SSL Unleashed

By Chintan Gurjar : Lucideus Tech. Pvt.Ltd.

In this article, I am going to tell you everything about SSL: What it is, why we need it, its technical and non-technical aspects, etc. This article covers the introduction, SSL certificate, encryption, the process of encryption, and how your browser interacts with and trusts that certificate provided by the website you are visiting.

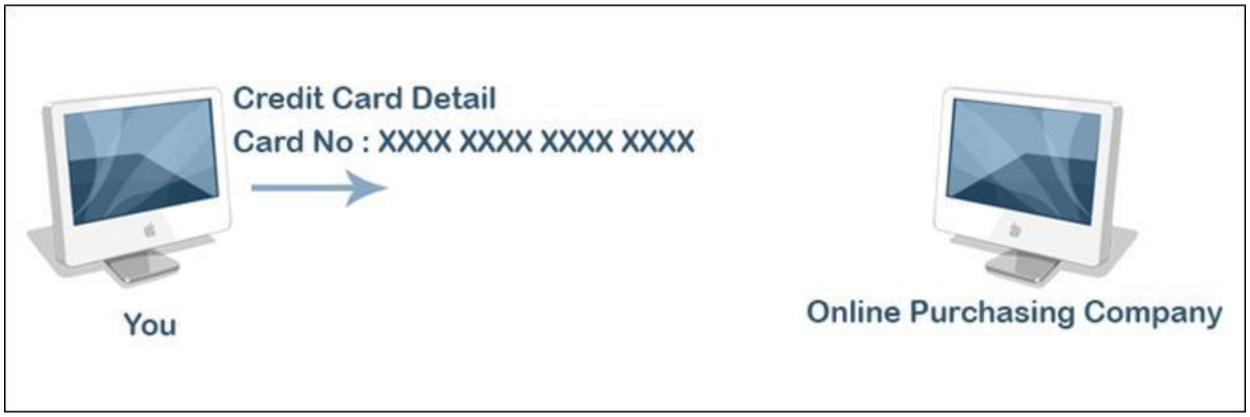
Existence of SSL

There are basically two aspects of SSL. One is **encryption** and the second is **identification**. **Encryption** is what you do to hide the content of the data sent from one machine to another machine. It is done by changing the content of the data so it looks like garbage that is human-readable but not human-understandable. It is exactly like speaking in a different language with which one person is not familiar. I am Indian; if someone speaks in the Russian language, it is not understandable by me, so to me the Russian language is like an encrypted language. However, if I get a translator and he/she translates that Russian language into Hindi then I can say that now it is understandable by me. So it is said that message has been decrypted.

Identification is related to trust. In the previous scenario, how can I trust the translator who is converting Russian language to Hindi? Is she/he legitimate? Can I trust him/her? In the digital world, it is something like this. Your machine has to trust the SSL certificate (security mechanism) provided by the website via an SSL certificate issuing vendor.

Encryption Explained

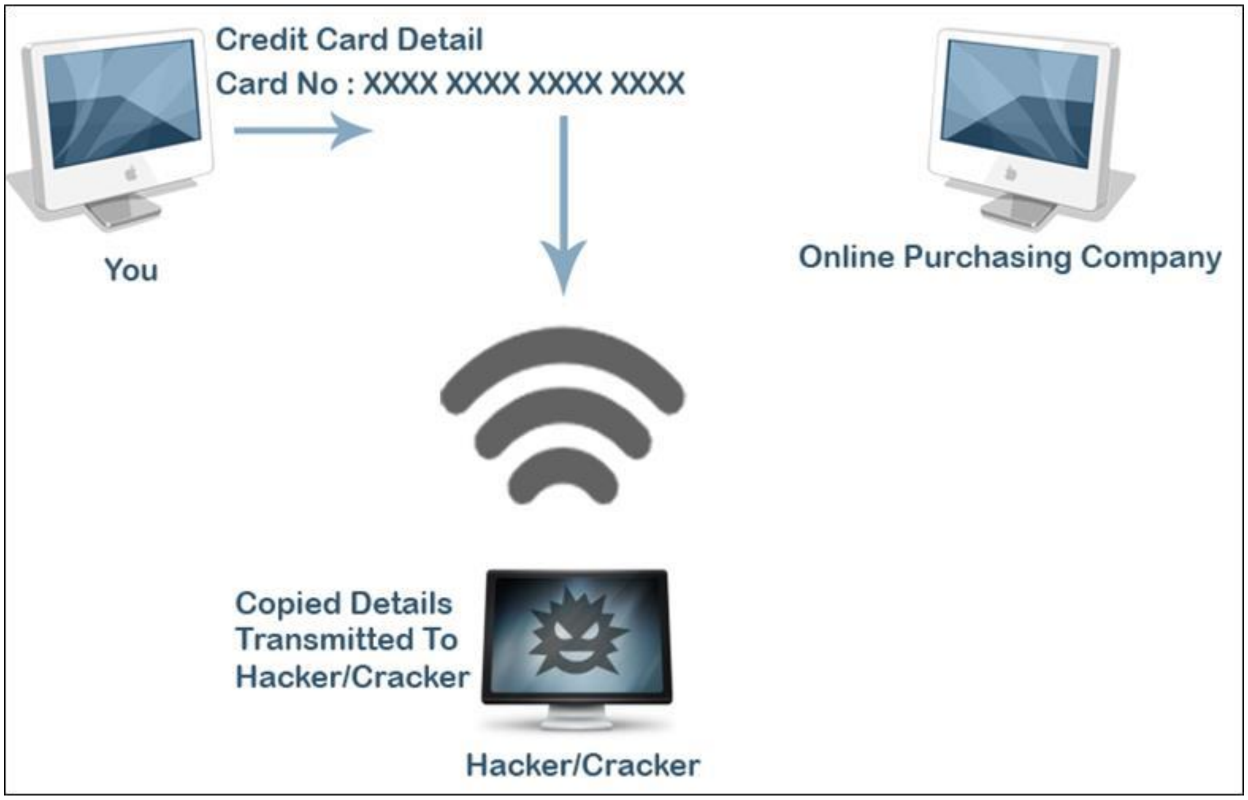
To understand the scenario, let's take an example directly. Let's suppose you are sending credit card details to the company (any company/online purchasing website, etc.)



So here is the scenario: You are on the left and you will be sending your card details to the other machine. Now there can be two scenarios:

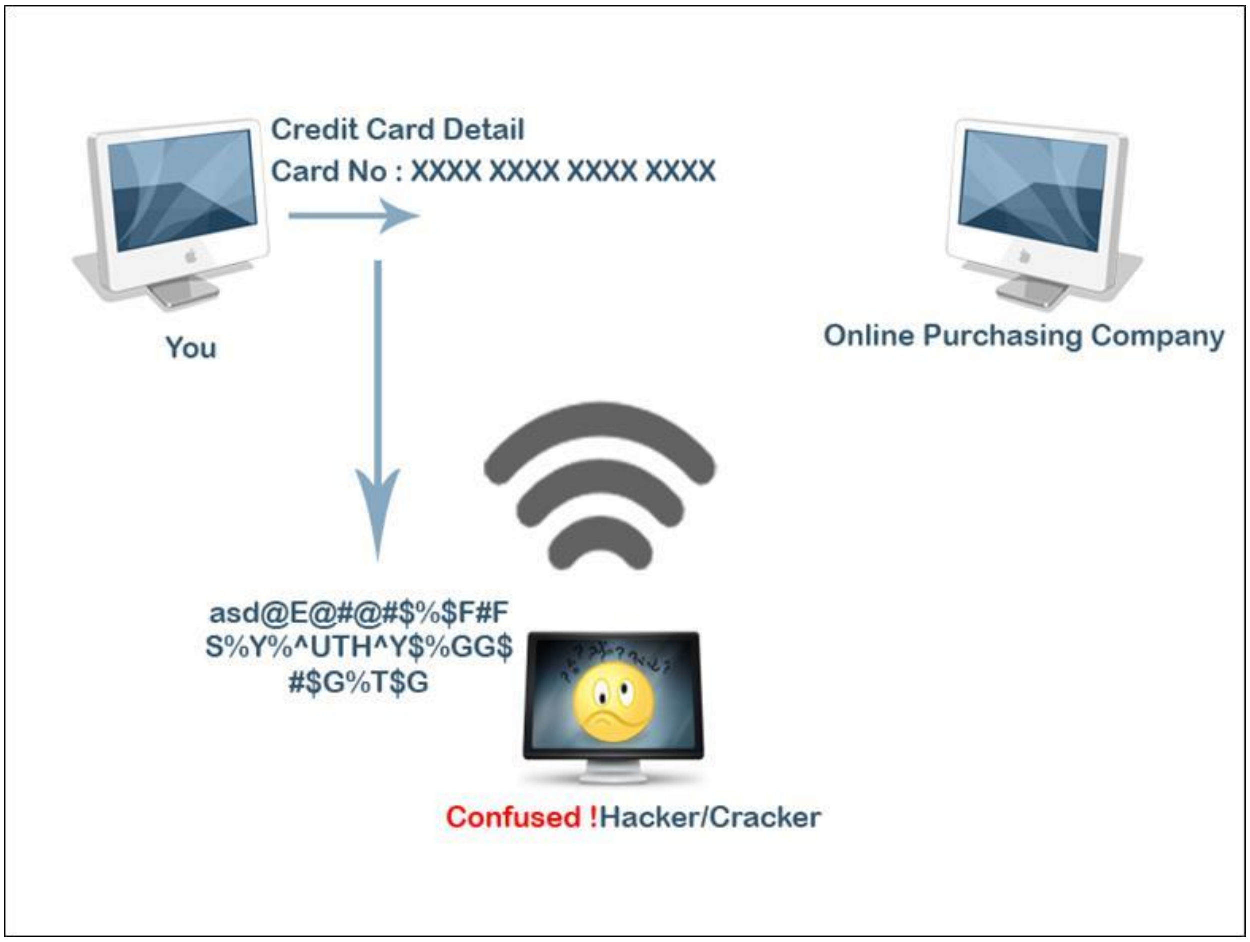
1. Without SSL
2. With SSL

Without SSL: In this scenario there can be another machine in your network that can grab the details sent by another machine, as shown in the figure below.


