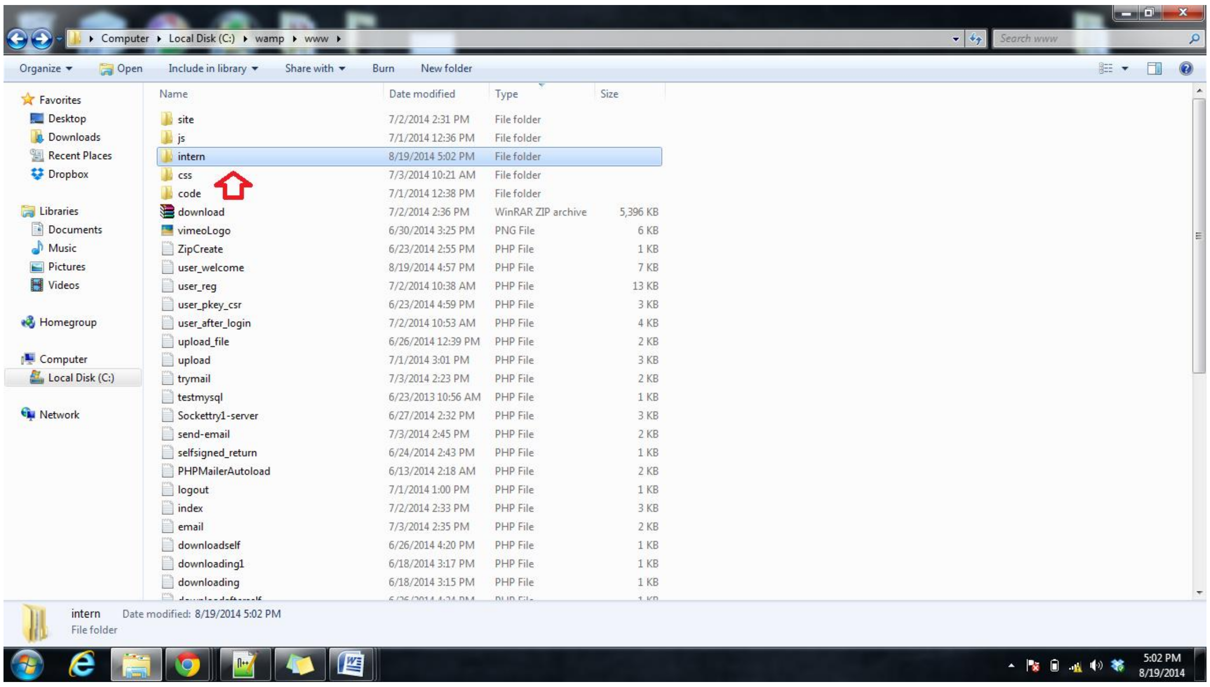
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Passphrase	Country_name	State_name	Locality	Org_name	Org_unit_name	Common_name	Email	Challenge_pwd	Comp_name
9b95553dc8da62d7e1fcbf744a1d5df3	IN	tamil_nadu	vandalur	VIT_	student	vit	sg010@ymail.com	ca2fd92c823e34bbdcecbe6705eca74c	vitchennai

Query results operations: Print view, Print view (with full texts), Export, Display chart, Create view

Figure 4.8

Corresponding files being created:



Name	Date modified	Type	Size
site	7/2/2014 2:31 PM	File folder	
js	7/1/2014 12:36 PM	File folder	
intern	8/19/2014 5:02 PM	File folder	
css	7/3/2014 10:21 AM	File folder	
code	7/1/2014 12:38 PM	File folder	
download	7/2/2014 2:36 PM	WinRAR ZIP archive	5,396 KB
vimeoLogo	6/30/2014 3:25 PM	PNG File	6 KB
ZipCreate	6/23/2014 2:55 PM	PHP File	1 KB
user_welcome	8/19/2014 4:57 PM	PHP File	7 KB
user_reg	7/2/2014 10:38 AM	PHP File	13 KB
user_pkey_csr	6/23/2014 4:59 PM	PHP File	3 KB
user_after_login	7/2/2014 10:53 AM	PHP File	4 KB
upload_file	6/26/2014 12:39 PM	PHP File	2 KB
upload	7/1/2014 3:01 PM	PHP File	3 KB
trymail	7/3/2014 2:23 PM	PHP File	2 KB
testmysql	6/23/2013 10:56 AM	PHP File	1 KB
Sockettry1-server	6/27/2014 2:32 PM	PHP File	3 KB
send-email	7/3/2014 2:45 PM	PHP File	2 KB
selfsigned_return	6/24/2014 2:43 PM	PHP File	1 KB
PHPMailerAutoload	6/13/2014 2:18 AM	PHP File	2 KB
logout	7/1/2014 1:00 PM	PHP File	1 KB
index	7/2/2014 2:33 PM	PHP File	3 KB
email	7/3/2014 2:35 PM	PHP File	2 KB
downloadself	6/26/2014 4:20 PM	PHP File	1 KB
downloading1	6/18/2014 3:17 PM	PHP File	1 KB
downloading	6/18/2014 3:15 PM	PHP File	1 KB

intern Date modified: 8/19/2014 5:02 PM
File folder

5:02 PM
8/19/2014

Figure 4.9

After user login:

