HP Jetdirect and SSL/TLS

whitepaper

June 2008

Table of Contents:

Introduction

HP Jetdirect introduced SSL/TLS support in early 2002 with the 615n EIO Print Server. A free firmware upgrade allowed the 610n EIO print server, shipped in 2000, the same capability. Suddenly, a few million HP Jetdirect EIO cards had SSL/TLS capability. Why?

The answer was secure management. HP printing and imaging devices were becoming more complex and more feature oriented. They were becoming valuable assets to a company's infrastructure. Having the ability to use a browser to manage a device using HTTP was one thing, using the same browser and using HTTPS to manage it securely was a great benefit. Unfortunately, many users of HTTPS are under a false sense of security because they have not deployed SSL/TLS

correctly. One of the purposes of this whitepaper is to show administrators how to properly deploy SSL/TLS so that it can be used securely.

SSL/TLS is also used in other applications, such as LDAPS and 802.1X. This whitepaper will discuss how SSL/TLS works when Jetdirect is operating as a client (e.g., LDAPS, IPPS). 802.1X is covered extensively in a separate whitepaper. See http://www.hp.com/go/secureprinting for the latest information regarding HP's printing and imaging products.

What is SSL/TLS?

SSL/TLS is a security protocol. It has a purpose: To provide authentication, integrity, and confidentiality to the data it encapsulates. While SSL/TLS is commonly associated with the TCP/IP protocol suite, it can be used within other frameworks as well. The most common protocol that uses SSL/TLS functionality is HTTPS. In this section, for the sake of familiarity and clarity, we'll discuss SSL/TLS within the context of TCP/IP, primarily with HTTP and HTTPS.

Refer to Figure 1: HTTP Application. Here is the normal view of an HTTP session from a web browser to a Jetdirect device.

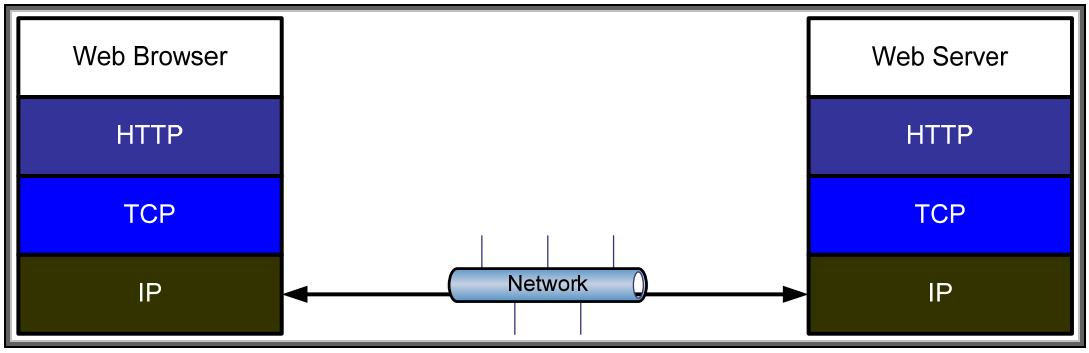


Figure 1 - HTTP Application

In Figure 2 – HTTPS Application, we can see how SSL/TLS is deployed. This would be done by using "HTTPS" in the URL of the browser.

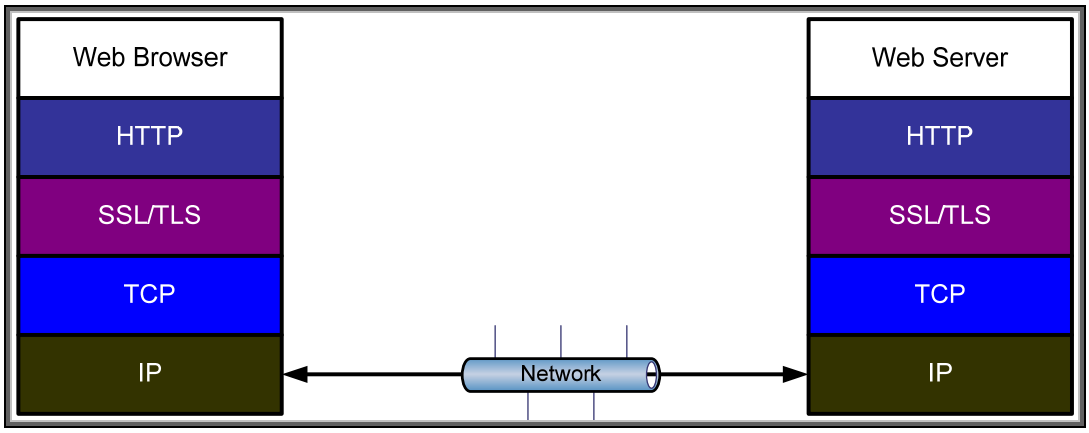


Figure 2 - HTTPS Application

We can see that HTTPS is really just running HTTP over SSL/TLS which runs over TCP. How does the browser know when to use SSL/TLS? Well, the URL of "https://" indicates to the browser that it needs to change its behavior and invoke SSL/TLS. Refer to Figure 3 – Application Changes. We can

see that SSL/TLS requires application changes in order to be utilized. These changes have to be made by every application that wishes to utilize SSL/TLS. In other words, SSL/TLS is not application transparent.

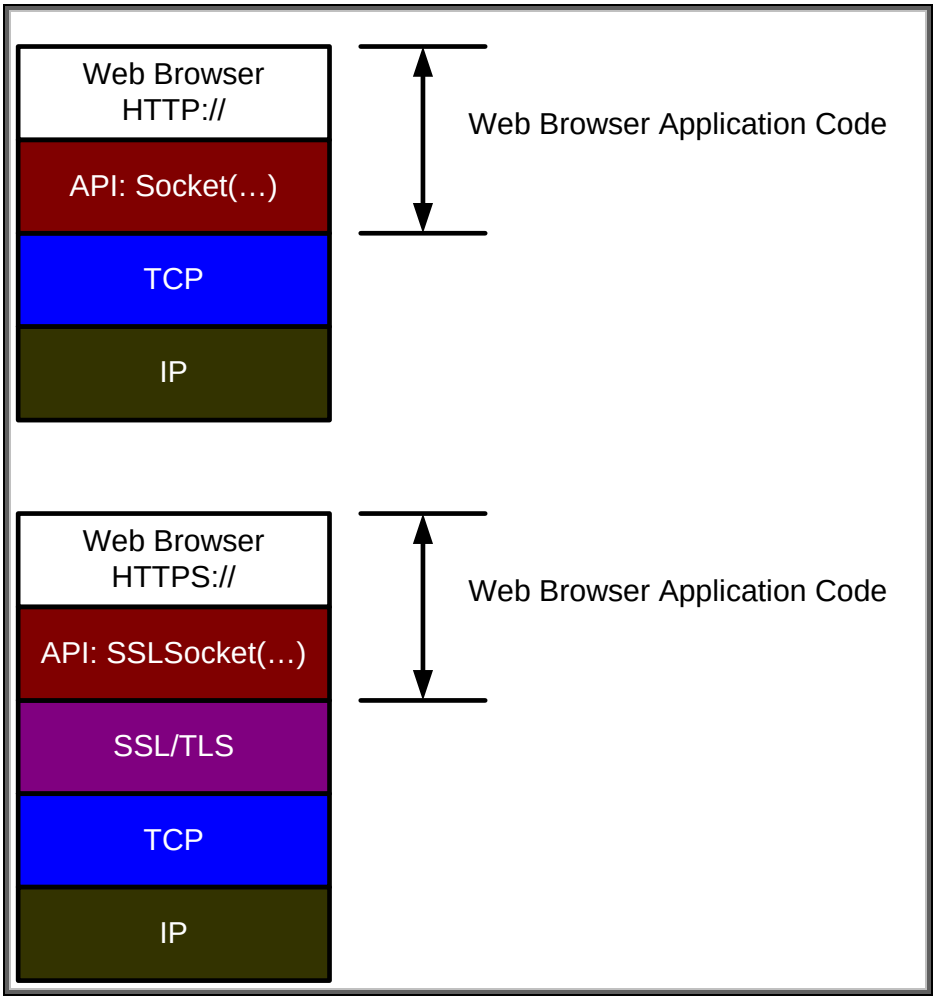


Figure 3 - Application Changes

Now, let's have a closer look at HTTPS.

HTTPS Decoded

In Figure 4 – HTTP Session, we bring up a normal HTTP session with an HP MFP.