Credit Card Detail
Card No : XXXX XXXX XXXX XXXX
You
Online Purchasing Company
Copied Details Transmitted To Hacker/Cracker
Hacker/Cracker

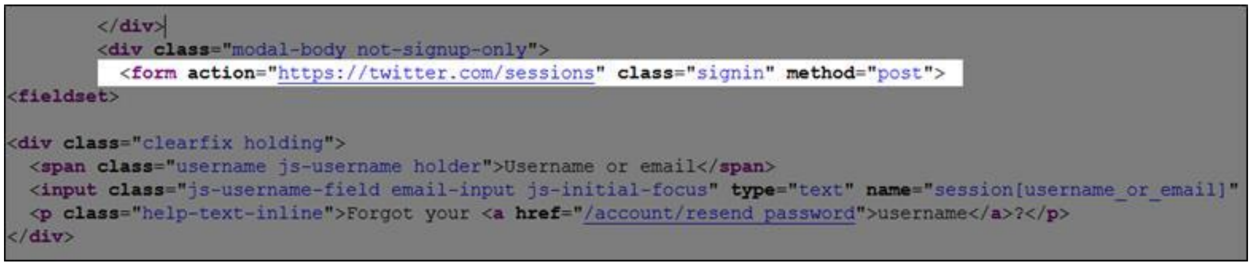
As you know, this scenario is without SSL. In this case, any malicious user lying in the same network can perform an MITM attack or any other attack that contains simple network traffic monitoring and can grab your credit card number or any other personal details. So it is always necessary to use SSL to act as a barrier. It creates a tunneling technique.

With SSL: SSL puts the security mechanism on the network layer before you transfer the data. As the picture below shows, it creates a barrier or tunnel through which the user can transfer any data to the other network. This time the malicious user (lying on the same network) will see the tunnel, so he won't be able to grab your private data as it passes through the tunnel.



As you can in the picture, a malicious user grabs the data passing through the tunnel, but she/he will get encrypted data, not the real plain text data. So the data can be grabbed but now it has only garbage value for the cracker/hacker, as she/he will never come to know that what exactly the real data was. In order to decrypt the data, the hacker will need an encryption key, which she/he will never get.

Let’s see what HTTPS coding looks like. Here I will give you Twitter ‘s sign-in page source code. As you all know, every sign-in page uses the POST method to pass our data to the server. Every POST method is defined under the form. A form action method is shown in the picture below.



```
        </div>
        <div class="modal-body not-signup-only">
          <form action="https://twitter.com/sessions" class="signin" method="post">
<fieldset>

<div class="clearfix holding">
  <span class="username js-username holder">Username or email</span>
  <input class="js-username-field email-input js-initial-focus" type="text" name="session[username_or_email]"
  <p class="help-text-inline">Forgot your <a href="/account/resend_password">username</a>?</p>
</div>
```

You can see here that the action in the HTML code is embedded with HTTPS, which is a secure HTTP method. It confirms that your log-in credentials are secure once you click on the submit button in order to log in. The actual process of Encryption is enclosed below.

The ‘Actual’ Process of Encryption

Everything we are doing is a digital process that happens in no matter of time but, in reality, it’s really a long process that has many steps. At each step, integrity and authenticity are involved.

As soon as you hit the button to log in, **First** it says that now your machine (computer) is ready for the encryption process.**Second,** the server will send one certificate to your machine, which has to be digitally signed by you. If you don’t go through the process of logging in, the server will not send your certificate as it’s not actually needed, because you are not going to give any credentials to the server. **Third,** your machine digitally signs the certificate in the backend and sends it to the server. Indirectly it tells the server that, *“I am ready now, kindly start the encryption process, please.“*

Fourth, the server receives the certificate signed by your machine and starts the encryption process at the client’s end. **Last,**the server actually encrypts all the messages passed by the client and, after successful encryption, all the data is passed through the secure tunnel at the server end.

This whole process is called a "handshake."

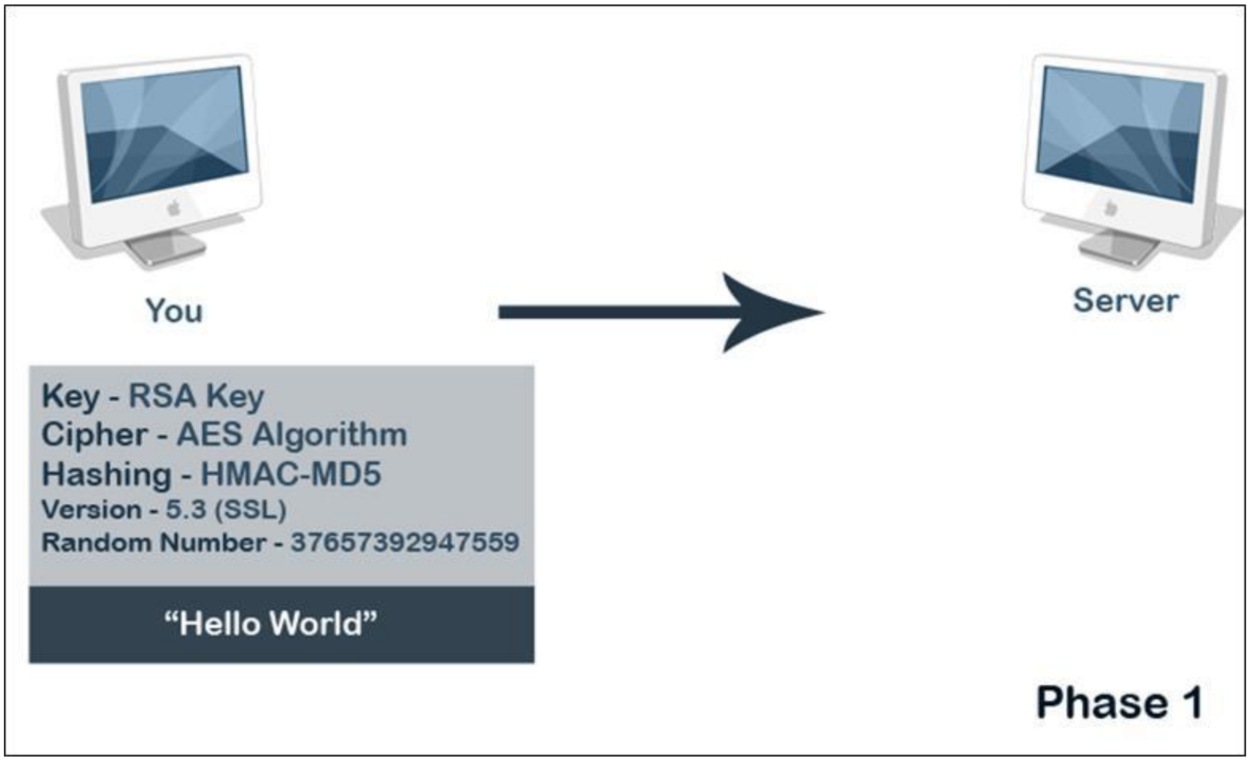
The 'Actual' Process of Encryption Unleashed in Detail

1. **How does your machine choose an encryption process? What are the different possibilities/methods involved in this process?**

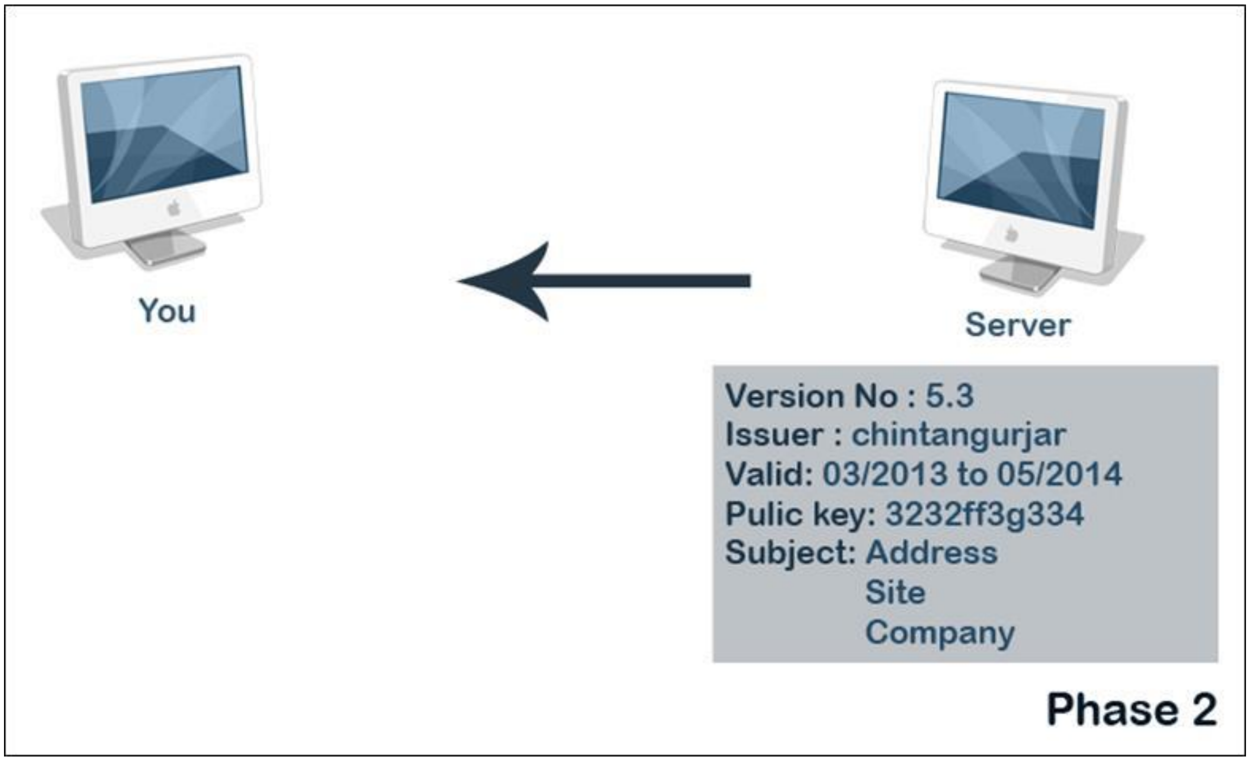
   Encryption process basically includes three parts.

   1. **Encrypting Message:** There are many algorithms for encrypting messages. The most famous are **AES**, **3-DES,**and **RC4**. People generally use one of these methods to encrypt the message. Each algorithm contains many operations, such as shiftrows, subbytes, mixcolumns, addroundkey etc.. That is the only strength of these algorithms. If the algorithm has more operations, it provides better encryption. It is the way of encrypting data between client and the server.
   2. **Hashing—**This is known as a message authentication code. This term is used in cryptography. It includes a hash function combined with a secret key and it is used for authentication as well as data integrity. The strength of the hashing depends on the cryptographic function used in that along with the key. Generally**HMAC-MD5** and **HMAC-SHA1** are used for the hashing.
   3. **Choosing a Key for Encryption—**It contains a key exchange mechanism such as **RSA, DSA,** or **Diffie-Hellman algorithm**. One can get more information on this part by searching "TLS and public key cryptography" on Wikipedia.

Scenario: Let's suppose the computer on the left side is about to send the "Hello World" message to the server and it chooses the AEC cipher method, HMAC-MD5 hashing technique, and RSA key. Along with all these three data, it also sends the version number of SSL used by your machine for TLS (transport layer security) and random number. A random number is nothing but the master calculation that is used to do all other calculation for the encryption process. All this information is passed to the server.