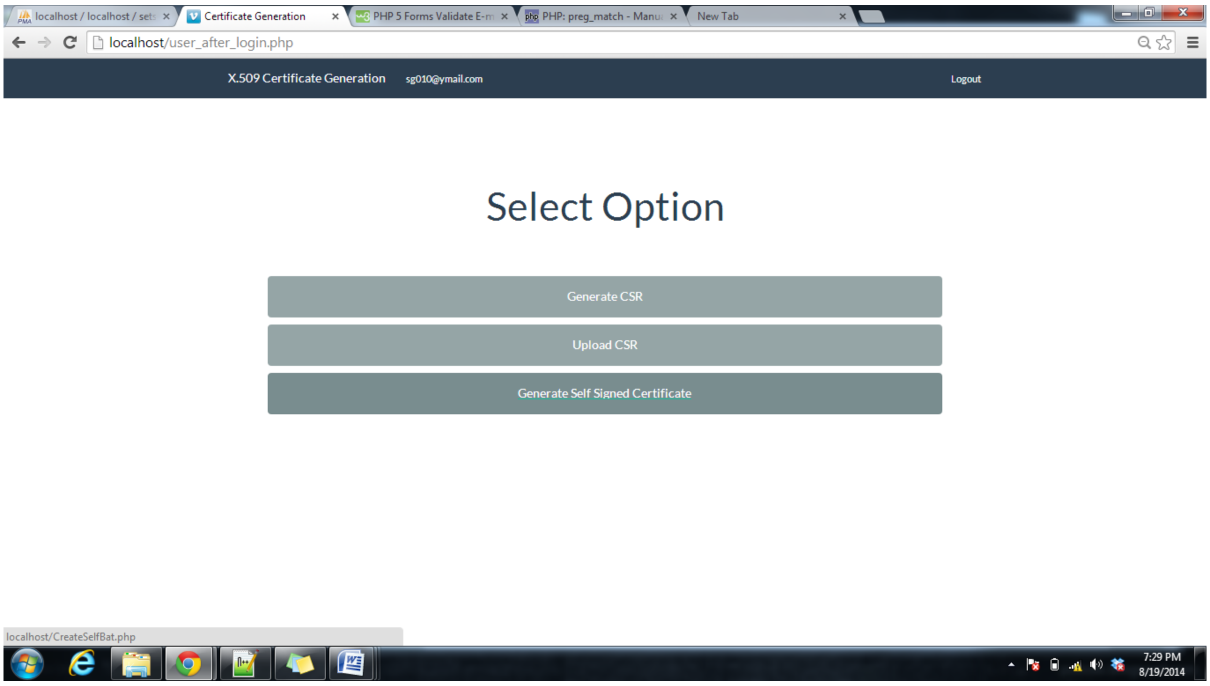


Figure 4.10

On selecting self-signed certificate generation:

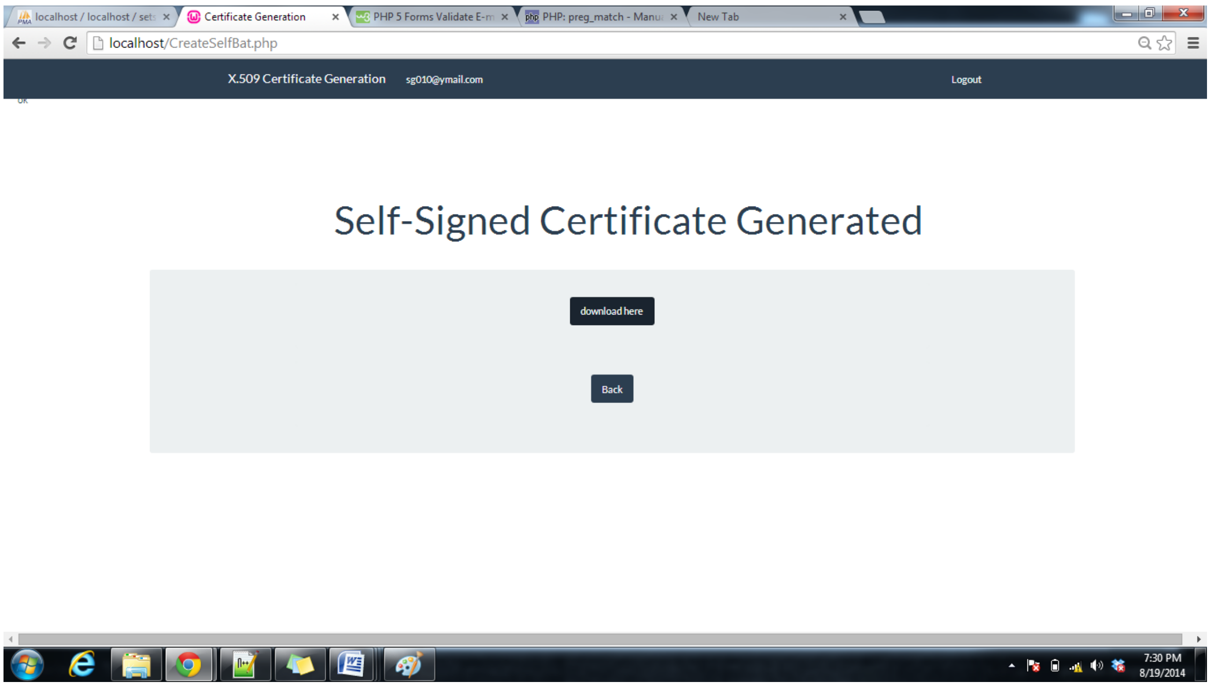


Figure 4.11

On clicking download here:

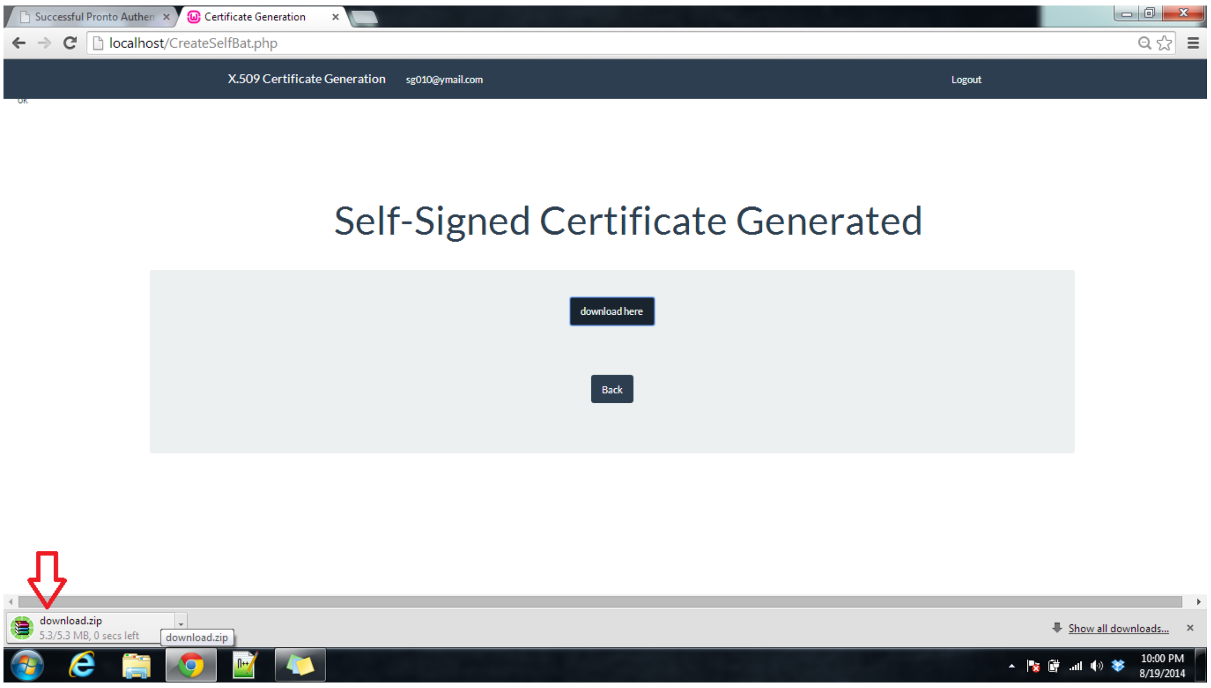
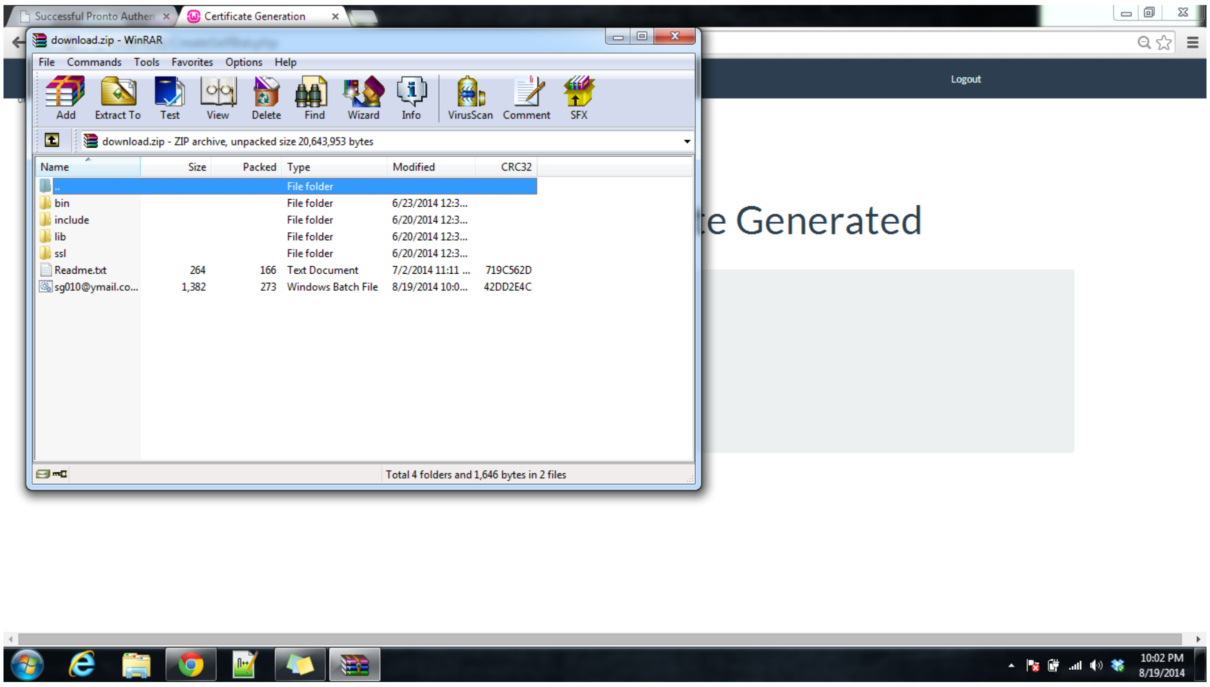


Figure 4.12

Extracting the files:

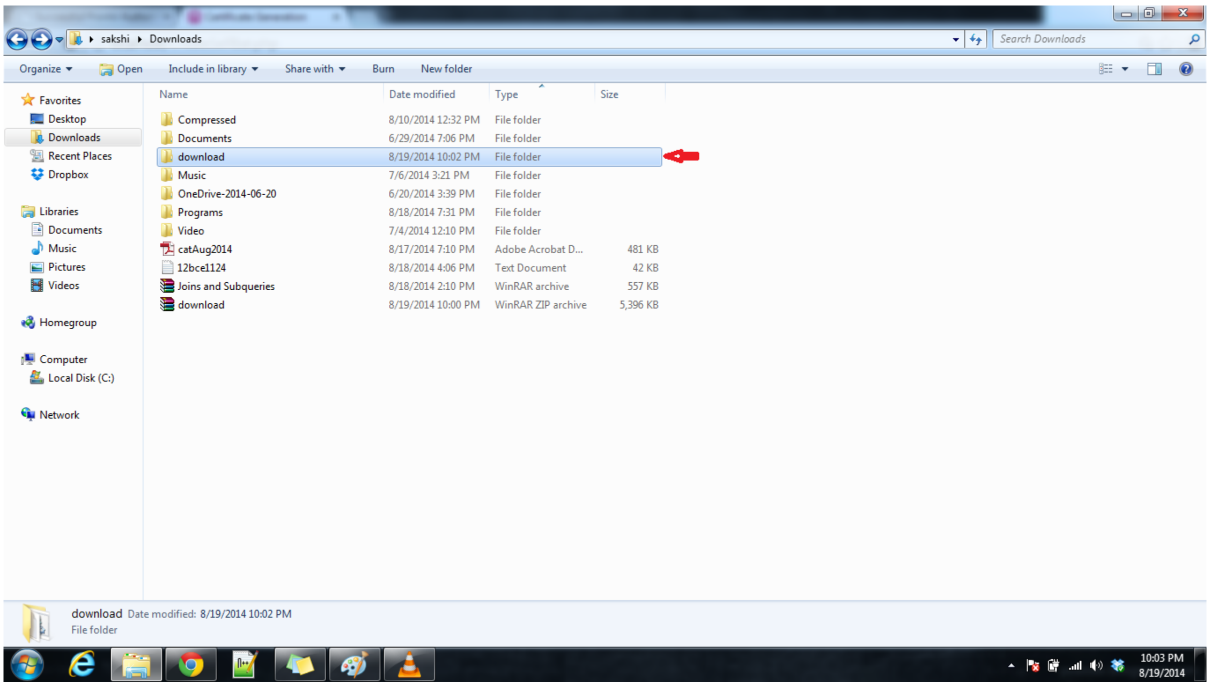


download.zip - ZIP archive, unpacked size 20,643,953 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
bin			File folder	6/23/2014 12:3...	
include			File folder	6/20/2014 12:3...	
lib			File folder	6/20/2014 12:3...	
ssl			File folder	6/20/2014 12:3...	
Readme.txt	264	166	Text Document	7/2/2014 11:11 ...	719C562D
sg010@ymail.co...	1,382	273	Windows Batch File	8/19/2014 10:0...	42DD2E4C