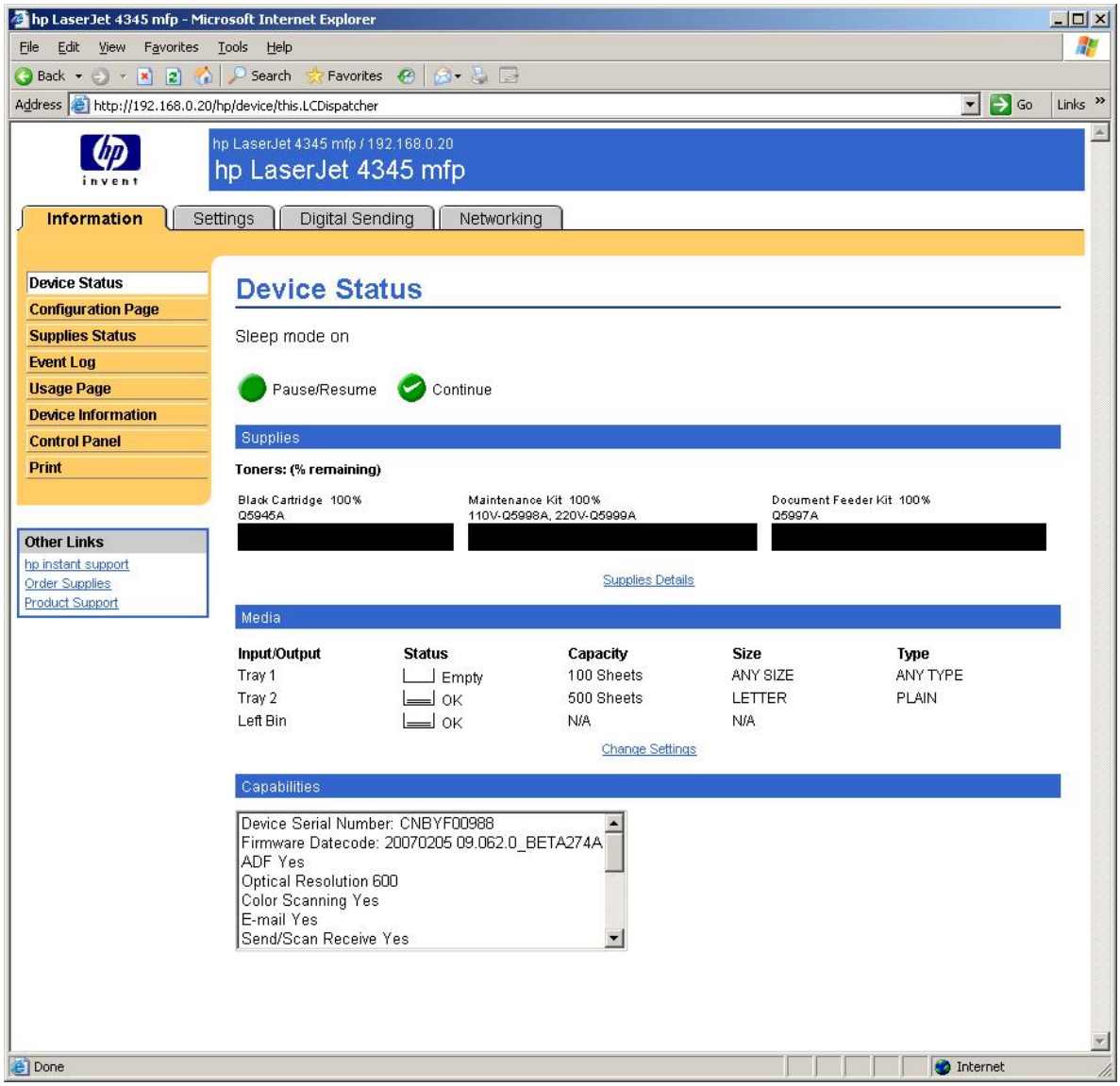


Figure 4 - HTTP Session

The URL starts with http:// and tells the browser that SSL/TLS is not required. Let's change it to https:// and hit the [Enter] key. We are now presented with the dialog in Figure 5.

Figure 5 - Secure Connection

Clicking "More Info", we get the dialog in Figure 6.

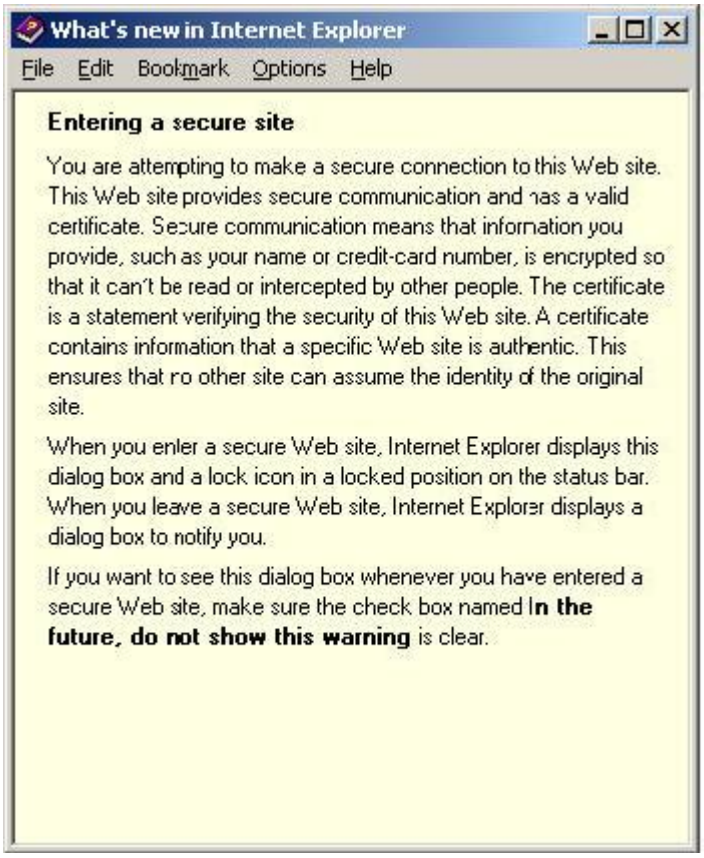


Figure 6 - More Info

Notice the sentence: "This Web site provides secure communication and has a valid certificate." After reading this whitepaper, you'll know whether that sentence is correct or not. Now that we have read the "More Info" text, let's go back to the dialog in Figure 5 and click "OK". Now we get a security alert dialog shown in Figure 7.

Figure 7 - Security Alert

Well, we've got one green checkmark and two yellow warnings. Good enough for us! Let's click "Yes" and establish the HTTPS session with the MFP.

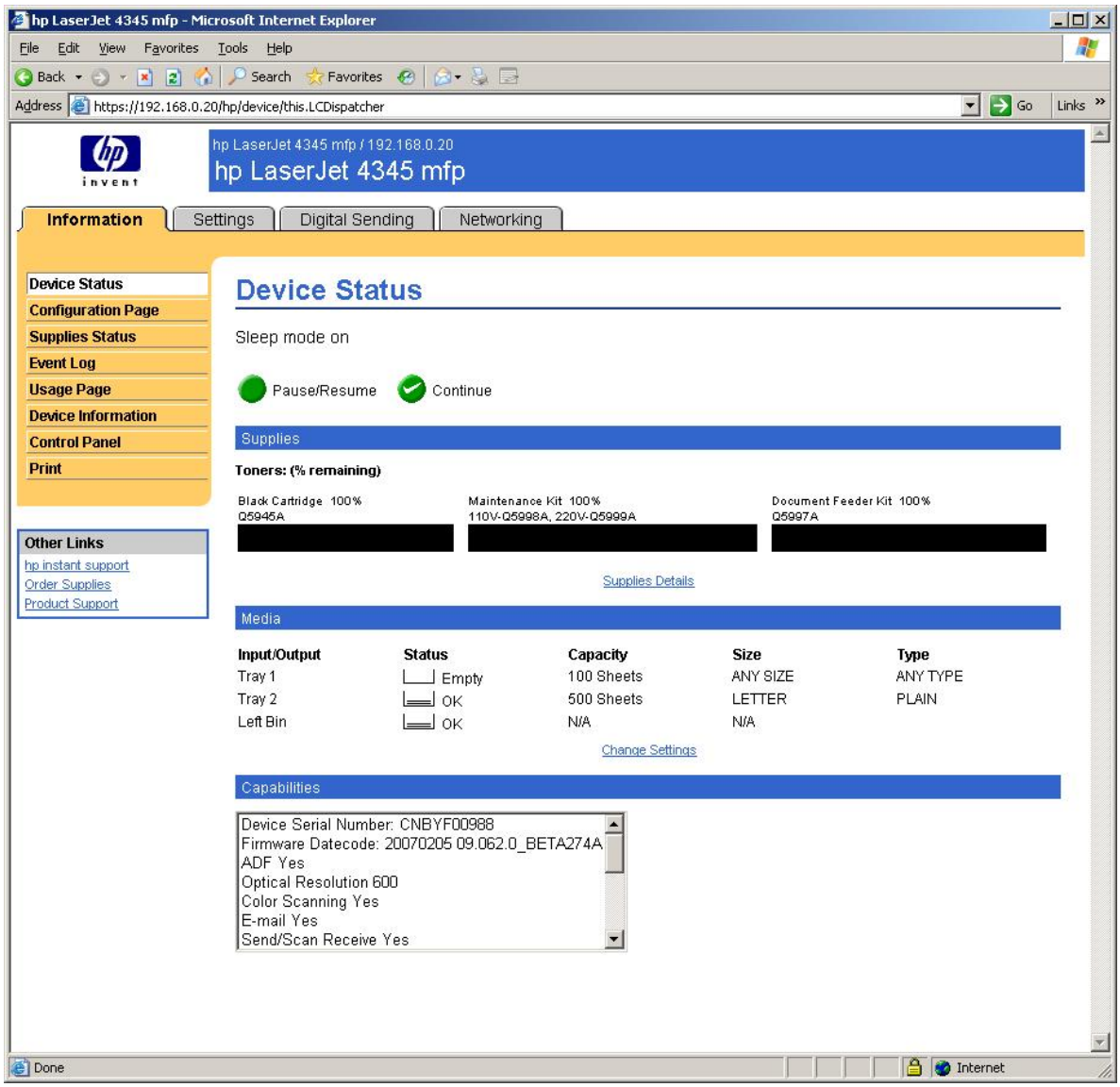


Figure 8 - HTTPS Session

We have now "secured" our web browser session with the HP MFP. How can we tell? Well, we can look at the URL and see https://, but we can also look at the bottom right of Figure 9 – Lock Icon.

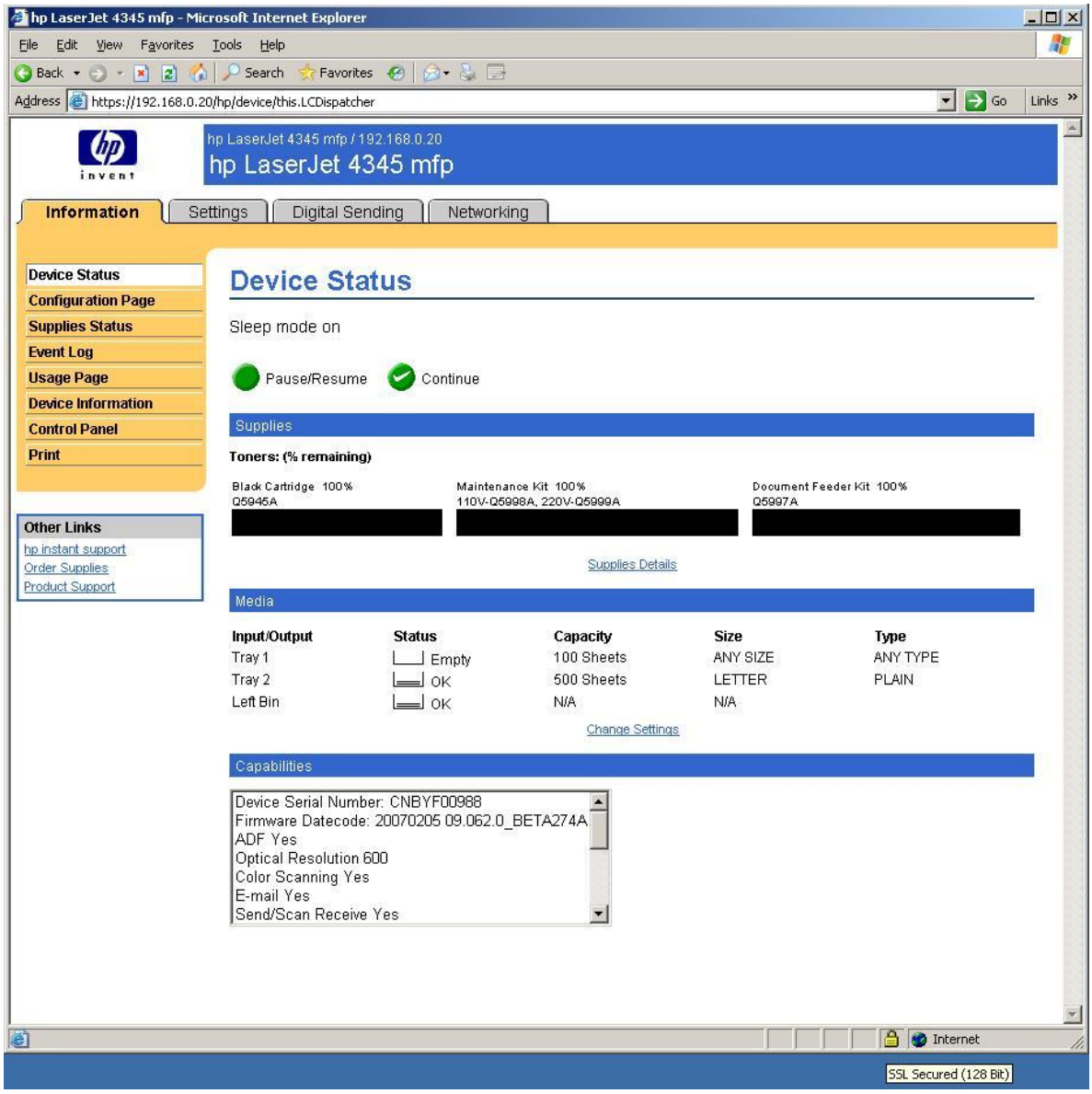


Figure 9 - Lock Icon

The mouse pointer was placed on the lock icon. Notice the "SSL Secured (128 bit)" shown in the bottom right. If we double-click on the lock icon, we get a dialog box similar to the one in Figure 10 – Certificate Details.