In the next stage, the server sends a certificate to the client. This certificate contains information such as version number, serial number, signature, issuer, validity, subject, IssuerUniqueIdentifier, SubjectUniqueIdentifier, Signature algorithm, signature value, public key, etc.



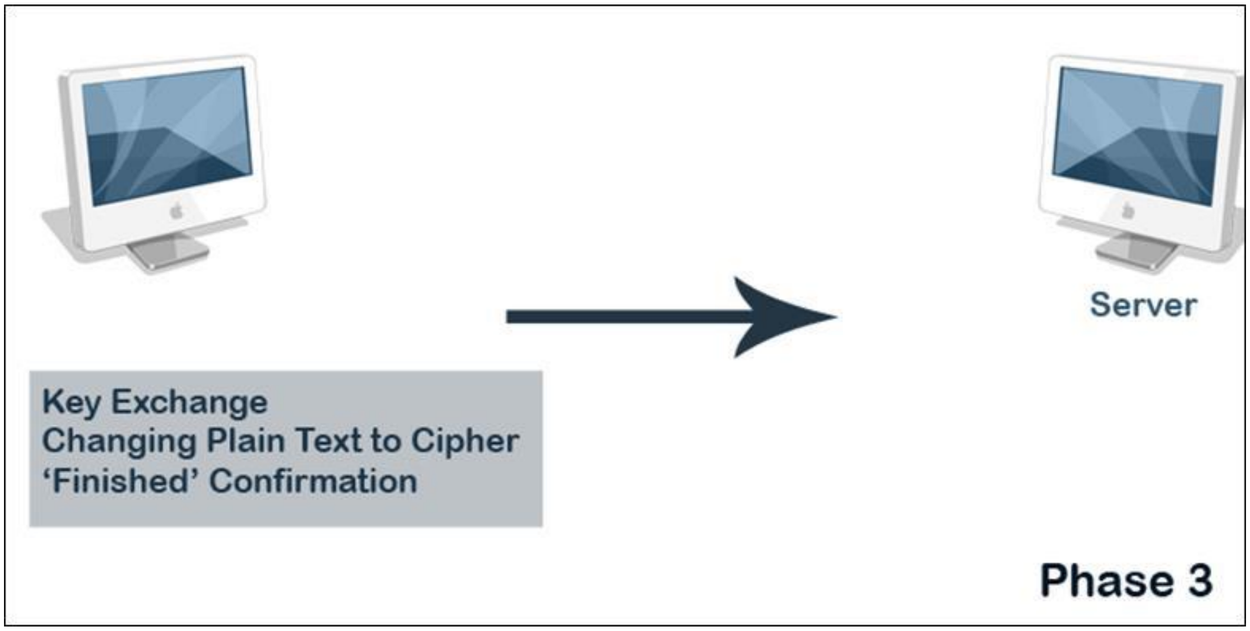
In phase 3, your computer tells the server to start encryption. Here both machines have to take part in the conversation. There are three steps to accomplish this process.

1. Key Exchange—Here both the machines calculate the master secret code that will be used for the encryption process.

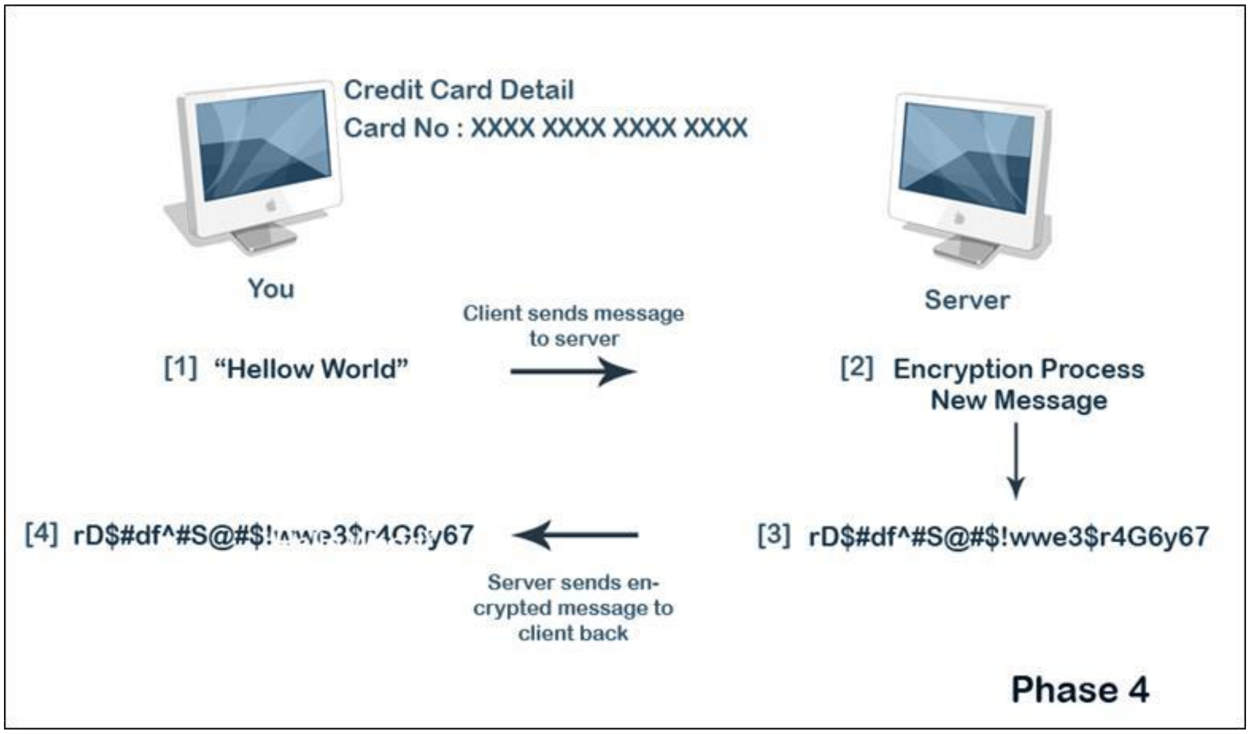
2. Change Original Text—Your computer tells the server to change the original text written in order to send the server. In our case, it's "Hello World." So the server actually receives confirmation of starting an encryption process and it starts the encryption process in order to generate the cipher text for our given text.

3. Finish State—In the last stage of the process, your computer tells the server that all the messages have been encrypted now and ready to go.

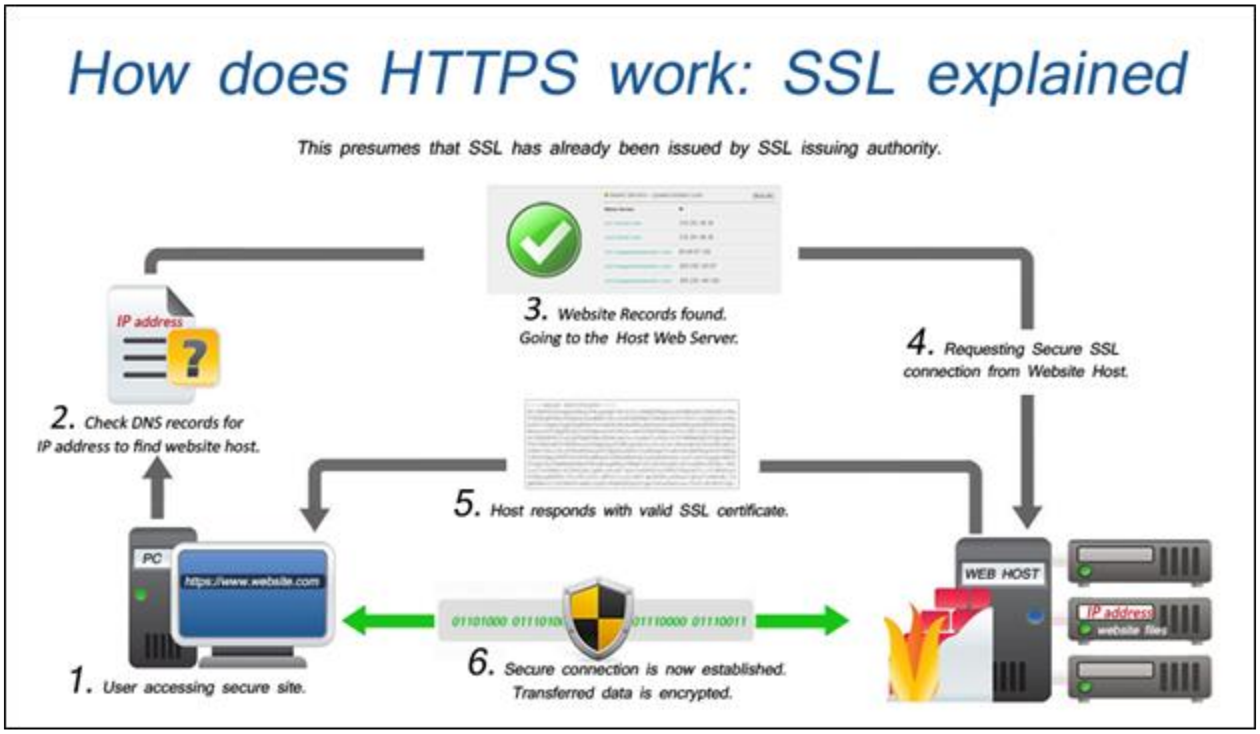
After all these three states occur, the message is sent to the server.



In phase 4, the server receives the messages from the client side and it encrypts those messages to make it cipher text. Then the server tells the client that it has finished encrypting the message and a message is sent back to the client is in encrypted form. Both of these scenarios are shown together in the pic below.