Total 4 folders and 1,646 bytes in 2 files

Figure 4.14

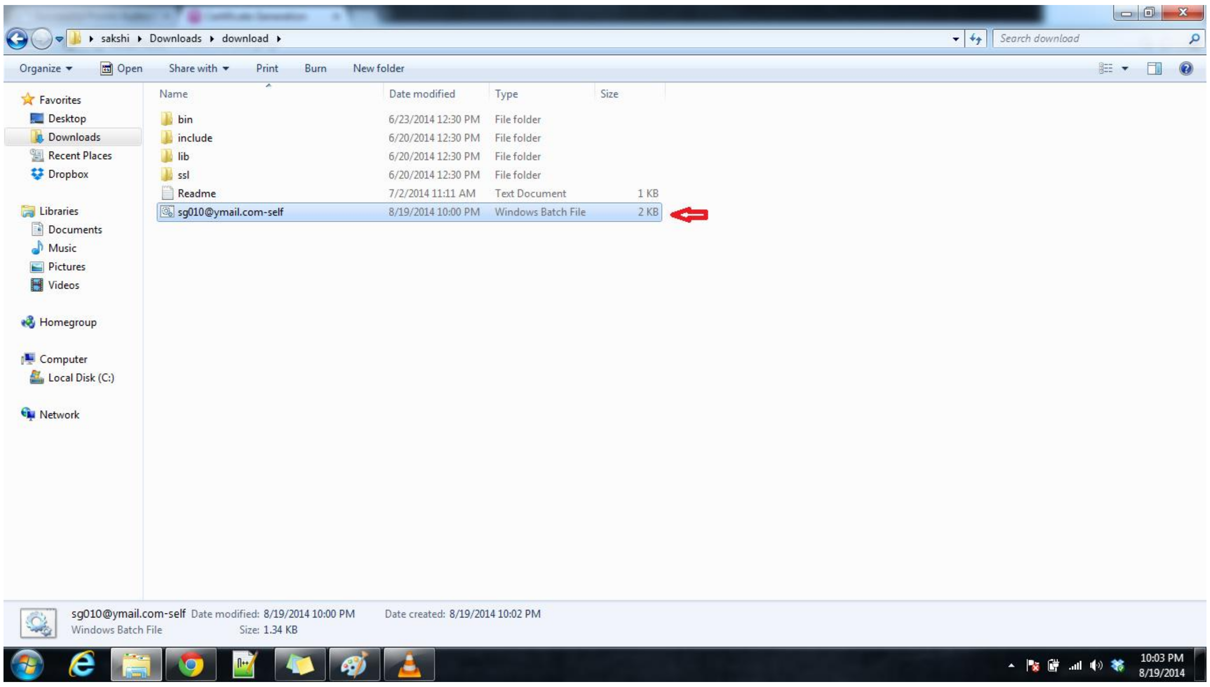


Name	Date modified	Type	Size
Compressed	8/10/2014 12:32 PM	File folder	
Documents	6/29/2014 7:06 PM	File folder	
download	8/19/2014 10:02 PM	File folder	
Music	7/6/2014 3:21 PM	File folder	
OneDrive-2014-06-20	6/20/2014 3:39 PM	File folder	
Programs	8/18/2014 7:31 PM	File folder	
Video	7/4/2014 12:10 PM	File folder	
catAug2014	8/17/2014 7:10 PM	Adobe Acrobat D...	481 KB
12bce1124	8/18/2014 4:06 PM	Text Document	42 KB
Joins and Subqueries	8/18/2014 2:10 PM	WinRAR archive	557 KB
download	8/19/2014 10:00 PM	WinRAR ZIP archive	5,396 KB

download Date modified: 8/19/2014 10:02 PM
File folder

Figure 4.14

On running the batch file:

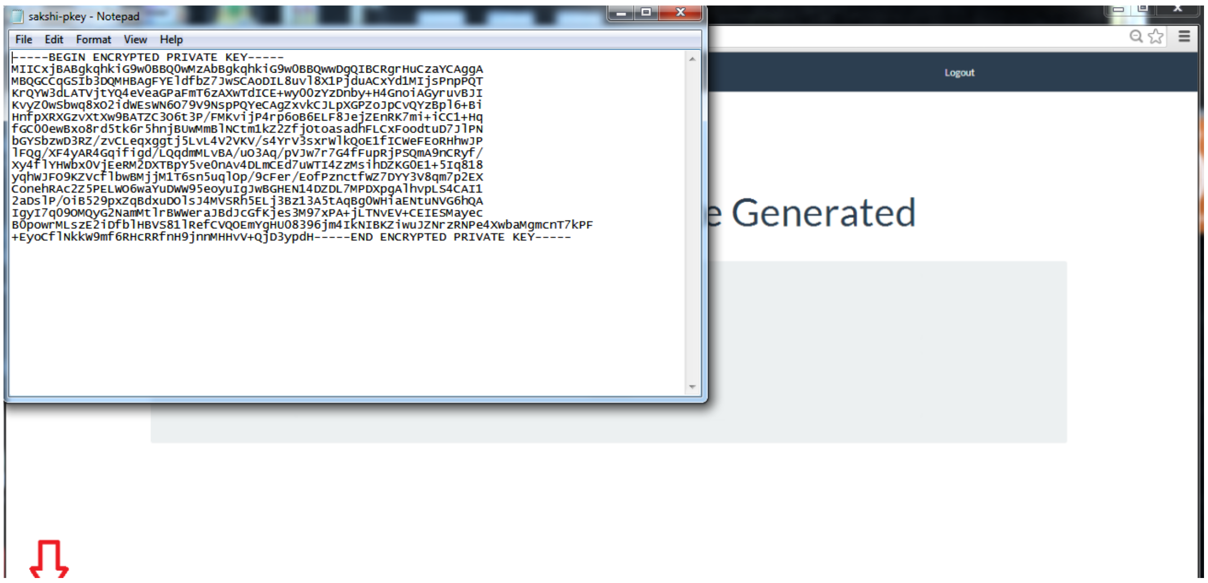


Name	Date modified	Type	Size
bin	6/23/2014 12:30 PM	File folder	
include	6/20/2014 12:30 PM	File folder	
lib	6/20/2014 12:30 PM	File folder	
ssl	6/20/2014 12:30 PM	File folder	
Readme	7/2/2014 11:11 AM	Text Document	1 KB
sg010@ymail.com-self	8/19/2014 10:00 PM	Windows Batch File	2 KB

sg010@ymail.com-self Date modified: 8/19/2014 10:00 PM Date created: 8/19/2014 10:02 PM
Windows Batch File Size: 1.34 KB

Figure 4.15

We get, Private key:

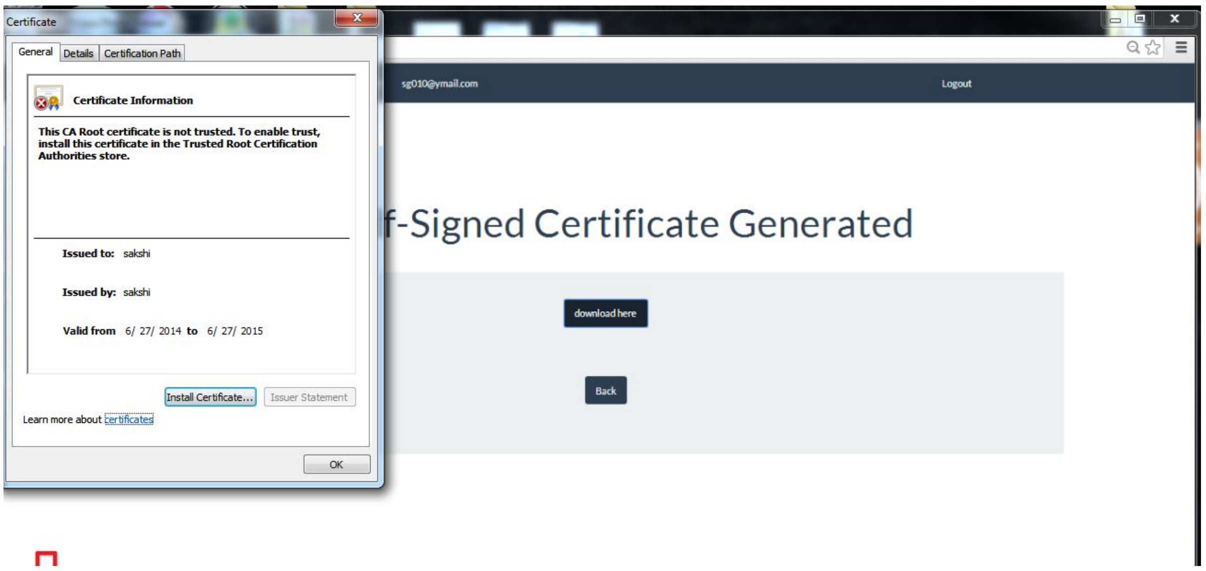


sakshi-pkey - Notepad

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIICxjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIBCRgrHuCzaYCAggA
MBQGCCqGSIb3DQMHBAgFYEldfbZ7JwSCAoDIL8uvl8X1PjduACxYd1MIjsPnpPQT
KrQYW3dLATVjtYQ4eVeaGPaFmT6zAXwTdICE+wy00zYzDnby+H4GnoiAGyruvBJI
KvyZ0wSbwq8xO2idWEsWN6O79V9NspPQYeCAgZxvkCJLpXGPZoJpCvQYzBpl6+Bi
HnfpXRXGzvXtXw9BATZC3O6t3P/FMKvijP4rp6oB6ELF8JejZEnRK7mi+iCC1+Hq
fGC0OewBxo8rd5tk6r5hnjBUwMmBlNCtm1kZ2ZfjOtoasadhFLCxFoodtuD7JlPN
bGYSbzwD3RZ/zvCLeqxggtj5LvL4V2VKV/s4YrV3sxrWlkQoE1fICWeFEoRHhwJP
lFQg/XF4yAR4Gqifigd/LQqdmMLvBA/uO3Aq/pVJw7r7G4fFupRjPSQmA9nCRyf/
Xy4flYHWbx0VjEeRM2DXTBpY5ve0nAv4DLmCEd7uWTI4ZzMsihDZKG0E1+5Iq818
yqhWJFO9KZVcflbwBMjjM1T6sn5uqlop/9cFer/EofPznctfWZ7DYY3V8qm7p2EX
ConehRAc2Z5PELWO6waYuDWW95eoyuIgJwBGHEN14DZDL7MPDXpgAlhvpLS4CAI1
2aDslP/OiB529pxZqBdxuDOlsJ4MVSRh5ELj3Bz13A5tAqBg0WHiaENtuNVG6hQA
IgyI7q09OMQyG2NamMtlrBWWeraJBdJcGfKjes3M97xPA+jLTNvEV+CEIESMayec
B0powrMLszE2iDfblHBVS81lRefCVQOEmYgHU08396jm4IkNIBKZiwuJZNrzRNPe4XwbaMgmcnT7kPF
+EyoCflNkkW9mf6RHcRRfnH9jnnMHHvV+QjD3ypdH-----END ENCRYPTED PRIVATE KEY-----
```

Figure 4.16

Self-signed certificate:


Certificate

General | Details | Certification Path

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: sakshi

Issued by: sakshi

Valid from 6/ 27/ 2014 **to** 6/ 27/ 2015

Install Certificate... | Issuer Statement