Figure 10 - Certificate Details

Something is very wrong here. We went from yellow warning symbols and green checkmark to a big red X symbol. We have a 128 bit SSL secured session with the HP MFP but we now have a big red X indicating a trust problem.

This problem is best explained through an example. Let's assume that you are a famous celebrity and that you are having an electrical contractor work on your mansion. The contractor completed the work but will only accept cash as payment and you don't have any cash on you. You have to go to the Automated Teller Machine (ATM). You summon 10 bodyguards and get into your armor plated limousine and go to the shopping mall. The bodyguards surround you and you go to the first ATM you see and put in your card and punch in your PIN#. The ATM returns the message: "Temporarily out of service" and gives you your card back. You then go to another ATM and get the money and return home. The next day, your bank account is cleaned out. You assume that one of your body guards saw your PIN # and "borrowed" your card while you were sleeping. You fire all your bodyguards. Was that the correct thing to do?

Probably not. You were most likely a victim of a fake ATM machine. You went to the ATM machine in a secure fashion. You left the ATM machine in a secure fashion. You just went to a non-trusted ATM machine! The same type of attack exists with SSL/TLS. To avoid being a victim of this attack, we need to pay attention to the digital certificate and to the dialog boxes associated with the SSL/TLS connection. In short, although we used HTTPS in this example to "secure" our session with the HP MFP, we actually are not secure and what is worse, we probably have a false sense of security. In order to utilize SSL/TLS securely, we need to learn more about digital certificates.

Digital Certificates

Much like a fake ATM machine, an unethical hacker could use technology to direct a user to a false web site when they are thinking they are going to a trusted website, even if they are using SSL. When typing in https://, Internet Explorer 6 (IE6) will pop-up a dialog when it encounters a digital certificate that it doesn't trust (i.e., a potential fake ATM machine) as shown in Figure 11:

Figure 11 - IE6 Security Alert

In many cases, a user may just click "Yes" without realizing what they are doing (as we did before) and then provide the unethical hacker with a lot of information – like their credit card number and billing address. After all, it really seems like just an annoying dialog. Luckily, the Internet Explorer 7 (IE7) experience is different in a profound way. Here is an example of IE7:

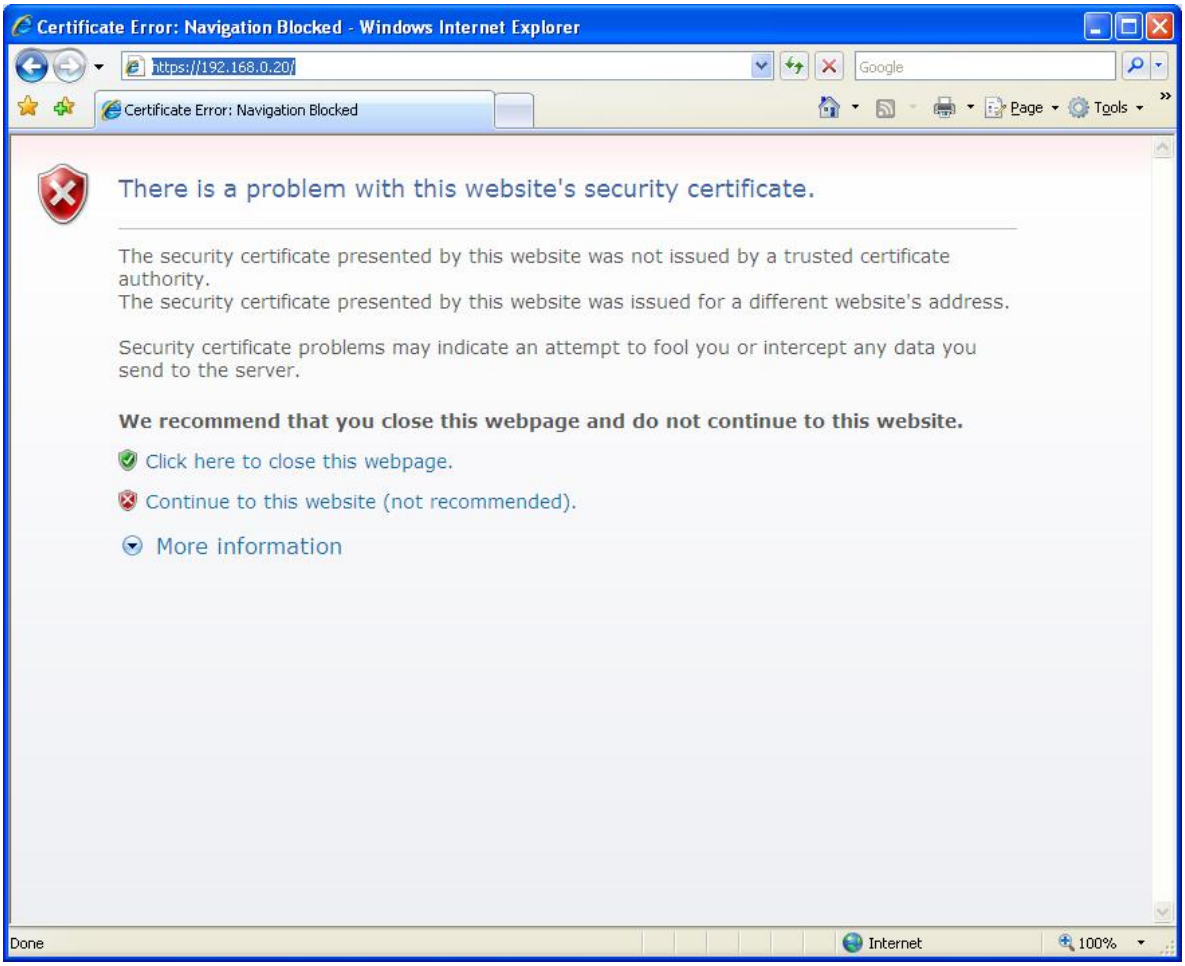


Figure 12 - IE7 Certificate Error 1

This screen is a lot different from IE6 – notice the red X symbols and explanatory text. The way the information is now presented, it will grab your attention. If we click the "Continue to this website (not recommended)" link, we get this screen shown in Figure 13:

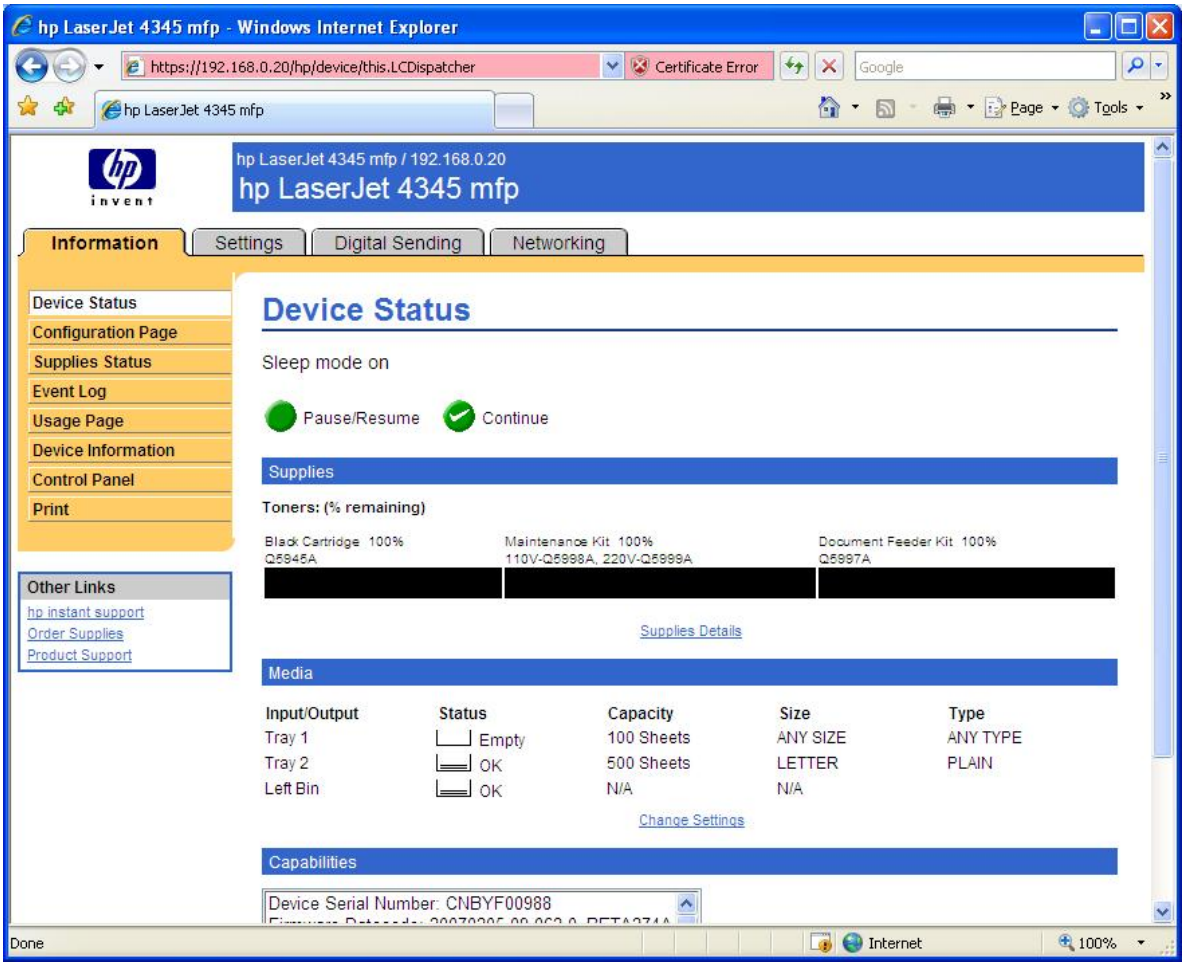


Figure 13 – IE7 Certificate Error 2

Notice the red URL and the "Certificate Error" message. Essentially, to go back to our story, Internet Explorer 7 is effectively saying "You may be at a fake ATM machine!"