Credit Card Detail
Card No : XXXX XXXX XXXX XXXX
You
Server
Client sends message to server
[1] "Hellow World"
[2] Encryption Process New Message
[3] rD$#df^#S@#$!wwe3$r4G6y67
[4] rD$#df^#S@#$!wwe3$r4G6y67
Server sends encrypted message to client back
Phase 4

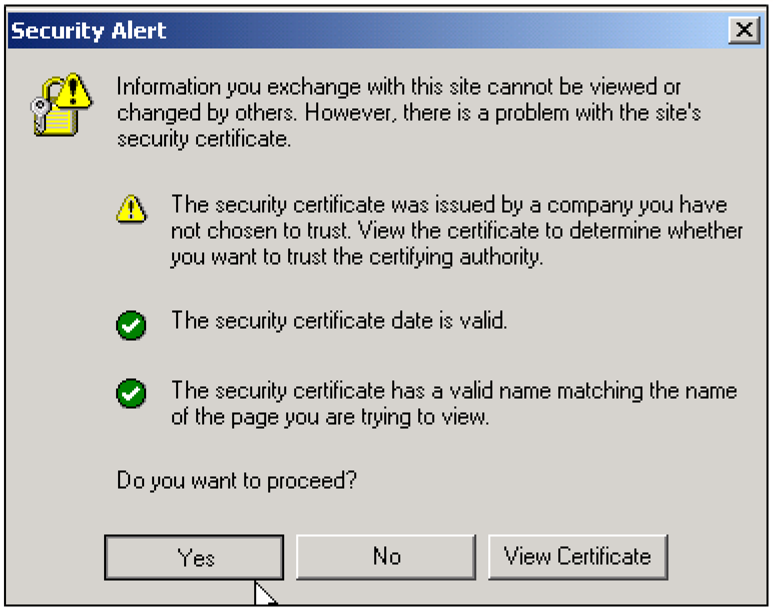
So this is the scenario of SSL. If we take the-real life scenario, then SSL works something like this:


How does HTTPS work: SSL explained
This presumes that SSL has already been issued by SSL issuing authority.
IP address
2. Check DNS records for IP address to find website host.
3. Website Records found. Going to the Host Web Server.
4. Requesting Secure SSL connection from Website Host.
5. Host responds with valid SSL certificate.
PC
https://www.website.com
1. User accessing secure site.
01101000 01110100
01110000 01110011
6. Secure connection is now established. Transferred data is encrypted.
WEB HOST
IP address
website files

SSL Identification

Another important thing has to be considered while using SSL. It is all about trust. You have to make sure that the website with which you are corresponding is the real website that you assume it is and that you trust. It doesn't mean that, if you have an SSL certificate, then you are secure. Having an SSL certificate only is not enough, So that is why identification comes into play. It's all about who you actually trust.

You might have seen the pic below when you try to access a website on the Internet. It represents SSL. There is a whole big process involved behind this dialogue box. The process behind this is something like this:



Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

The security certificate date is valid.

The security certificate has a valid name matching the name of the page you are trying to view.

Do you want to proceed?

1. First the website that you are going to access through SSL has to go to an organization that provides certificate authority. That organization will look into the details of the company/website and it will verify that the information is true about that company.

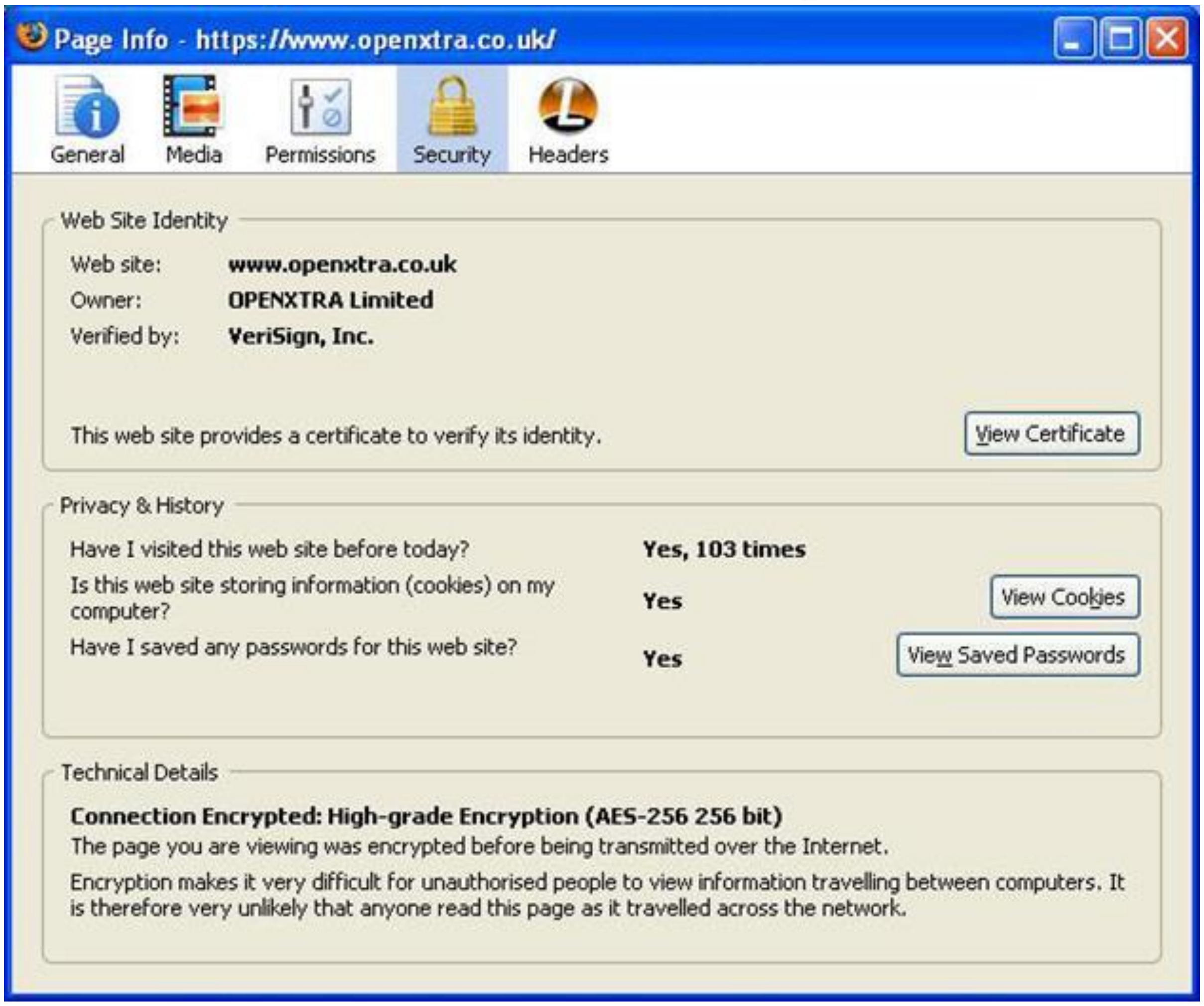
   To get the certificate, the company has to give certain information about itself. That information may include:

   1. The web server name
   2. Company name
   3. Location of company

   Then the authority company checks whether this information is correct or incorrect. If it is correct, it goes to step 2.

2. That company then creates one certificate and digitally signs it with cryptographic methods explained previously in this article.

   One certificate looks like this (there are different certificates):



Web Site Identity

Web site:	www.openxtra.co.uk
Owner:	OPENXTRA Limited
Verified by:	VeriSign, Inc.

This web site provides a certificate to verify its identity.

Privacy & History

Have I visited this web site before today?	Yes, 103 times
Is this web site storing information (cookies) on my computer?	Yes
Have I saved any passwords for this web site?	Yes

Technical Details

Connection Encrypted: High-grade Encryption (AES-256 256 bit)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it very difficult for unauthorised people to view information travelling between computers. It is therefore very unlikely that anyone read this page as it travelled across the network.

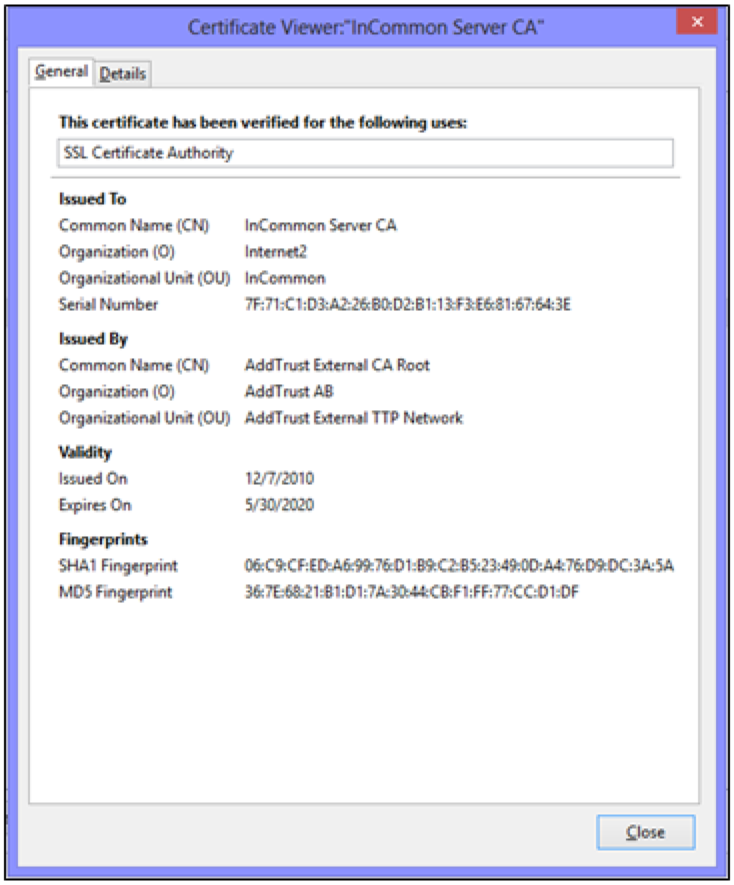
The certificate contains this information:

- Signature
- Signature algorithm
- Version
- Serial number
- Algorithm ID
- Issuer
- Identifier for issuer
- Identifier for company
- Public key Information
  - Key Algorithm
- Validity

- Company details

Now the signature is generated. A certificate contains all this information and the numbers used in this information are taken by the hashing algorithm, which does the calculations and generates signature. So the only number field that contains all this information is taken by the hashing algorithm in order to generate the signature. Then that number gets encrypted with the private key, so anyone who is holding a public key can verify that number.

1. Then they send the certificate to the company that you are going to access. The certificate generated in step 2 is given back to the company that asked for that certificate. Let’s suppose it runs an IIS/APACHE/TOMCAT web server. It installs that certificate on the web server. One can configure any of these web servers to use that certificate.
2. On the other hand, when you are trying to access the website, the certificate comes from the website at your end. Your browser will verify the details of the certificate and check that the information is true, then your browser will also sign that certificate. Make sure your browser won’t sign the incorrect certificate. At the client side, the certificate looks as shown below. This will be used in the handshake.



Certificate Viewer:"InCommon Server CA"

This certificate has been verified for the following uses:

SSL Certificate Authority

Issued To	
Common Name (CN)	InCommon Server CA
Organization (O)	Internet2
Organizational Unit (OU)	InCommon
Serial Number	7F:71:C1:D3:A2:26:B0:D2:B1:13:F3:E6:81:67:64:3E
Issued By	
Common Name (CN)	AddTrust External CA Root
Organization (O)	AddTrust AB
Organizational Unit (OU)	AddTrust External TTP Network
Validity	
Issued On	12/7/2010
Expires On	5/30/2020
Fingerprints	
SHA1 Fingerprint	06:C9:CF:ED:A6:99:76:D1:B9:C2:B5:23:49:0D:A4:76:D9:DC:3A:5A
MD5 Fingerprint	36:7E:68:21:B1:D1:7A:30:44:CB:F1:FF:77:CC:D1:DF

3. When your browser gets certificate, it verifies it via the signature and then it encrypts the data.

So I have simplified and unleashed SSL in front of you. Now I will give you a practical demonstration of how you can generate your own SSL certificate and how to use it.