Learn more about certificates

OK

sg010@ymail.com Logout

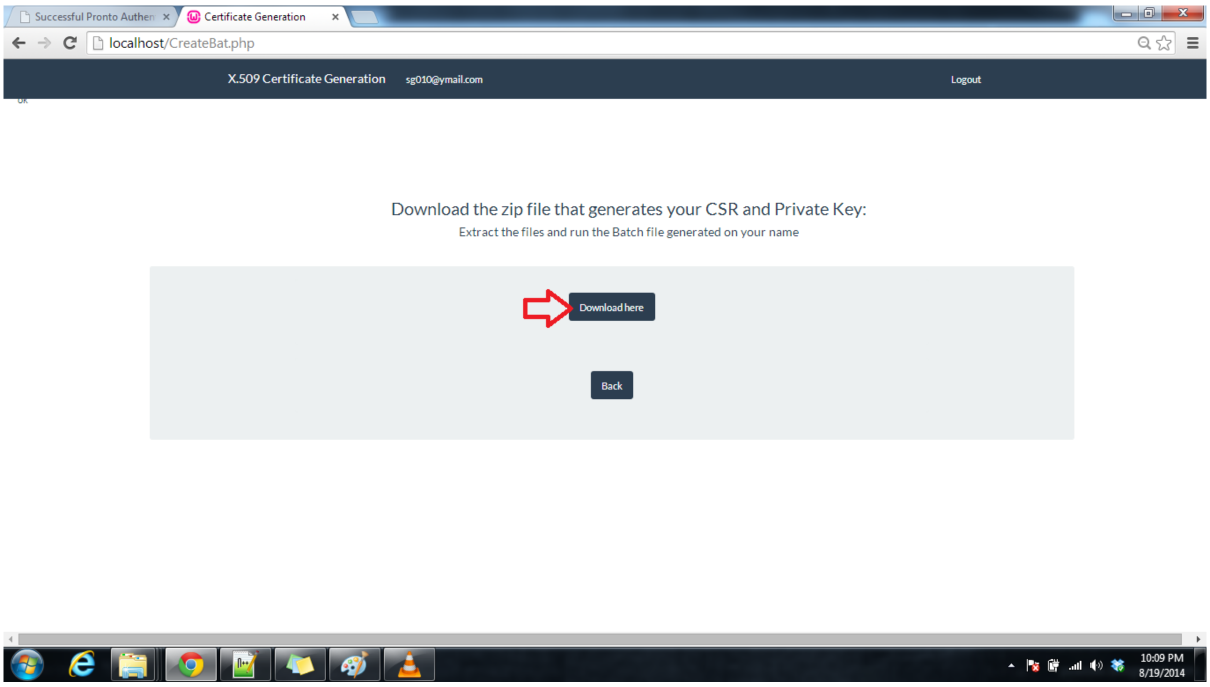
f-Signed Certificate Generated

download here

Back

Figure 4.17

For generating csr which is uploaded in another link:


Successful Pronto Authen | Certificate Generation

localhost/CreateBat.php

X.509 Certificate Generation sg010@ymail.com Logout

Download the zip file that generates your CSR and Private Key:

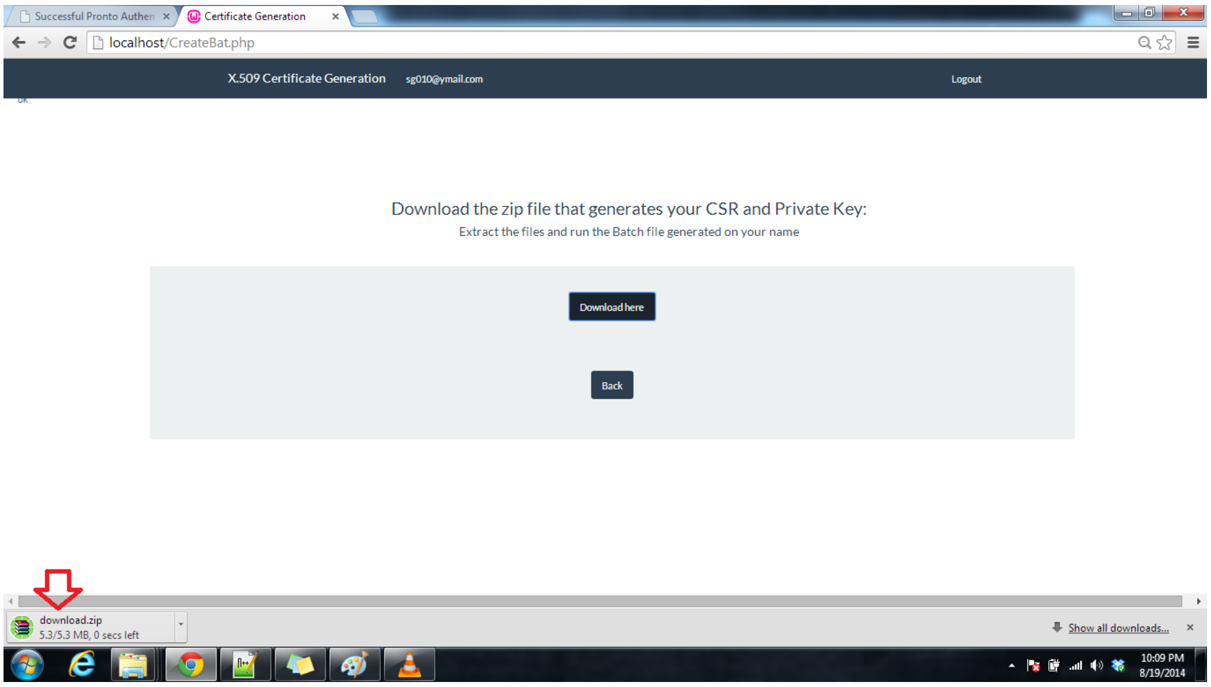
Extract the files and run the Batch file generated on your name

Download here

Back

10:09 PM 8/19/2014

Figure 4.18

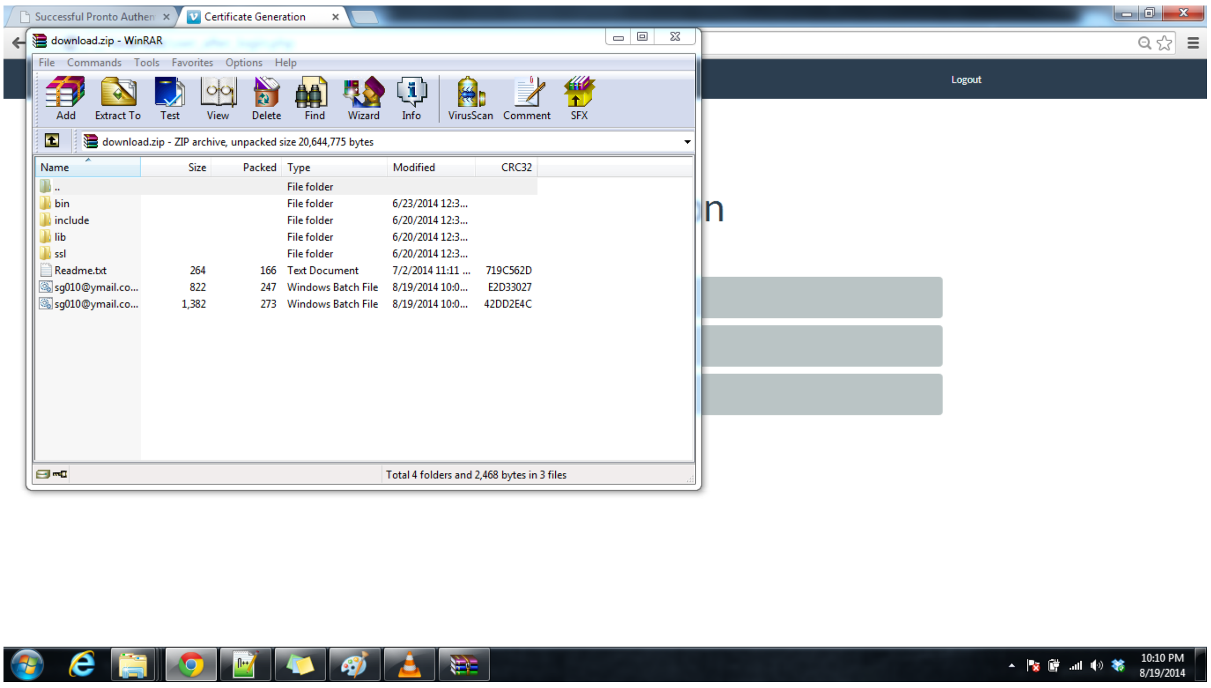


Download the zip file that generates your CSR and Private Key:

Extract the files and run the Batch file generated on your name

Figure 4.19

Extracting the files and generating the CSR:

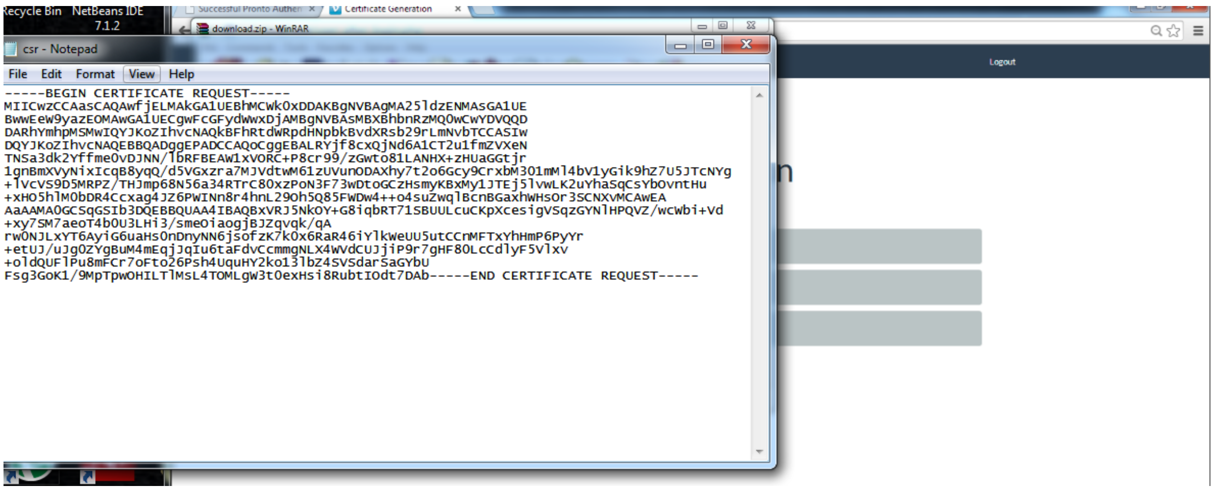


download.zip - ZIP archive, unpacked size 20,644,775 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
bin			File folder	6/23/2014 12:3...	
include			File folder	6/20/2014 12:3...	
lib			File folder	6/20/2014 12:3...	
ssl			File folder	6/20/2014 12:3...	
Readme.txt	264	166	Text Document	7/2/2014 11:11 ...	719C562D
sg010@ymail.co...	822	247	Windows Batch File	8/19/2014 10:0...	E2D33027
sg010@ymail.co...	1,382	273	Windows Batch File	8/19/2014 10:0...	42DD2E4C

Total 4 folders and 2,468 bytes in 3 files

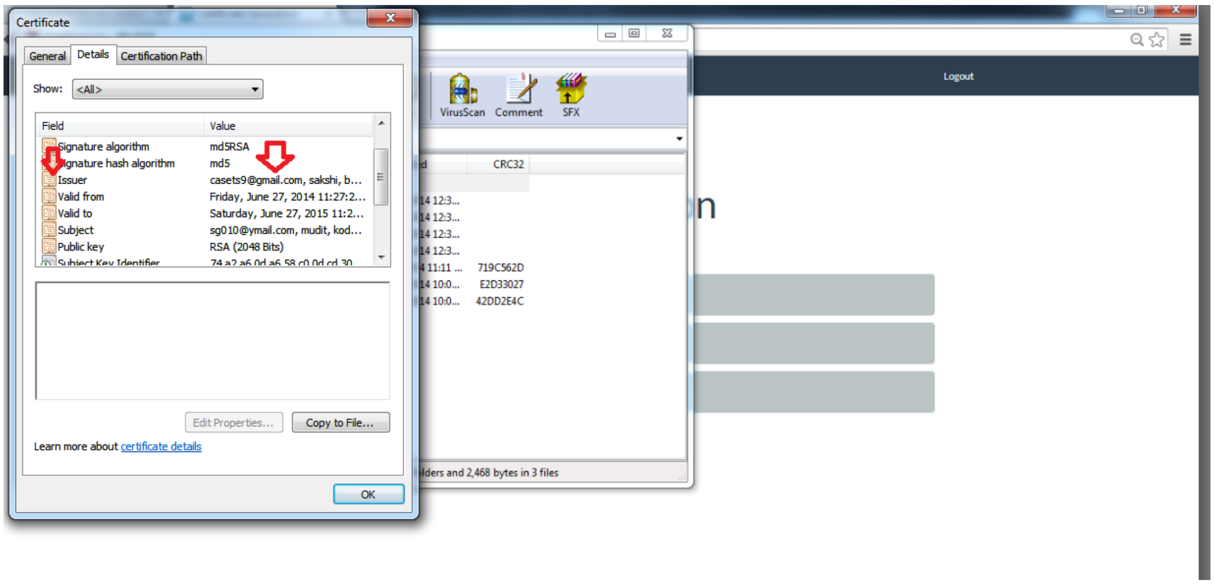
Figure 4.20