The big question is "How can digital certificates help me determine that I'm going to the right website?" Well, there are two main components around digital certificates – the digital certificate issuer (Issued by:) and the holder of the digital certificate (Issued to:). A useful analogy is to think of the certificate issuer like a Department of Motor Vehicles (DMV). Each state in the United States has a DMV run by the state's government. The DMV issues driver's licenses which grant the privilege to drive in a given state. A person that goes to the DMV to get a driver's license must pass a series of tests that helps the DMV determine if they are fit to drive on the state's roads. The state's Highway Patrol, a group which enforces the rules of the road, recognizes the validity of the DMV to issue driver's licenses. Therefore, if one violates one of the rules of the road and is pulled over by a Highway Patrol officer, showing a driver's license issued by the DMV is a requirement. The Highway Patrol will not recognize a driver's license issued by an institution other than the DMV as being valid. In short, the DMV is a trusted third party that issues "certificates" (driver's licenses) to individuals. These "certificates", issued by the DMV, are trusted by the Highway Patrol. Essentially, the Highway Patrol, the DMV, and the licensed driver are the participants in a Driver's License Infrastructure or DLI. Let's move back to digital certificates and talk about a Public Key Infrastructure.

Let's go back to the certificate information dialog, shown in Figure 14:

Figure 14 - Certificate Information

Here is the message: "This CA Root certificate is not trusted." To enable trust, install this certificate in the Trusted Root Certification Authorities store". What the message is trying to say is that "HP Jetdirect 85C1F319", who issued the certificate "HP Jetdirect 85C1F319", is not trusted. Because the "Issued by:" name is the same as the "Issued to:" name, this is a self-signed certificate.

The Security Alert dialog is troubling because it is indicative of a trust problem. In the terms of our analogy, it would be like a driver, who has been pulled over by the Highway Patrol, handing the officer a driver's license that the driver has created for himself indicating that he has the privilege to drive in the state. The Highway Patrol would obviously not trust it and unfortunately may not consider it a laughing matter.

In essence, a digital certificate, one used by computers, binds an identity to a key and needs to be issued by a trusted third party. What is a key? A key is a secret that is used in cryptographic algorithms. There are public keys and private keys used for asymmetric cryptography and symmetric keys used for symmetric cryptography. Let's look at symmetric cryptography first.

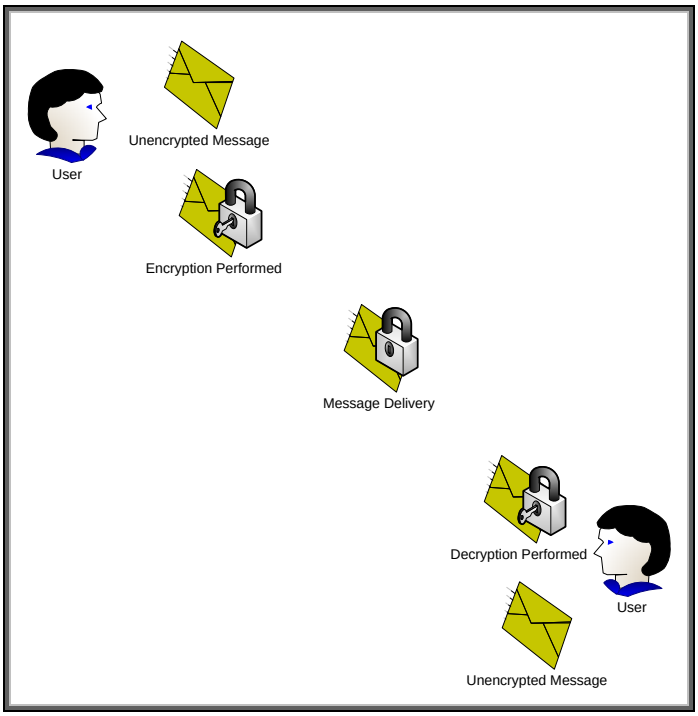


Figure 15 - Symmetric Cryptography

In Figure 15, the confidentiality provided to the message is done via a single key. Because the same key is used for encryption and decryption, this process is known as symmetric cryptography. Symmetric cryptography commonly has two attributes associated with it:

- It performs well – it is fast and easy to implement
- It has a key distribution problem – how do you get the symmetric key to everyone that needs it in a secure way?

Asymmetric cryptography is also available and functions very different than symmetric cryptography. It has two keys – one Public and one Private. The private key is not shared with anyone. The Public key is like a public telephone number. You can share it with everyone. Let's look at Figure 16 – Asymmetric Cryptography.

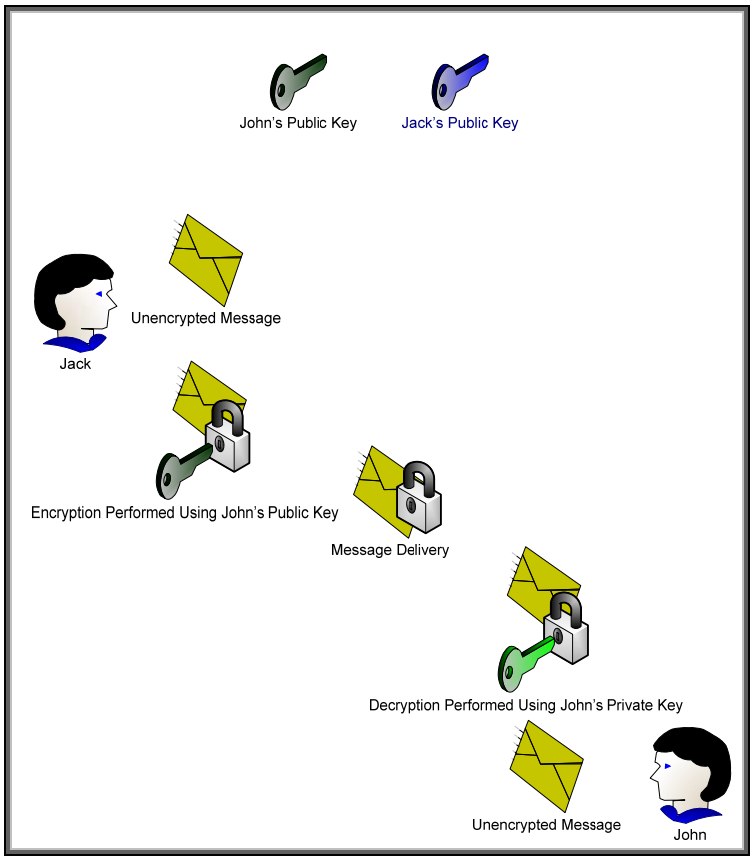


Figure 16 - Asymmetric Cryptography

Here we can see the difference between asymmetric and symmetric cryptography. One key can be used for encryption and then the corresponding key can be used for decryption. It appears that asymmetric cryptography has solved the key distribution issue; however there are two new attributes usually associated with asymmetric cryptography

- It is slow
- It has a trust problem. How do I know that this is John's public key and not someone pretending to be John?

To solve the first problem, asymmetric cryptography is usually used to securely distribute symmetric keys and sign hash codes. In short, what is actually being encrypted and decrypted is usually much smaller than actual messages. This has the nice benefit of solving the key distribution issue with symmetrical cryptography. So, in essence, symmetric keys are sent securely using asymmetric cryptography and the actual messages themselves are protected using symmetric cryptography. Cool! We get the flexibility of asymmetric cryptography and the speed of symmetric cryptography. Now we only have to solve the trust problem.

In order to solve the trust problem, five things will need to be discussed:

- A certificate authority – a trusted third party that creates digital certificates from certificate requests
- A certificate request – a public key associated with identity information that will serve as the basic building block for a digital certificate that the certificate authority will create and sign.
- A digital certificate – a public key associated with identity information that is digitally signed by the certificate authority.
- A digital signature – the hash of the digital certificate encrypted by the private key of the certificate authority.

- A hash – also known as a message digest. A hash is the output of a one way function that attempts to ensure the integrity of the message (i.e., that the message has not been altered). It is usually combined with authentication information to ensure that the message originator can be authenticated and that the integrity of the message has not been disrupted. You can think of a hash like an advanced checksum or an advanced cyclic redundancy check (CRC).

Let's cover hashes and digital signatures first. We'll assume that Jack wants to send John a message. Jack wants to make sure that John knows the message came from him and that the message was not altered in transit. However, Jack doesn't care about confidentiality – in other words, the actual message can be sent "in the clear" – but does care about authentication and integrity. We can accomplish this through hashes and digital signatures as shown in Figure 17.

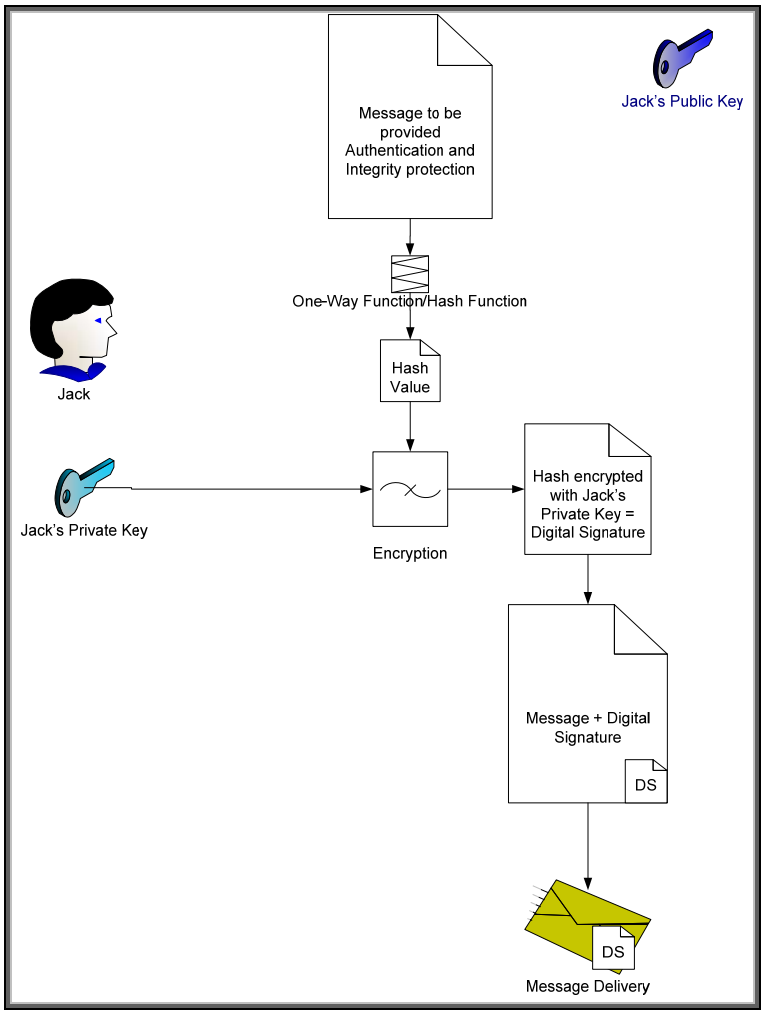


Figure 17 - Digital Signature

In Figure 17, Jack has sent John a message with a digital signature. Let's see how John would validate this message to make sure it came from Jack and was not altered. Refer to Figure 18.