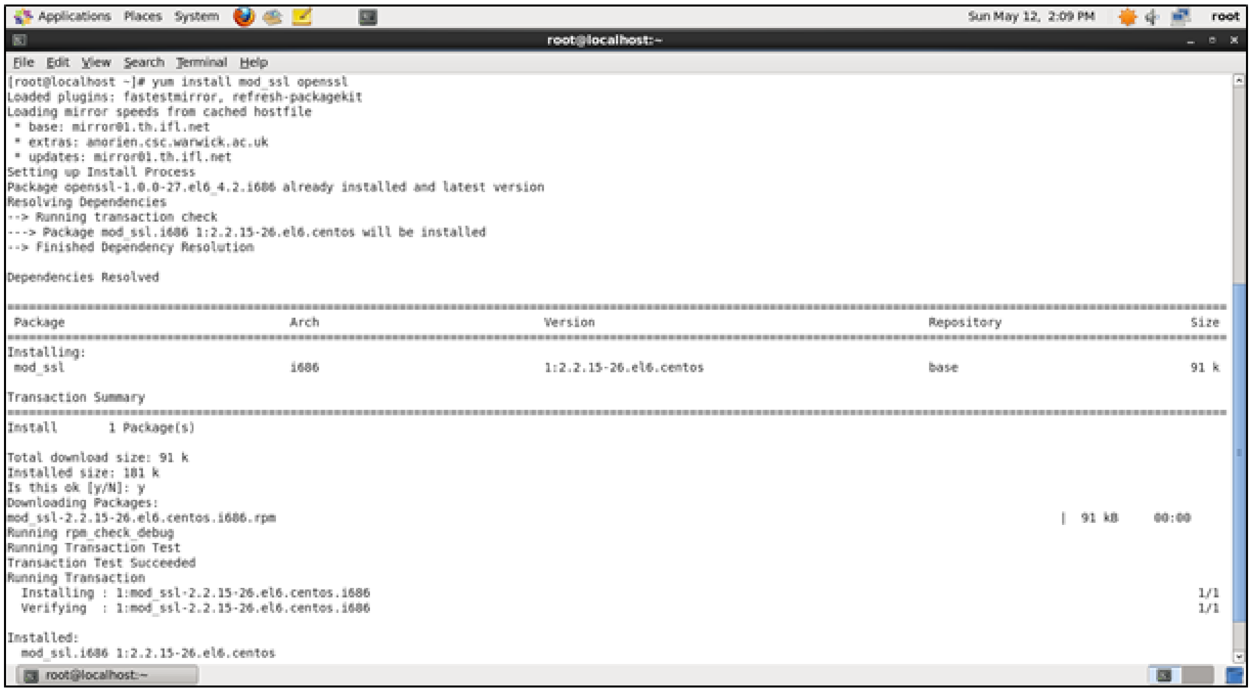
How to Generate SSL Certificate

We will download and install the required package to install an SSL certificate. Then we will sign the certificate manually by giving a personal information to the certificate. After this has been completed, we will restart the HTTPD service in order to run our SSL certificate successfully.

Step 1: **Downloading and Installing Required Software**

To generate an SSL certificate, OpenSSL and mod_ssl must be installed on the system. This tool may have already been installed in Apache, however this varies from system to system. So we will install these tools by using the following command:

yum install mod_ssl openssl



```
[root@localhost ~]# yum install mod_ssl openssl
Loaded plugins: fastestmirror, refresh-packagekit
Loading mirror speeds from cached hostfile
 * base: mirror01.th.ifl.net
 * extras: anorien.csc.warwick.ac.uk
 * updates: mirror01.th.ifl.net
Setting up Install Process
Package openssl-1.0.0-27.el6_4.2.i686 already installed and latest version
Resolving Dependencies
--> Running transaction check
---> Package mod_ssl.i686 1:2.2.15-26.el6.centos will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================================
 Package                     Arch                     Version                                   Repository                 Size
================================================================================================================================
Installing:
 mod_ssl                     i686                     1:2.2.15-26.el6.centos                    base                       91 k

Transaction Summary
================================================================================================================================
Install       1 Package(s)

Total download size: 91 k
Installed size: 181 k
Is this ok [y/N]: y
Downloading Packages:
mod_ssl-2.2.15-26.el6.centos.i686.rpm                                                                    |  91 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : 1:mod_ssl-2.2.15-26.el6.centos.i686                                                                          1/1
  Verifying  : 1:mod_ssl-2.2.15-26.el6.centos.i686                                                                          1/1

Installed:
  mod_ssl.i686 1:2.2.15-26.el6.centos
```

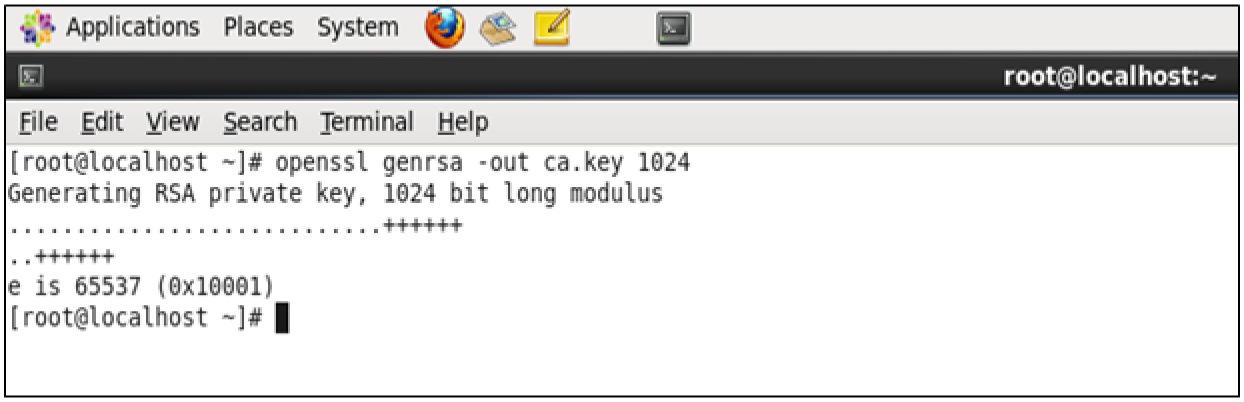
Figure 1 : mod_ssl Installation process

Step 2: Sign Our SSL Certificate Manually

We will now use OpenSSL to sign our certificate. It is important to have this SSL certificate from a valid and trusted certificate providing authority if the use is for any company or organization. However, if it's for personal use then one can do it at home and there will be no need for a trusted company to provide a signed certificate.

We will have to follow the steps below to create a self-signed certificate:

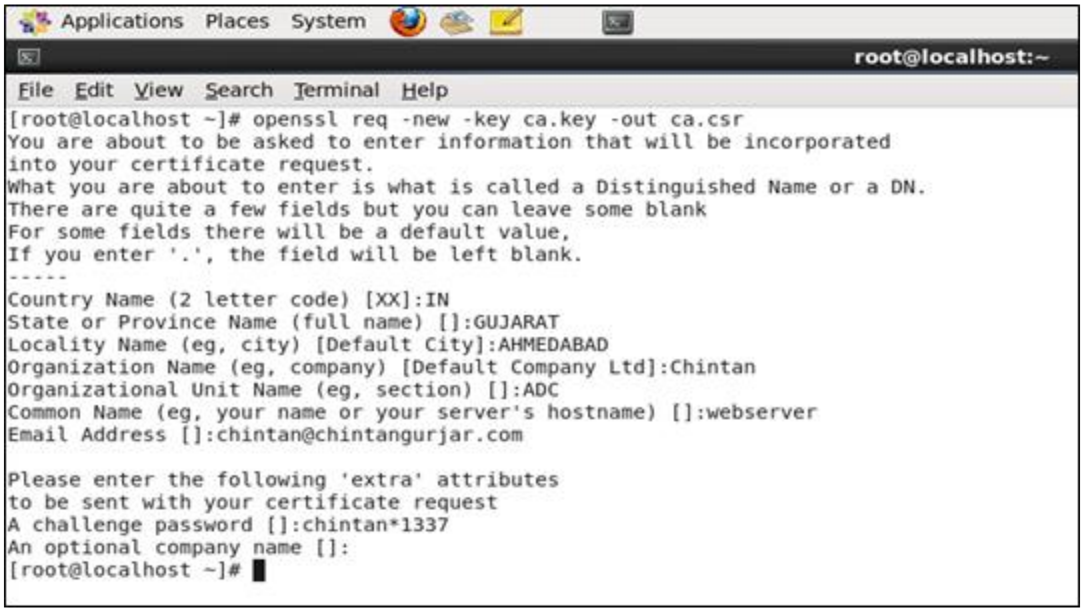
- **Create A Private Key**


Applications Places System
root@localhost:~
File Edit View Search Terminal Help

```
[root@localhost ~]# openssl genrsa -out ca.key 1024
Generating RSA private key, 1024 bit long modulus
............................++++++
..++++++
e is 65537 (0x10001)
[root@localhost ~]# 
```

Figure 2 : Generate Private Key

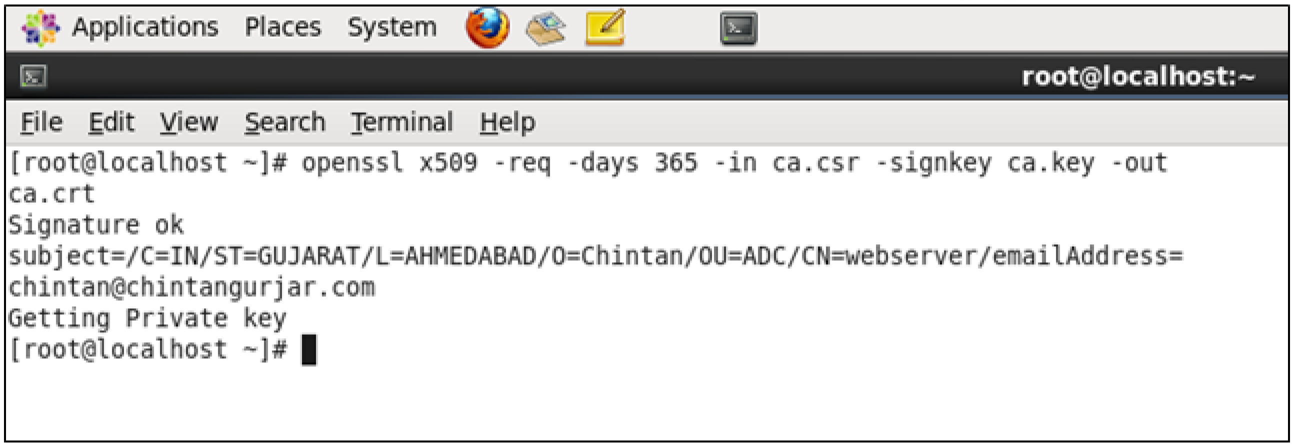
- **CSR Generation**


Applications Places System
root@localhost:~
File Edit View Search Terminal Help

```
[root@localhost ~]# openssl req -new -key ca.key -out ca.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:GUJARAT
Locality Name (eg, city) [Default City]:AHMEDABAD
Organization Name (eg, company) [Default Company Ltd]:Chintan
Organizational Unit Name (eg, section) []:ADC
Common Name (eg, your name or your server's hostname) []:webserver
Email Address []:chintan@chintangurjar.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:chintan*1337
An optional company name []:
[root@localhost ~]# 
```