```
-----BEGIN CERTIFICATE REQUEST-----
MIICwzCCAasCAQAwfjELMAkGA1UEBhMCWk0xDDAKBgNVBAgMA25ldzENMAsGA1UE
BwwEeW9yazEOMAwGA1UECgwFcGFydWwxDjAMBgNVBAsMBXBhbnRzMQ0wCwYDVQQD
DARhYmhpMSMwIQYJKoZIhvcNAQkBFhRtdWRpdHNpbkBvdXRsb29rLmNvbTCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALRYjf8cxQjNd6A1CT2u1fmZVXeN
TNSa3dk2YffmeOvDJNN/lbRFBEAW1xVORC+P8cr99/zGwto81LANHX+zHUaGGtjr
1gnBmXVyNixIcqB8yqQ/d5VGxzra7MJVdtwM61zUVunODAXhy7t2o6Gcy9CrxbM301mMl4bV1yGik9hZ7U5JTcNYg
+lVcVS9D5MRPZ/THJmp68N56a34RTrC80xzPoN3F73wDtoGCzHsmyKBxMy1JTEj5lvwLK2uYhaSqCsYbOvntHu
+xHO5hlM0bDR4Ccxag4JZ6PWINn8r4hnL29Oh5Q85FWDw4++o4suZwqlBcnBGaxhWHsOr3SCNXvMCAwEA
AaAAMA0GCSqGSIb3DQEBBQUAA4IBAQBxVRJ5NkOY+G8iqbRT71SBUULcuCKpXcesigVSqzGYNlHPQVZ/wcWbi+Vd
+xy7SM7aeoT4bOU3LHi3/smeOiaogjBJZqvqk/qA
rwONJLxYT6AyiG6uaHsOnDnyNN6jsofzK7kOx6RaR46iYlkWeUU5utCCnMFTxYhHmP6PyYr
+etUJ/uJg0ZYgBuM4mEqjJqIu6taFdvCcmmgNLX4WVdCUJjiP9r7gHF80LcCdlyF5Vlxv
+oldQUFlPu8mFCr7oFto26Psh4UquHY2ko13lbZ4SVSdarSaGYbU
Fsg3GoK1/9MpTpwOHILTlMsL4TOMLgW3t0exHsi8RubtIOdt7DAb-----END CERTIFICATE REQUEST-----
```

Figure 4.21

After ca self-signed certificate generation and viewing the pending csr the certificate generated is mailed and is,



Show: <All>

Field	Value
Signature algorithm	md5RSA
Signature hash algorithm	md5
Issuer	casets9@gmail.com, sakshi, b...
Valid from	Friday, June 27, 2014 11:27:2...
Valid to	Saturday, June 27, 2015 11:2...
Subject	sg010@ymail.com, mudit, kod...
Public key	RSA (2048 Bits)
Subject Key Identifier	74 a2 a6 0d a6 58 c0 0d cd 30

Learn more about certificate details