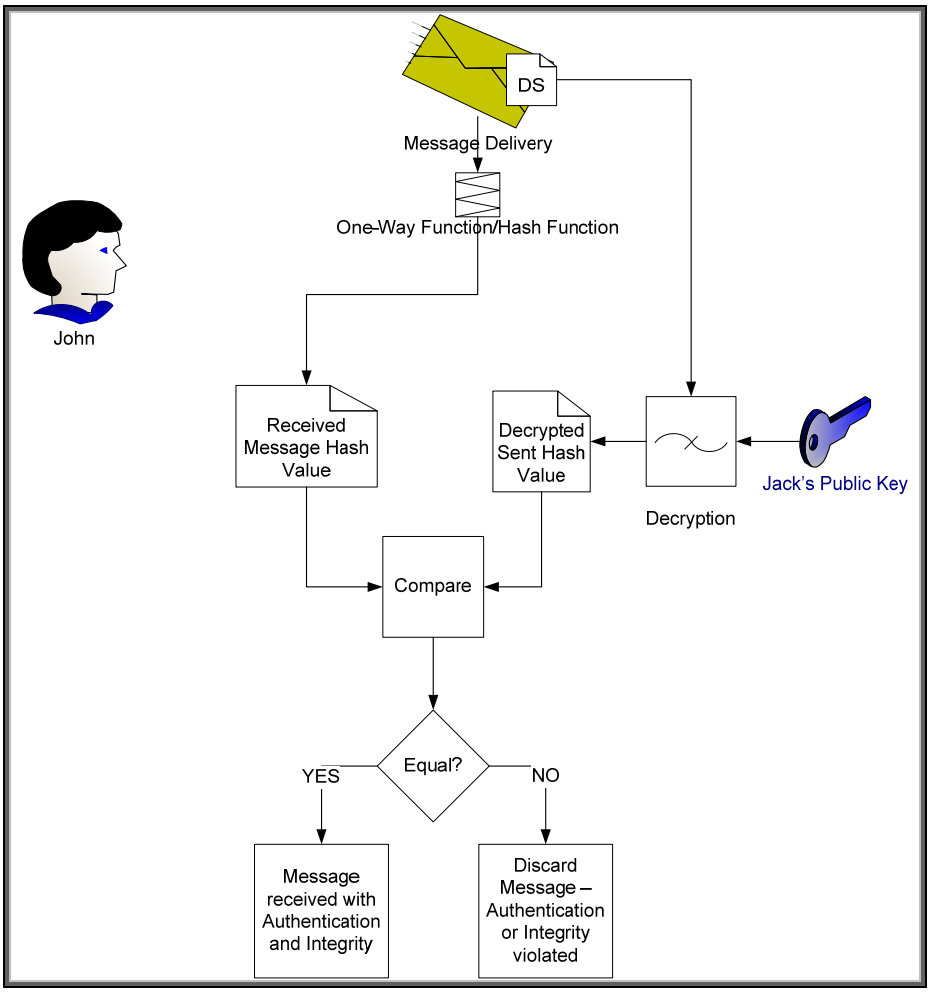


Figure 18 - Digital Signature Verification

Here we see how John uses Jack's public key to verify the message. Jack's public key is the only key that can decrypt the digital signature and obtain the hash value of the message that Jack calculated before sending the message. Because the hash was encrypted with Jack's private key, which no one should know but Jack, John can be sure that Jack was the one that sent it.

We still have a problem – How does John know that Jack's public key really belongs to the person that he knows as "Jack"? There are many people in the world named "Jack" – how does John know it isn't one of them? We still need a trusted third party to provide Jack's public key in a format John can trust and we probably need Jack to provide a little more identity information too. Here is where the Certificate Authority comes into play. Refer to Figure 19 – Certificate Authority.

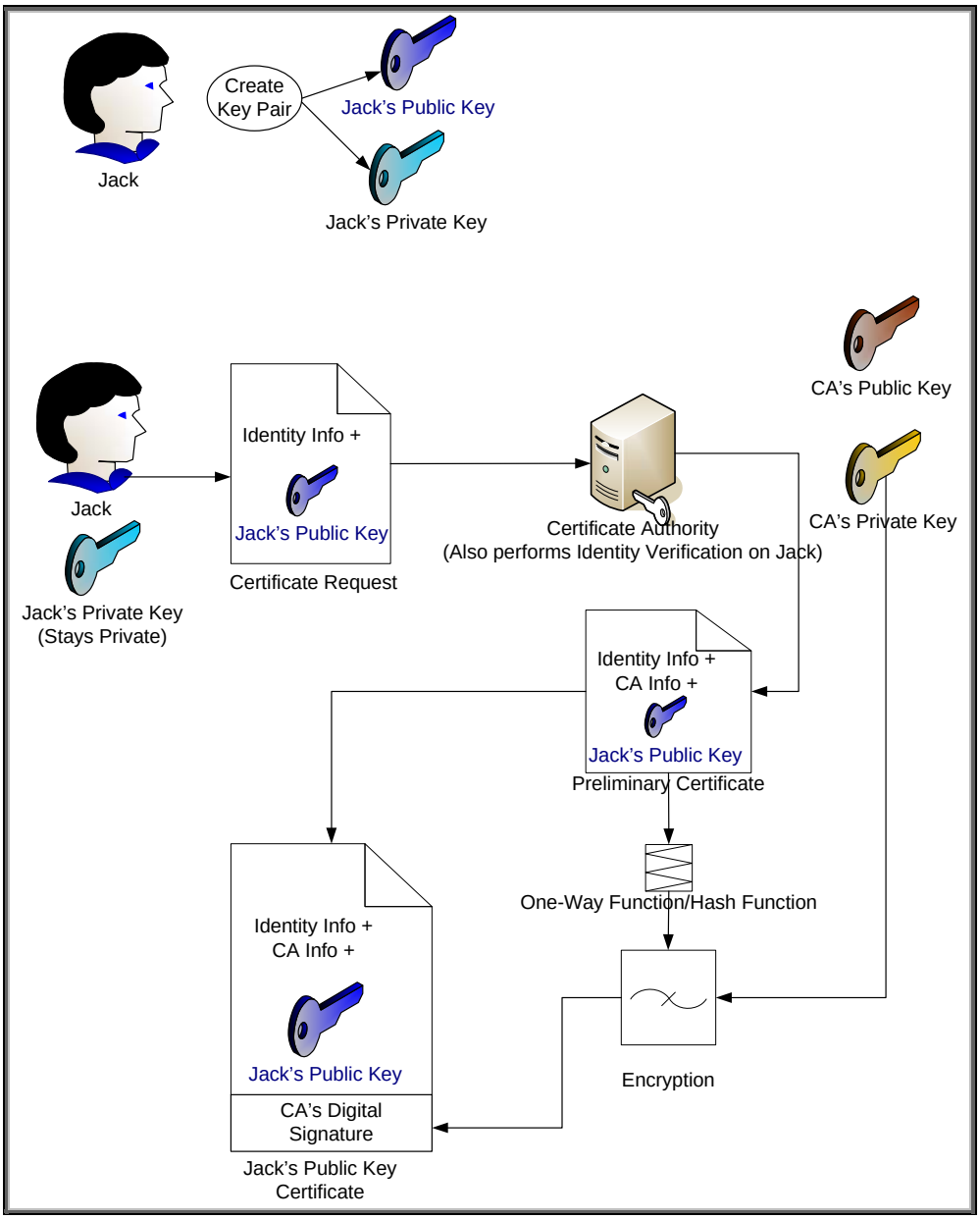


Figure 19 - Certificate Authority

Jack goes through a key pair generation process and creates a public and private key pair. The private key is kept secret. The public key is associated with some identity information and is given to a Certificate Authority. The certificate authority generates a certificate, usually specific to a purpose such as email, and signs the certificate with its digital signature. Assuming there is a place where these digital certificates are publicly available, as long as Jack and John can agree to trust a specific certificate authority, they'll be fine trusting certificates signed by that authority. Refer to Figure 20.