Figure 3 : Certificate Generation

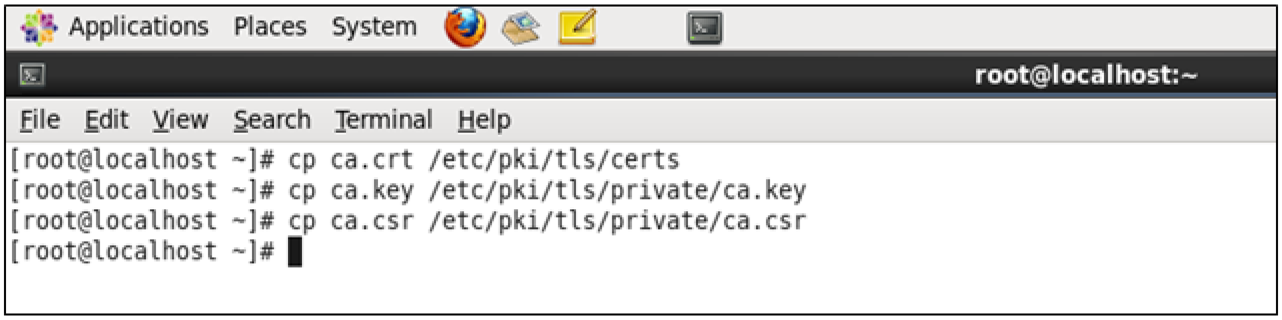
- **Key Generation (Self-Signed)**


Applications Places System
root@localhost:~
File Edit View Search Terminal Help

```
[root@localhost ~]# openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out
ca.crt
Signature ok
subject=/C=IN/ST=GUJARAT/L=AHMEDABAD/O=Chintan/OU=ADC/CN=webserver/emailAddress=
chintan@chintangurjar.com
Getting Private key
[root@localhost ~]#
```

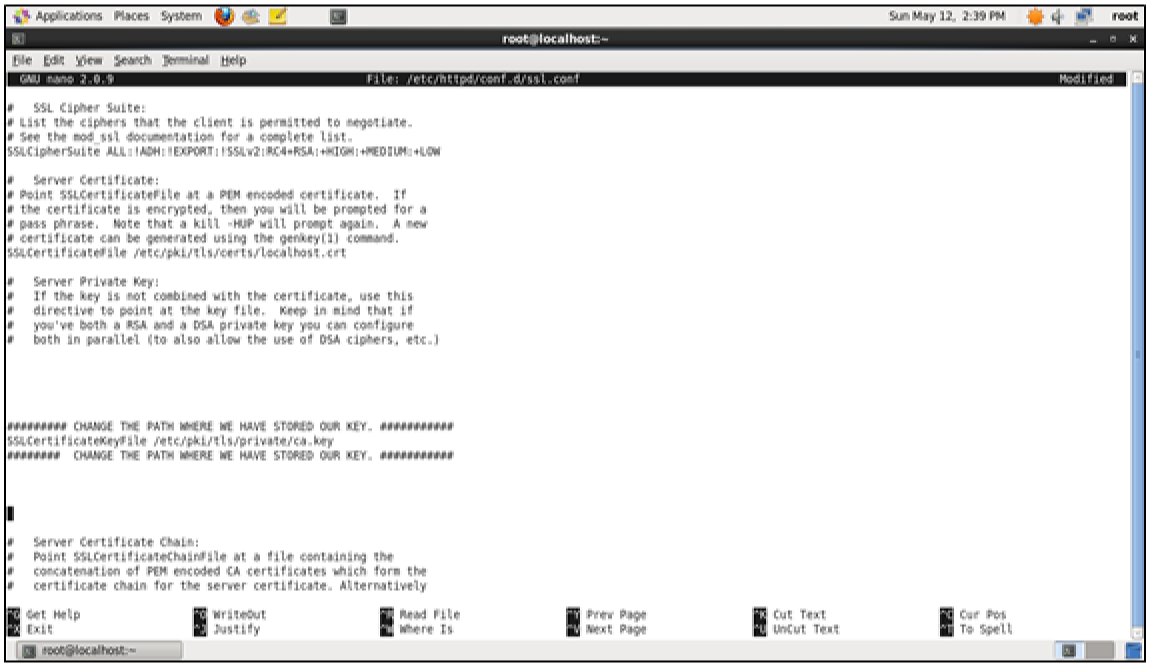
Figure 4 : Key Generation

- **Put Files In Proper Location**


Applications Places System
root@localhost:~
File Edit View Search Terminal Help

```
[root@localhost ~]# cp ca.crt /etc/pki/tls/certs
[root@localhost ~]# cp ca.key /etc/pki/tls/private/ca.key
[root@localhost ~]# cp ca.csr /etc/pki/tls/private/ca.csr
[root@localhost ~]#
```

Now that we have configured our task and put files into the right folder, we will update the Apache server by updating the SSL configuration file, which is **ssl.conf**. We will use the nano editor to update our ssl.conf file and **change the path of the key file,** where we have stored our files, such as the certificate and key.


Applications Places System
Sun May 12, 2:39 PM
root
root@localhost:~
File Edit View Search Terminal Help
root@localhost:~

```
  GNU nano 2.0.9                File: /etc/httpd/conf.d/ssl.conf                Modified

#   SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW

#   Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/pki/tls/certs/localhost.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)




######### CHANGE THE PATH WHERE WE HAVE STORED OUR KEY. ###########
SSLCertificateKeyFile /etc/pki/tls/private/ca.key
########  CHANGE THE PATH WHERE WE HAVE STORED OUR KEY. ###########




#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively

^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

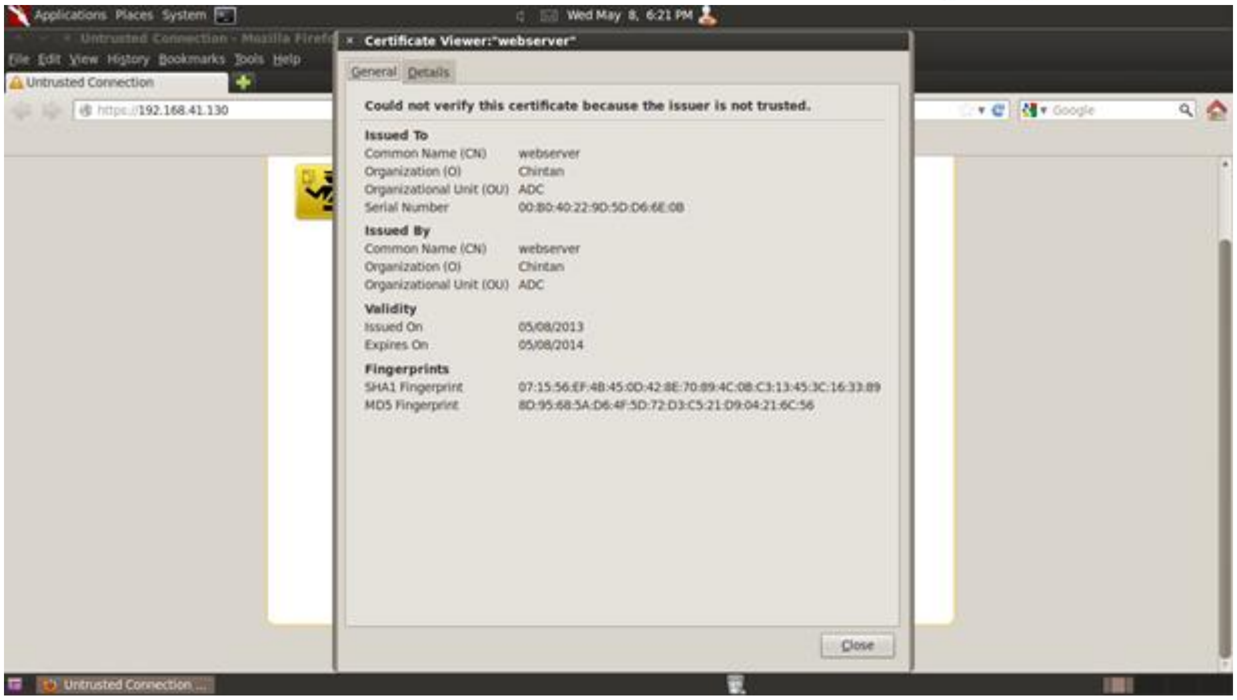
Figure 5 : Changing key path in ssl.conf file to make it work properly.

The next step is to save and exit this file, after which we will start HTTPD service by typing the following command in the terminal:

```
# /etc/init.d/httpd restart
```

Now that we have set up our **secure socket layer**, if we attempt to log in to our the server machine through the client machine or attacking machine via the **https** protocol, it will ask for the authentication. Before the authentication, a dialogue box is shown that warns the “connection is not trusted” and asks if they want to see the certificate that is going to be downloaded to the machine or if they want to stay in offline mode.

Once the user clicks on the view certificate, it will display the certificate that was created earlier. This screenshot below illustrates this step:


Applications Places System
Wed May 8, 6:21 PM
Untrusted Connection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Untrusted Connection
https://192.168.41.130
Google
Certificate Viewer:"webserver"
General Details
Close
Untrusted Connection ...

Could not verify this certificate because the issuer is not trusted.

Issued To

Common Name (CN)	webserver
Organization (O)	Chintan
Organizational Unit (OU)	ADC
Serial Number	00:B0:40:22:9D:5D:D6:6E:0B

Issued By

Common Name (CN)	webserver
Organization (O)	Chintan
Organizational Unit (OU)	ADC

Validity

Issued On	05/08/2013
Expires On	05/08/2014

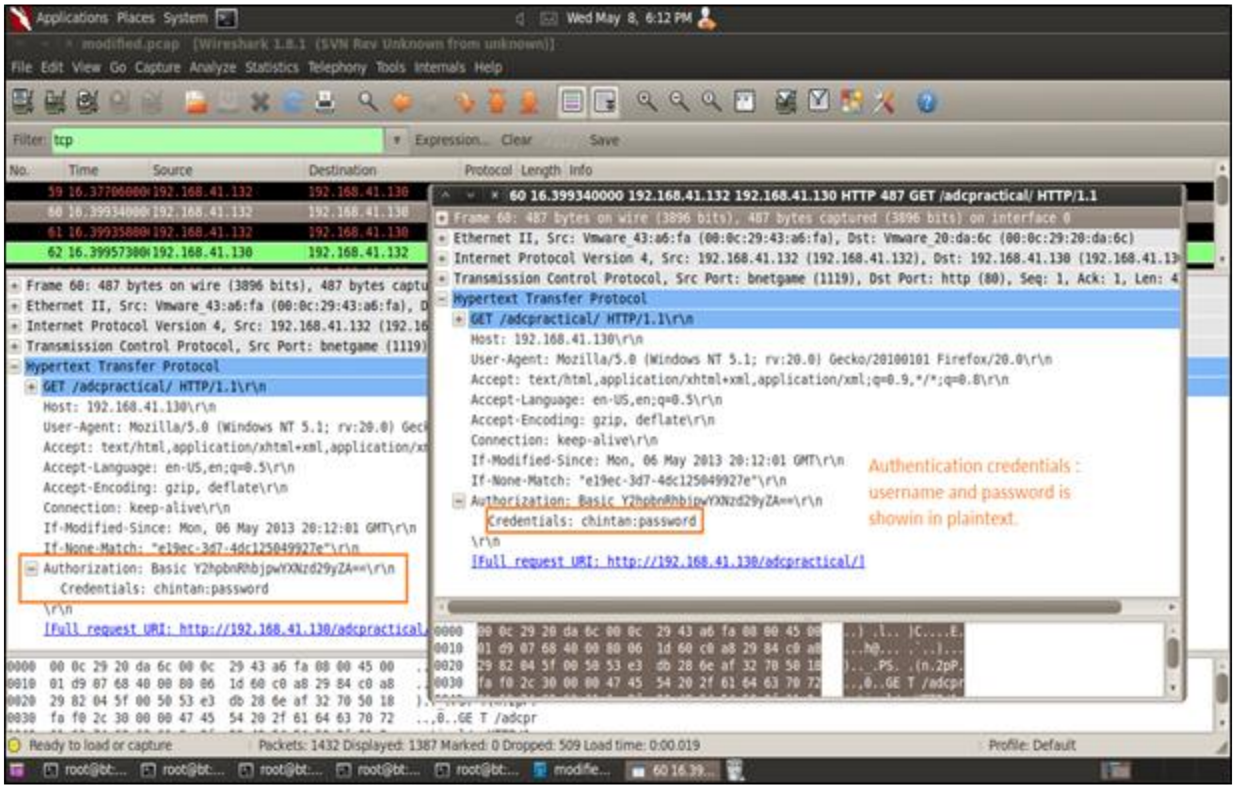
Fingerprints

SHA1 Fingerprint	07:15:56:EF:4B:45:0D:42:8E:70:89:4C:0B:C3:13:45:3C:16:33:89
MD5 Fingerprint	8D:95:68:5A:D6:4F:5D:72:D3:C5:21:D9:04:21:6C:56

Figure 6 : Showing our self generated certificate digitally signed by us

It shows the issuer's name, organization name, and validity of the certificate. With this secure connection, if one logs in into the website or network, it will encrypt the plain text data and an attacker won't be able to capture it straight away with theWireshark option.

Experiment: I have set up the Apache server and I have performed an MITM attack in which I pass log-in credentials without SSL and with SSL. For both ways I captured Wireshark screenshots. Those screenshots are as follows:


Applications Places System
Wed May 8, 6:12 PM
modified.pcap [Wireshark 1.8.1 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp
Expression... Clear Save

```
No.  Time            Source          Destination     Protocol Length Info
 59 16.377060000 192.168.41.132  192.168.41.130
 60 16.399340000 192.168.41.132  192.168.41.130
 61 16.399358000 192.168.41.132  192.168.41.130
 62 16.399573000 192.168.41.130  192.168.41.132
```

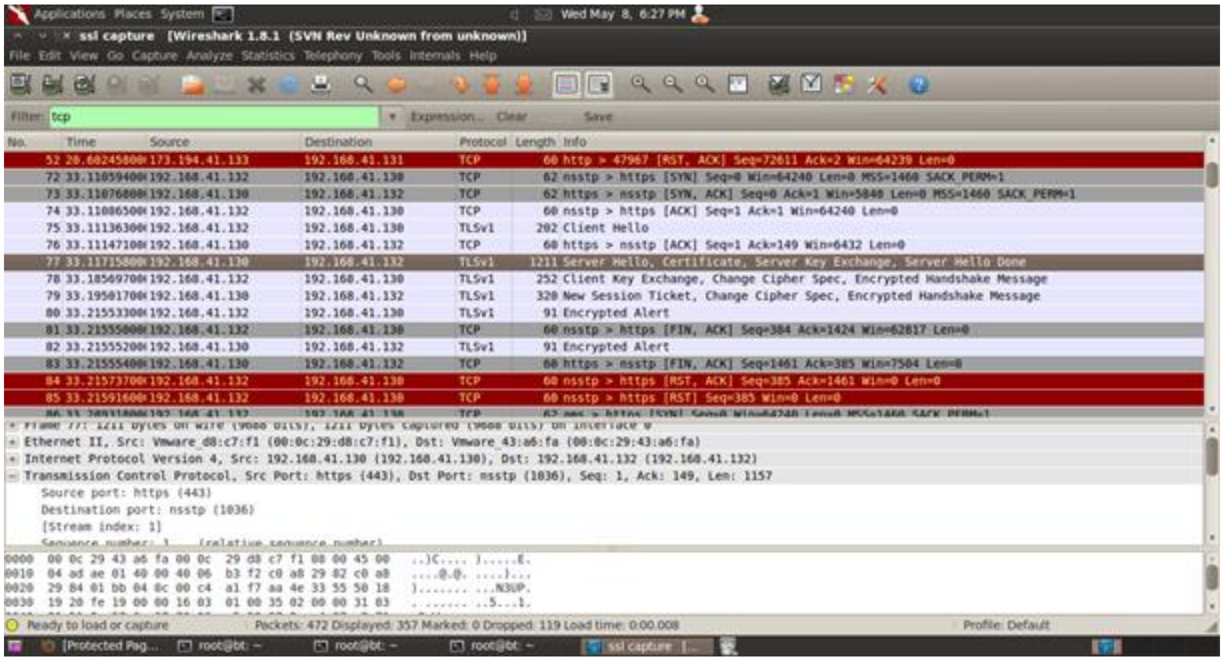
```
60 16.399340000 192.168.41.132 192.168.41.130 HTTP 487 GET /adcpractical/ HTTP/1.1
+ Frame 60: 487 bytes on wire (3896 bits), 487 bytes captured (3896 bits) on interface 0
+ Ethernet II, Src: Vmware_43:a6:fa (00:0c:29:43:a6:fa), Dst: Vmware_20:da:6c (00:0c:29:20:da:6c)
+ Internet Protocol Version 4, Src: 192.168.41.132 (192.168.41.132), Dst: 192.168.41.130 (192.168.41.13
+ Transmission Control Protocol, Src Port: bnetgame (1119), Dst Port: http (80), Seq: 1, Ack: 1, Len: 4
- Hypertext Transfer Protocol
  + GET /adcpractical/ HTTP/1.1\r\n
    Host: 192.168.41.130\r\n
    User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:20.0) Gecko/20100101 Firefox/20.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    If-Modified-Since: Mon, 06 May 2013 20:12:01 GMT\r\n
    If-None-Match: "e19ec-3d7-4dc125049927e"\r\n
  - Authorization: Basic Y2hpbnRhbjpwYXNzd29yZA==\r\n
      Credentials: chintan:password
    \r\n
    [Full request URI: http://192.168.41.130/adcpractical/]

0000  00 0c 29 20 da 6c 00 0c  29 43 a6 fa 08 00 45 00   ..) .l.. )C....E.
0010  01 d9 07 68 40 00 80 06  1d 60 c0 a8 29 84 c0 a8   ...h@... .`..)...
0020  29 82 04 5f 00 50 53 e3  db 28 6e af 32 70 50 18   )..._.PS. .(n.2pP.
0030  fa f0 2c 30 00 00 47 45  54 20 2f 61 64 63 70 72   ..,0..GE T /adcpr
```

Authentication credentials : username and password is showin in plaintext.

Ready to load or capture
Packets: 1432 Displayed: 1387 Marked: 0 Dropped: 509 Load time: 0:00.019
Profile: Default

Figure 7: Authentication Credentials have been captured in plaintext – WITHOUT SSL

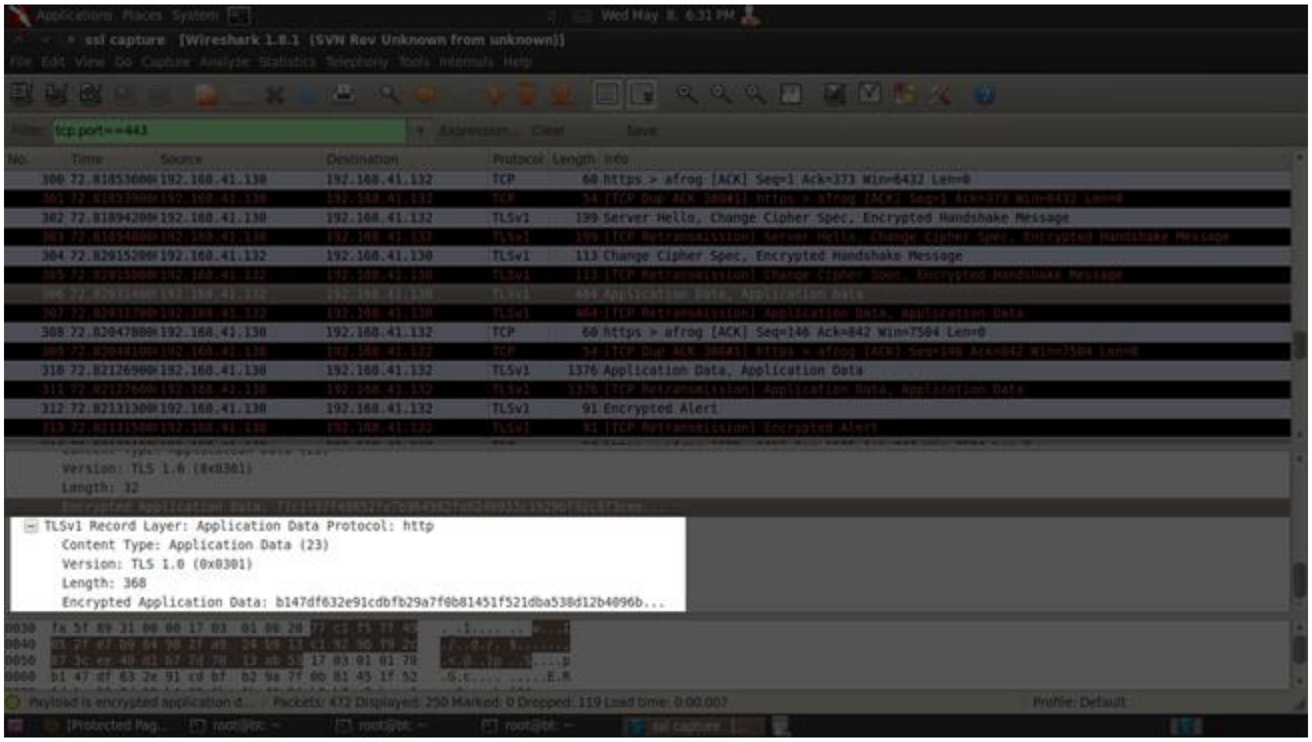
As you can see, one can capture your log-in credentials if you are not using SSL and not encrypting your data.