```
14 12:3...
14 12:3...
14 12:3...
14 12:3...
14 11:11 ...   719C562D
14 10:0...     E2D33027
14 10:0...     42DD2E4C
```

...lders and 2,468 bytes in 3 files

Figure 4.22

<u>CHAPTER 5</u>-<u>Results & Discussions</u>

The x.509 certificate authority system was successfully developed. RSA is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. It is used to generate public and private keys for every user. Certificates are generated using the x.509 standard, using RSA keys, which is one of the primary web standards for certificates.

CHAPTER 6-Conclusion & Future Work

In today's world, where most major transactions and important information occurs over the internet or a network, certificate systems are crucial part of security. They are used in authentication of a sender, and in keeping data secure over transmissions. With increasing amount of black hats trying to intercept data for personal gain, encryption and authentication are a must.

In future, it is essential for us to devise more complicated algorithms than RSA to prevent the hackers from cracking the system as their techniques are also upgrading every day!

REFERENCES

Books Referred:

SL. NO.	Author Name	Book Name
1	William Stalling	Cryptography and Network Security
2	van Oorschot, Menezes, and Vanstone	Handbook of Applied Cryptography

Online Sources:

1. Wikipedia
2. RFCs
3. PHP Manual
4. W3Schools

APPENDIX 1

Here are a few openssl commands for your reference:

A.1 openssl_cse_new-

Resource

openssl_csr_new (array $dn , resource &$privkey [, array $configargs [, array $extraattribs]])

Parameter

dn-

The Distinguished Name to be used in the certificate.

Privkey-

privkey should be set to a private key that was previously generated by openssl_pkey_new()

Configargs-

By default, the information in your system openssl.conf is used to initialize the request;

Extraattribs-

extraattribs is used to specify additional configuration options for the CSR.

Return Values

Returns the CSR.

A.2 openssl_pkey_new-

Resource

- openssl_pkey_new ([array $configargs])
- generates a new private and public key pair. The public component of the key can be obtained usingopenssl_pkey_get_public().

Parameters

configargs

finetunes the key generation (such as specifying the number of bits) using configargs.

Return Values

Returns a resource identifier for the pkey on success, or FALSE on error.

A.3 openssl_pkey_get_public

Resource

openssl_get_publickey()

- extracts the public key from certificate and prepares it for use by other functions.

Parameters:

certificate

certificate can be one of the following:

- an X.509 certificate resource
- a string having the format file://path/to/file.pem. The named file must contain a PEM encoded certificate/public key (it may contain both).
- A PEM formatted public key.

Return Values

Returns a positive key resource identifier on success, or FALSE on error.

A.4 openssl_csr_sign

Resource

- Signs a CSR with another certificate (or itself) and generate a certificate.
- resource openssl_csr_sign (mixed $csr , mixed $cacert , mixed $priv_key , int $days [, array $configargs [, int$serial = 0]])
- openssl_csr_sign() generates an x509 certificate resource from the given CSR.

Parameters

csr

A CSR previously generated by openssl_csr_new(). It can also be the path to a PEM encoded CSR when specified as file://path/to/csror an exported string generated by openssl_csr_export().

cacert

The generated certificate will be signed by cacert. If cacert is NULL, the generated certificate will be a self-signed certificate.

priv_key

priv_key is the private key that corresponds to cacert.

days

days specifies the length of time for which the generated certificate will be valid, in days.

configargs

You can finetune the CSR signing by configargs. See openssl_csr_new() for more information about configargs.

serial

An optional the serial number of issued certificate. If not specified it will default to 0.

Return Values

Returns an x509 certificate resource on success, FALSE on failure.

A.5 openssl_csr_export

Resource

Exports a CSR as a string.

Description

bool openssl_csr_export (resource $csr , string &$out [, bool $notext = true])

openssl_csr_export() takes the Certificate Signing Request represented by csr and stores it as ascii-armoured text into out, which is passed by reference.

Parameters

Csr

A CSR previously generated by openssl_csr_new(). It can also be the path to a PEM encoded CSR when specified as file://path/to/csror an exported string generated by openssl_csr_export().

notext

The optional parameter notext affects the verbosity of the output; if it is FALSE, then additional human-readable information is included in the output. The default value of notext is TRUE.

Return Values

Returns TRUE on success or FALSE on failure.

A.5 openssl_x509_export

Resource

openssl_x509_export — Exports a certificate as a string

Description

bool openssl_x509_export (mixed $x509 , string &$output [, bool $notext = TRUE])

openssl_x509_export() stores x509 into a string named by output in a PEM encoded format.

Parameters

x509

See Key/Certificate parameters for a list of valid values.

output

On success, this will hold the PEM.

Notext

The optional parameter notext affects the verbosity of the output; if it is FALSE, then additional human-readable information is included in the output. The default value of notext is TRUE.

Return Values

Returns TRUE on success or FALSE on failure.

A.6 openssl_pkey_export

Resource

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_export — Gets an exportable representation of a key into a string

Description

bool openssl_pkey_export (mixed $key , string &$out [, string $passphrase [, array $configargs]])

openssl_pkey_export() exports key as a PEM encoded string and stores it into out (which is passed by reference).

Note: You need to have a valid openssl.cnf installed for this function to operate correctly. See the notes under the installation sectionfor more information.

Parameters

passphrase

The key is optionally protected by passphrase.

configargs

configargs can be used to fine-tune the export process by specifying and/or overriding options for the openssl configuration file.

Returns the Value:

TRUE on success

(or)

FALSE on failure