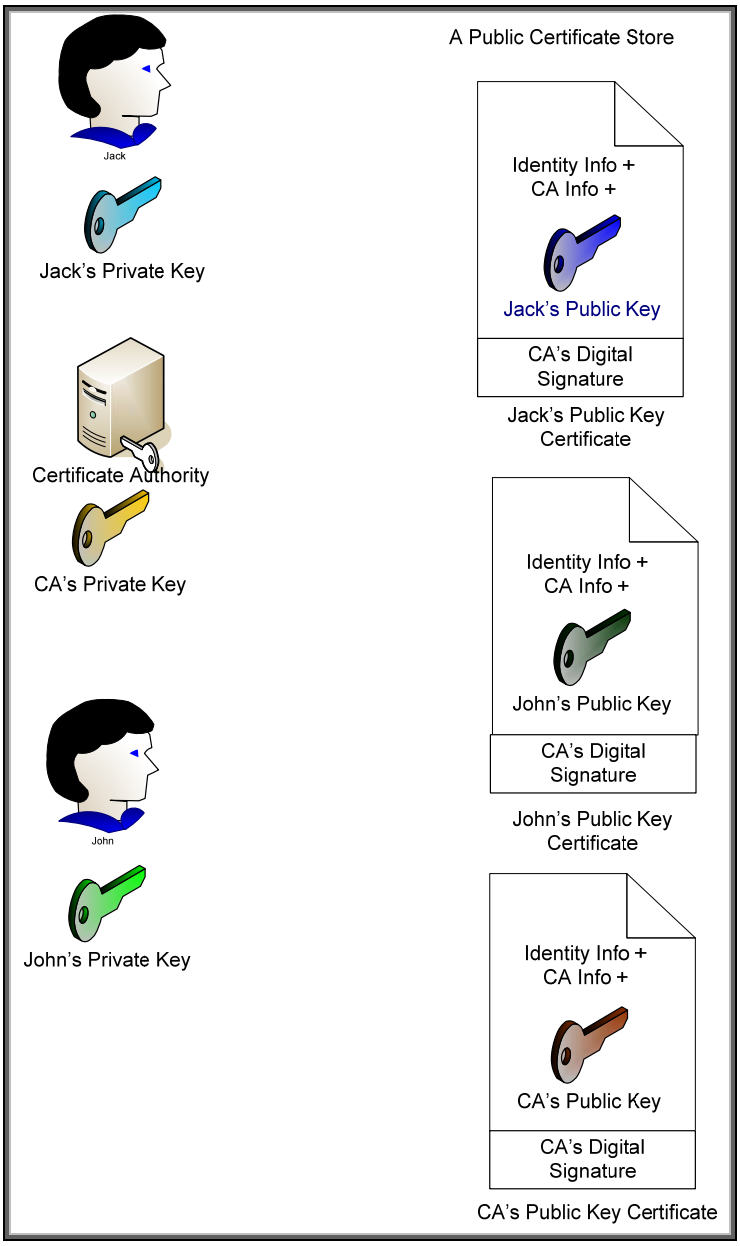


Figure 20 - Public Key Certificates

Here we can see that everyone's public key certificate is, well – um, public. The important thing to note is that the certificate authority also has a public key certificate that identifies itself. This certificate is signed with its own private key and is a "self-signed" certificate. As you may remember, Jetdirect also creates a self-signed certificate. What is the difference between a certificate authority's self-signed certificate and Jetdirect's self-signed certificate? Good Question! First let's describe what a self-signed certificate actually is. Let's assume Jack realizes that he doesn't have a CA but he needs a certificate. Here is what he does

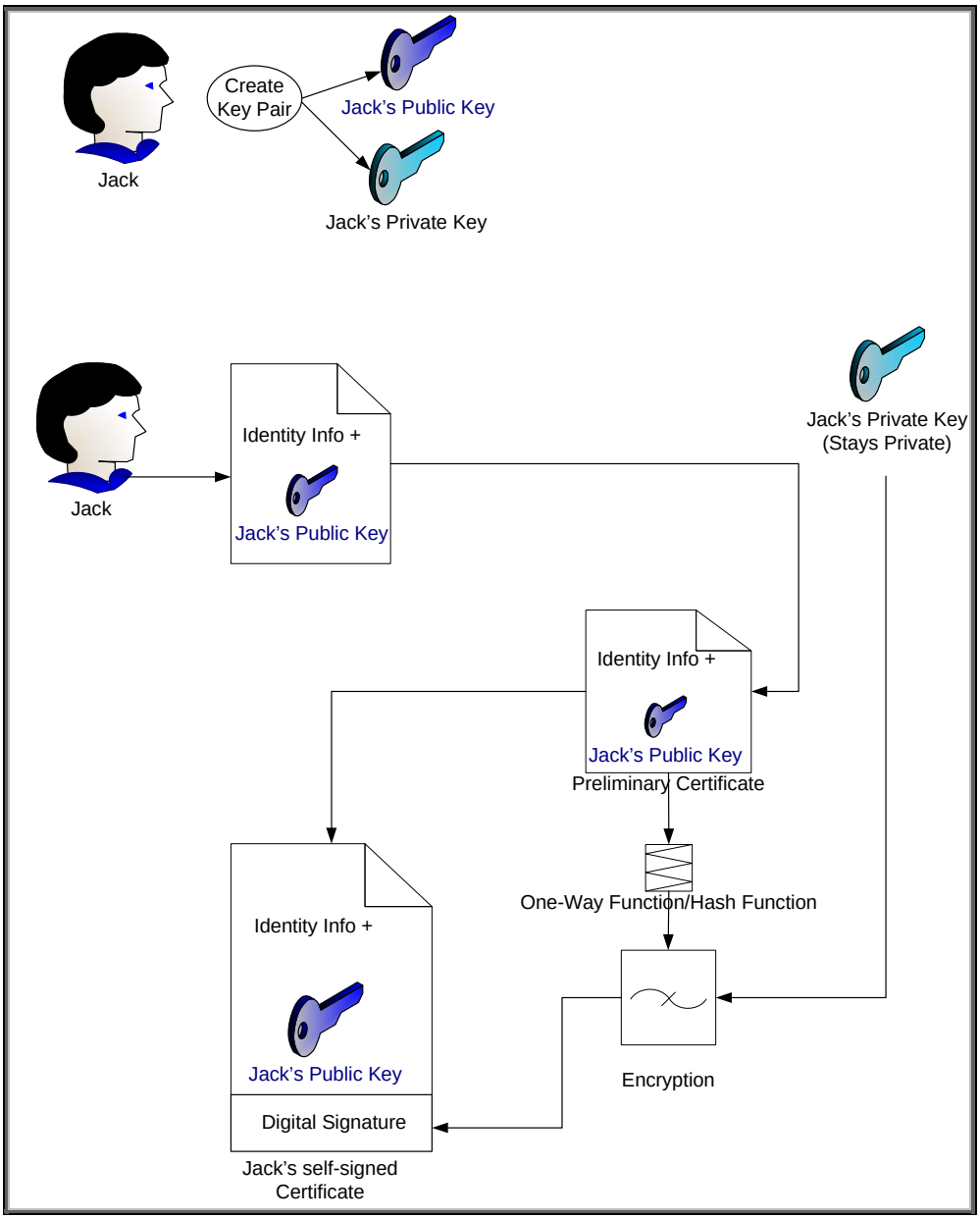


Figure 21 - Self-Signed Certificate

Basically, Jack's private key does the signing on his public key certificate. A root (top of the chain) certificate authority is going to go through the same process. So why is it okay for a Root CA to have a self-signed certificate but no one else to have one? Well, it is all about trust. There is no "higher" level of trust then the top level certificate authority. Therefore, in our previous examples, John and Jack must choose a particular certificate authority that they both trust. In most cases, there is a hierarchy of certificate authorities at customer sites. This forms what is known as a certificate chain and there is a top level CA or Root CA where the ultimate trust resides.

Also, we should take care to point out that there is usually a difference between Internet trust using certificates and Intranet trust using certificates. Internet trust will involve well-known certificate authorities like Verisign and Entrust. However, Intranet models usually revolve around Microsoft's certificate authority that comes with Windows 2003 server. Each company establishes their own Public Key Infrastructure (PKI) that includes an entire policy around certificates.

There is one other important thing to cover about certificates. Each certificate has a one or more "certificate purposes" that the certificate can be used for. For example, a Jetdirect self-signed certificate will have two purposes: client authentication and server authentication. A root certificate

authority's self-signed certificate will have a purpose to create certificates for other entities, usually subordinate certificate authorities. It may be of help to go back to our driver's license example to explain certificate purposes. A driver's license purpose is to clearly identify the person it has been issued to and to show that that person has the right to drive in a given state. Because a driver's license also lists the date of birth, it is often used to determine age and whether the holder is able to purchase various products that have age limitations. This purpose is actually above and beyond the original purpose of a driver's license. In the digital certificate world, this additional purpose would more than likely not be allowed.

So, can't someone who is not a CA create a self-signed certificate with the ability to create other certificates for entities? Sure they can! Will this be trusted? Probably not. However, if an unethical hacker can somehow install a CA certificate of their own choosing into your trusted certificate store, you will be in for a lot of problems. They will now have the ability to fool the browser and other applications into connecting to malicious sites that are now "trusted" and the browser or other application will not be able to detect it. Keep that certificate store protected!

SSL/TLS Protocol Basics

Okay, now that we know something about SSL/TLS basics and a PKI, we can talk about how the SSL/TLS protocol goes about its business. While there are many interesting protocol specifics, we are only going to talk about common situations with HP Jetdirect and "normal" SSL/TLS protocol interactions. A basic breakdown of SSL/TLS protocol structures is shown in Figure 22:

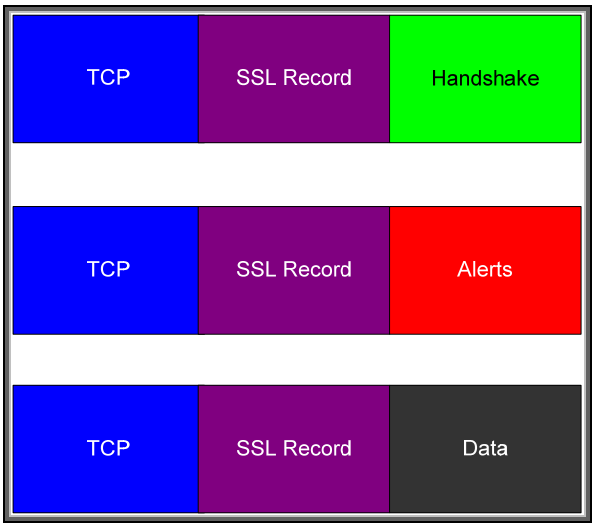


Figure 22 - SSL/TLS Protocol Structures

(Note: In order to enhance understanding, this diagram was simplified. Please refer to the many excellent SSL/TLS references for a more complete and more accurate protocol description). SSL/TLS makes a strong distinction between a Client and a Server. Unlike a protocol like IPsec where each endpoint is a peer, SSL/TLS has specific roles for each endpoint. The endpoint initiating the SSL/TLS connection, like a web browser to a secure shopping site, is the client. The endpoint responding to the connection request is the server. There are two primary phases in an SSL/TLS connection: The handshake and then the data transfer. The handshake messages get everything started. We can see the start of them if Figure 23.