```
ssl capture  [Wireshark 1.8.1 (SVN Rev Unknown from unknown)]
Filter: tcp

No.  Time        Source          Destination     Protocol Length Info
  52 20.60245800 173.194.41.133  192.168.41.131  TCP        60 http > 47967 [RST, ACK] Seq=72611 Ack=2 Win=64239 Len=0
  72 33.11059400 192.168.41.132  192.168.41.130  TCP        62 nsstp > https [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
  73 33.11076800 192.168.41.130  192.168.41.132  TCP        62 https > nsstp [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
  74 33.11086500 192.168.41.132  192.168.41.130  TCP        60 nsstp > https [ACK] Seq=1 Ack=1 Win=64240 Len=0
  75 33.11136300 192.168.41.132  192.168.41.130  TLSv1     202 Client Hello
  76 33.11147100 192.168.41.130  192.168.41.132  TCP        60 https > nsstp [ACK] Seq=1 Ack=149 Win=6432 Len=0
  77 33.11715800 192.168.41.130  192.168.41.132  TLSv1    1211 Server Hello, Certificate, Server Key Exchange, Server Hello Done
  78 33.18569700 192.168.41.132  192.168.41.130  TLSv1     252 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
  79 33.19501700 192.168.41.130  192.168.41.132  TLSv1     320 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
  80 33.21553300 192.168.41.132  192.168.41.130  TLSv1      91 Encrypted Alert
  81 33.21555000 192.168.41.132  192.168.41.130  TCP        60 nsstp > https [FIN, ACK] Seq=384 Ack=1424 Win=62817 Len=0
  82 33.21555200 192.168.41.130  192.168.41.132  TLSv1      91 Encrypted Alert
  83 33.21555400 192.168.41.130  192.168.41.132  TCP        60 https > nsstp [FIN, ACK] Seq=1461 Ack=385 Win=7504 Len=0
  84 33.21573700 192.168.41.132  192.168.41.130  TCP        60 nsstp > https [RST, ACK] Seq=385 Ack=1461 Win=0 Len=0
  85 33.21591600 192.168.41.132  192.168.41.130  TCP        60 nsstp > https [RST] Seq=385 Win=0 Len=0

+ Ethernet II, Src: Vmware_d8:c7:f1 (00:0c:29:d8:c7:f1), Dst: Vmware_43:a6:fa (00:0c:29:43:a6:fa)
+ Internet Protocol Version 4, Src: 192.168.41.130 (192.168.41.130), Dst: 192.168.41.132 (192.168.41.132)
- Transmission Control Protocol, Src Port: https (443), Dst Port: nsstp (1036), Seq: 1, Ack: 149, Len: 1157
    Source port: https (443)
    Destination port: nsstp (1036)
    [Stream index: 1]

0000  00 0c 29 43 a6 fa 00 0c  29 d8 c7 f1 08 00 45 00   ..)C.... )......E.
0010  04 ad ae 01 40 00 40 06  b3 f2 c0 a8 29 82 c0 a8   ....@.@. ....)...
0020  29 84 01 bb 04 0c 00 c4  a1 f7 aa 4e 33 55 50 18   )....... ...N3UP.
0030  19 20 fe 19 00 00 16 03  01 00 35 02 00 00 31 03   . ...... ..5...1.

Ready to load or capture    Packets: 472 Displayed: 357 Marked: 0 Dropped: 119 Load time: 0:00.008    Profile: Default
```

Figure 8 : SSL Handshake is Being Done

The SSL handshake is being done now in the next picture; you will see the cipher text being generated and passing through the network.



```
ssl capture  [Wireshark 1.8.1 (SVN Rev Unknown from unknown)]
Filter: tcp.port==443

No.  Time        Source          Destination     Protocol Length Info
 300 72.81853600 192.168.41.130  192.168.41.132  TCP        60 https > afrog [ACK] Seq=1 Ack=373 Win=6432 Len=0
 301 72.81853900 192.168.41.130  192.168.41.132  TCP        54 [TCP Dup ACK 300#1] https > afrog [ACK] Seq=1 Ack=373 Win=6432 Len=0
 302 72.81894200 192.168.41.130  192.168.41.132  TLSv1     199 Server Hello, Change Cipher Spec, Encrypted Handshake Message
 303 72.81894800 192.168.41.130  192.168.41.132  TLSv1     199 [TCP Retransmission] Server Hello, Change Cipher Spec, Encrypted Handshake Message
 304 72.82915200 192.168.41.132  192.168.41.130  TLSv1     113 Change Cipher Spec, Encrypted Handshake Message
 305 72.82915900 192.168.41.132  192.168.41.130  TLSv1     113 [TCP Retransmission] Change Cipher Spec, Encrypted Handshake Message
 306 72.82931400 192.168.41.132  192.168.41.130  TLSv1     464 Application Data, Application Data
 307 72.82931700 192.168.41.132  192.168.41.130  TLSv1     464 [TCP Retransmission] Application Data, Application Data
 308 72.82047800 192.168.41.130  192.168.41.132  TCP        60 https > afrog [ACK] Seq=146 Ack=842 Win=7504 Len=0
 309 72.82048100 192.168.41.130  192.168.41.132  TCP        54 [TCP Dup ACK 308#1] https > afrog [ACK] Seq=146 Ack=842 Win=7504 Len=0
 310 72.82126900 192.168.41.130  192.168.41.132  TLSv1    1376 Application Data, Application Data
 311 72.82127600 192.168.41.130  192.168.41.132  TLSv1    1376 [TCP Retransmission] Application Data, Application Data
 312 72.82131300 192.168.41.130  192.168.41.132  TLSv1      91 Encrypted Alert
 313 72.82131500 192.168.41.130  192.168.41.132  TLSv1      91 [TCP Retransmission] Encrypted Alert

    Version: TLS 1.0 (0x0301)
    Length: 32
    Encrypted Application Data: [illegible]
- TLSv1 Record Layer: Application Data Protocol: http
    Content Type: Application Data (23)
    Version: TLS 1.0 (0x0301)
    Length: 368
    Encrypted Application Data: b147df632e91cdbfb29a7f0b81451f521dba538d12b4096b...

0030  fa 5f 89 31 00 00 17 03  01 00 20 [illegible]
0040  [illegible]
0050  [illegible]  17 03 01 01 70
0060  b1 47 df 63 2e 91 cd bf  b2 9a 7f 0b 81 45 1f 52   .G.c.... .....E.R

Payload is encrypted application d...    Packets: 472 Displayed: 250 Marked: 0 Dropped: 119 Load time: 0:00.007    Profile: Default
```

As you can see here, the credentials are encrypted now. No one can see the real credentials of the client passing through the network. Thus we see how SSL is useful to encrypt the data.

- **Conclusion**

  SSL can give protection against man-in-the-middle attacks. It can protect sensitive data such as log-in credentials, credit card details, home address, phone number, etc.

About Author

Chintan Gurjar post graduated in Computer Security & Forensics from London (UK), Now Working as System Security Analyst at **Lucideus Tech Pvt Ltd.** He has contributed articles to Europe based magazine namely “Hakin9”, "PenTestMag".

Talk to Chintan : Facebook

LUCIDEUS™
SECURING CYBER SPACE

Lucideus Winter Certification Training

Lucideus introduces Asia's coolest 40 hour program on **Information Security & Digital Forensics**. Buckle up as you are about to be bombarded with doses of awesomeness like never before.

About Winter Training
It's a training program where you get to see and try from your own hands how a real world hacker works so that you can defend yourself from the attacks.
We deliver this training in 2 ways:

Delivery Mode	Duration Per Day	No of days	Course Duration
LCCSA - Regular	2 Hours	20 days	40 Hours
LCCSA - Boot camp	8 Hours	5 days	40 Hours

Course Contents

1. Web Application Hacking & Security
2. Email Accounts Hacking & Security
3. Mobile Hacking & Security
4. System Hacking & Security
5. Wireless Hacking & Security
6. Reverse Engineering
7. Metasploit Framework
8. Cyber Forensics
9. Cyber Crime Investigation

And many more…

Why Lucideus Training?

☞ 95% practical sessions with hands-on training.
☞ Trained over 60,000 students from more than 200 establishments across the globe.
☞ Experienced and highly qualified trainers.
☞ Most advanced IT Security Labs.
☞ Live Projects during training. (Offered to desired candidates)
☞ Maximum of 12 students per batch.
☞ Internship Opportunity to deservers.
☞ Placement assistance to desired and deserver candidates.

LUCIDEUS™
SECURING CYBER SPACE

Batch Dates

Duration	Start Date	End Date
5 Day Boot camp	22nd Dec	27th Dec
5 Day Boot camp	29th Dec	4th Jan
5 Day Boot camp	5th Jan	10th Jan
5 Day Boot camp	12th Jan	17th Jan
5 Day Boot camp	19th Jan	24th Jan
5 Day Boot camp	27th Jan	2nd Feb
Duration	Start Date	End Date
1 Month Regular	24th Nov	24th Dec
1 Month Regular	8th Dec	7th Jan
1 Month Regular	15th Dec	15th Jan
1 Month Regular	22nd Dec	20th Jan
1 Month Regular	29nd Dec	28th Jan

Who all will teach us in the course?

Saket Modi   Vidit Baxi
Rahul Tyagi   Aman Sachdev
Rajat Sharma   Atul Jalan

Don't know who they are?
Try and ask Google, it will help you with all the required info.

TO REGISTER THIS WINTER FILL THE FORM:
http://www.lucideus.com/form/winter.html

References Taken by Author for The Article

1. http://pubs.vmware.com/view-51/index.jsp?topic=%2Fcom.vmware.view.installation.doc%2FGUID-B238BB47-D710-4DAB-945F-B54D46164FE0.html
2. http://validationcertificateinfo.blogspot.co.uk/2012/12/the-advantages-and-disadvantages-of.html
3. http://uk.godaddy.com/ssl/ssl-certificates.aspx
4. http://en.wikipedia.org/wiki/Transport_Layer_Security
5. https://developer.mozilla.org/en-US/docs/NSS

6. http://tools.ietf.org/html/rfc6101
7. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html
8. http://www.lucideus.com/winter