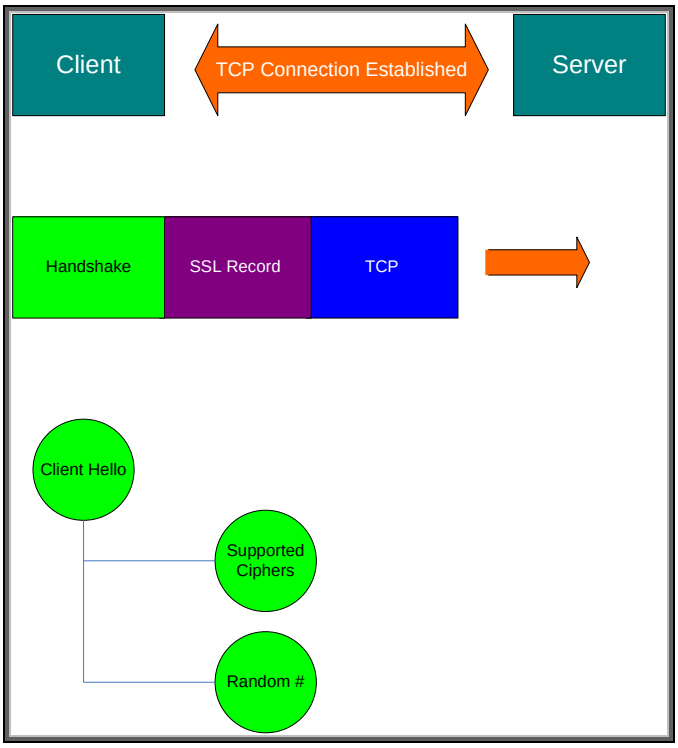


Figure 23 -Client Hello

Here we already have a TCP connection in place. The TCP connection was initiated by the client. Once we have this reliability, the client now sends the SSL Client Hello message to the server. This message has a random number and a list of cipher suites the client supports. Now it is the server's turn in Figure 24 – Server Hello.

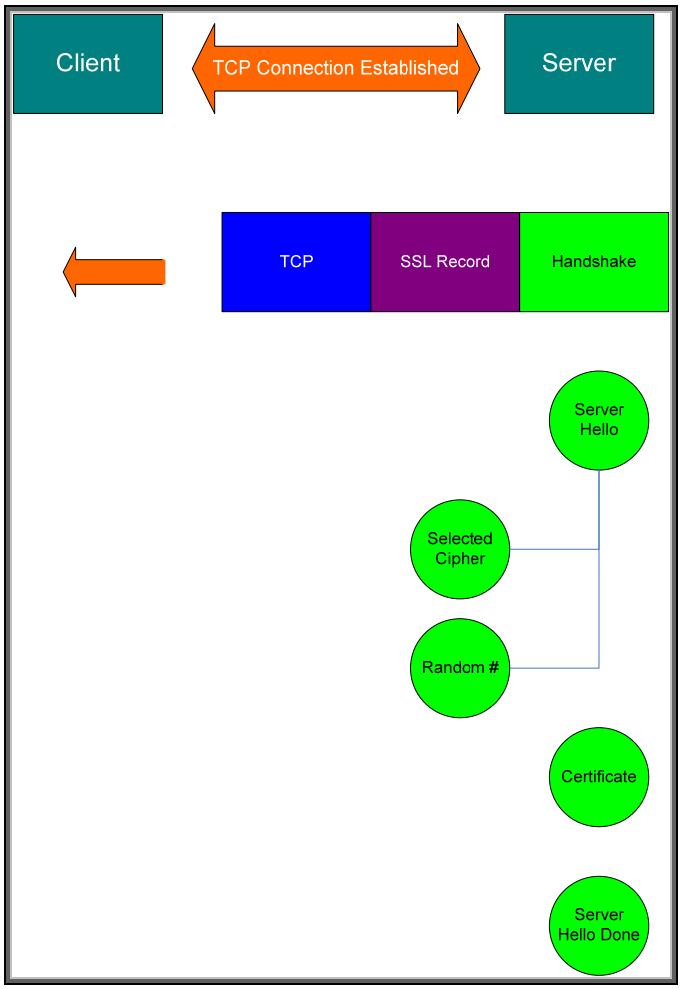


Figure 24 - Server Hello

The server responds with a Server Hello message which includes another random number and the server selected cipher. It also sends back its public key certificate along with a message indicating that it is done with this part of the handshake. Now, the client has some work to do.

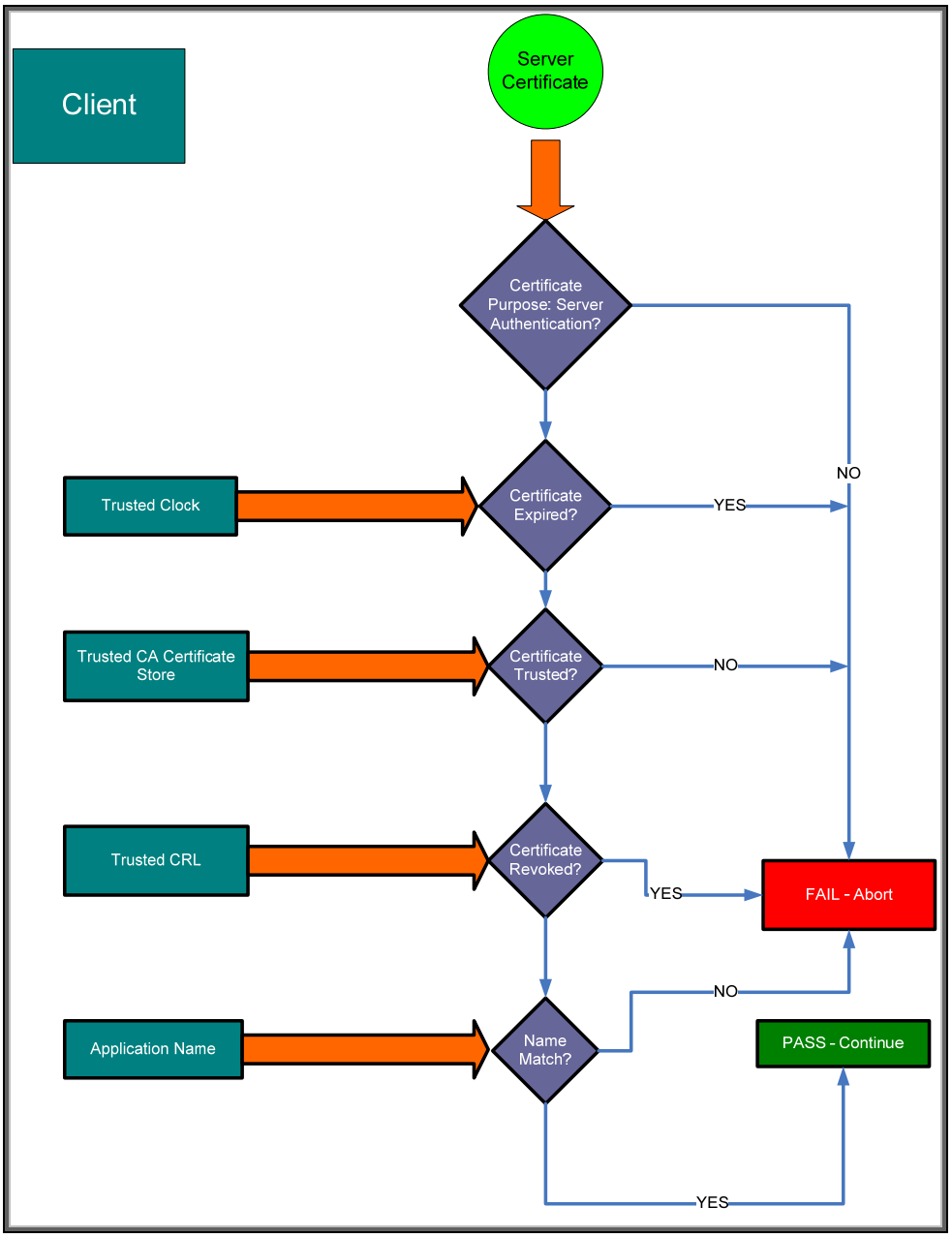


Figure 25 - Server Certificate Verification

Here the client needs to verify the server is really who they say they are. There are a lot of checks against the certificate. If any of these checks fail, there is a good chance the client is not talking to the "real" server.

Assuming that everything is fine, the client still has more work to do. It needs to come up with some keying material.

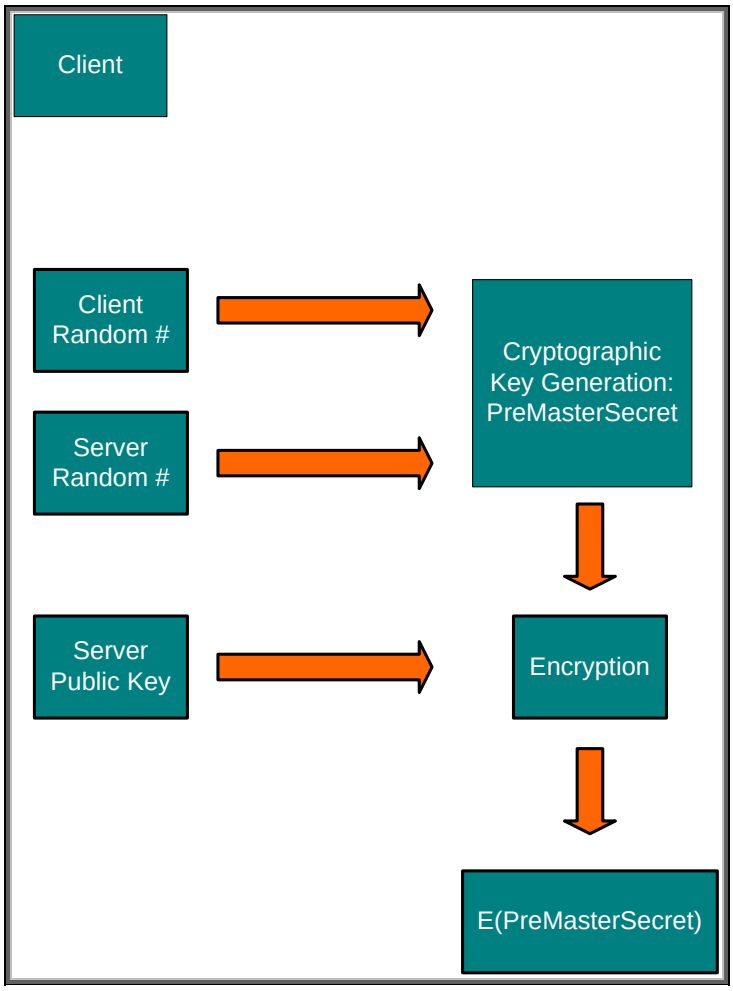


Figure 26 - Keying Material

The client generates what is called a "pre_master_secret" using the random numbers as well as a function called the key derivation function. This is encrypted with the server's public key. Only a server with knowledge of the private key would be able to decrypt it. The ability to decrypt the pre_master_secret proves that the server is in possession of the private key – the final proof for the server's identity.

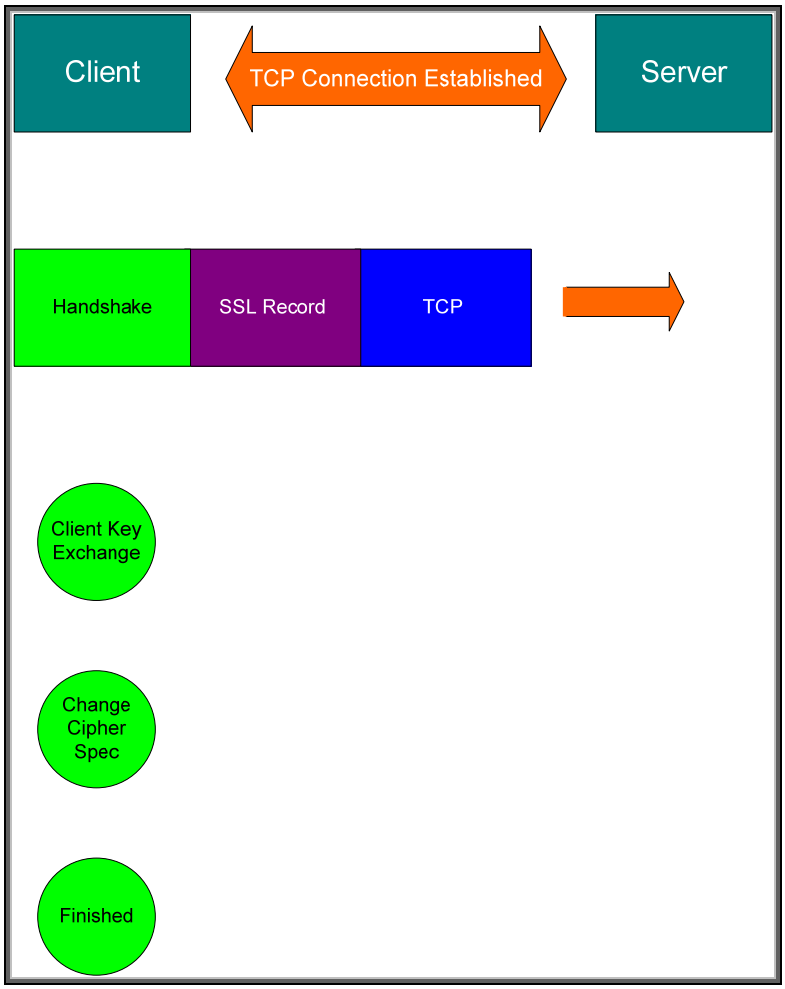


Figure 27 – Client Finished

The client goes ahead and sends over the encrypted pre_master_secret and let's the server know that it is changing over to use the master_secret and proves that it knows the master secret by providing a cryptographic hash of all data sent over to the server.

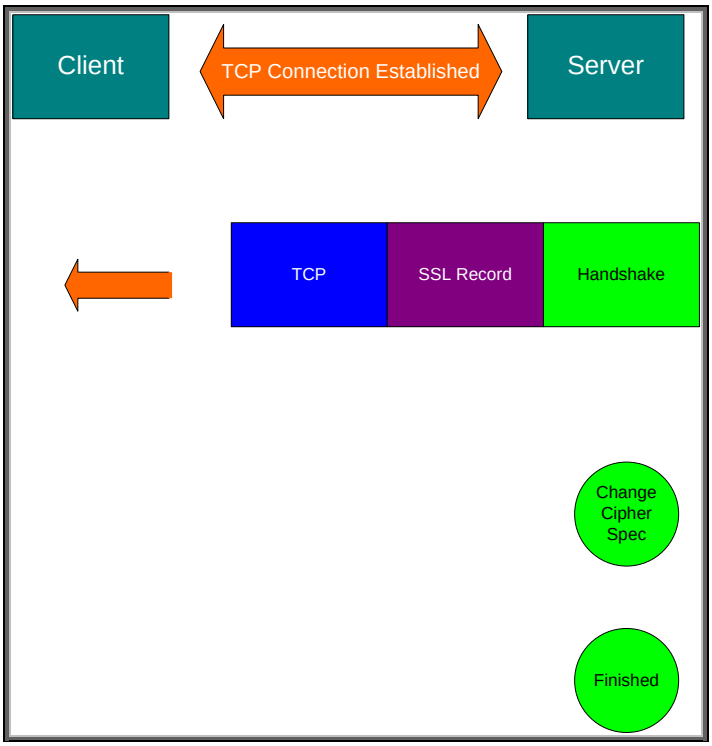


Figure 28 - Server Finished

The server decrypts the pre_master_secret and generates the master_secret. It goes ahead and let's the client know that it is changing over to use the master_secret and proves that it knows the master secret by providing a cryptographic hash of all data sent over to the client.

Once the client and server both verify the cryptographic hashes, the handshake process is done and actual client data can be sent over the SSL/TLS connection.

Let's see how SSL/TLS works in its most popular form: HTTPS.

Using HTTPS with HP Jetdirect

Before we begin, we need a little info on the setup. We have a RootCA with a subordinate CA called R2. The subordinate CA issues certificates to clients on the network. Refer to Figure 29 – CA Hierarchy.

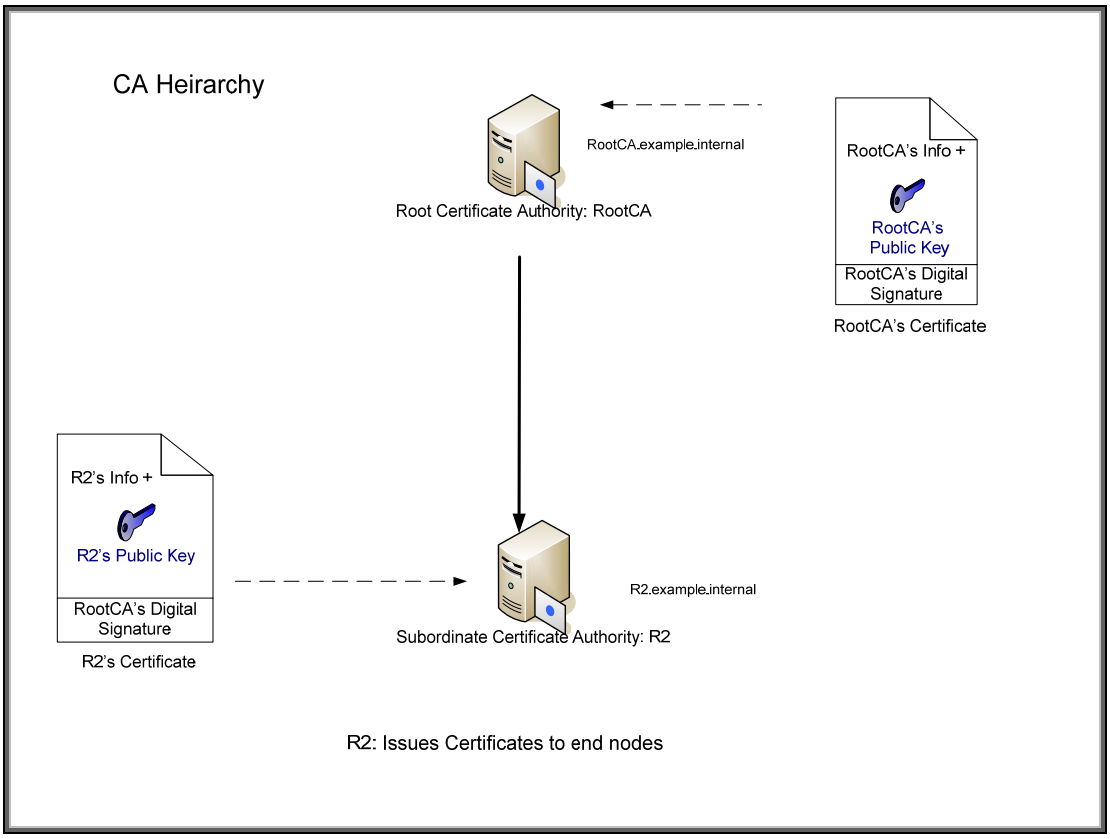


Figure 29 - CA Heirarchy

The network is really simple and is composed of these CAs, a DNS server, a client, and an HP LaserJet MFP. Refer to Figure 30 – Network Diagram.

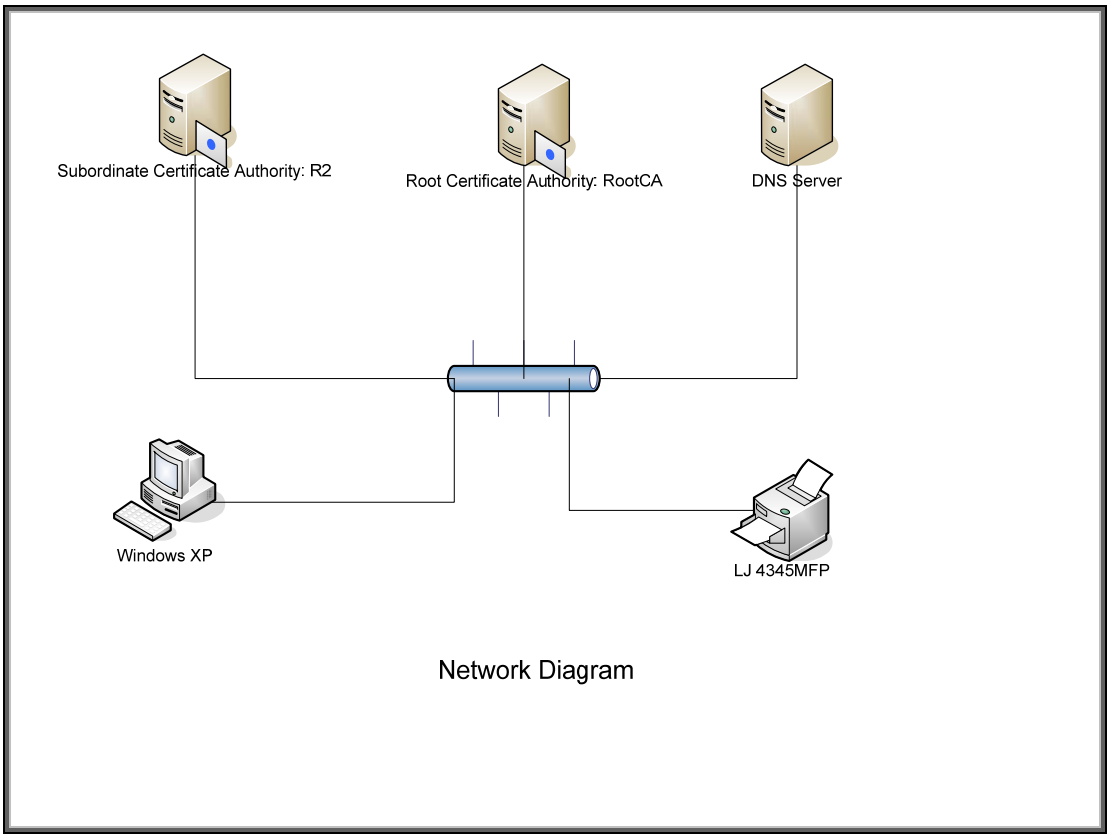
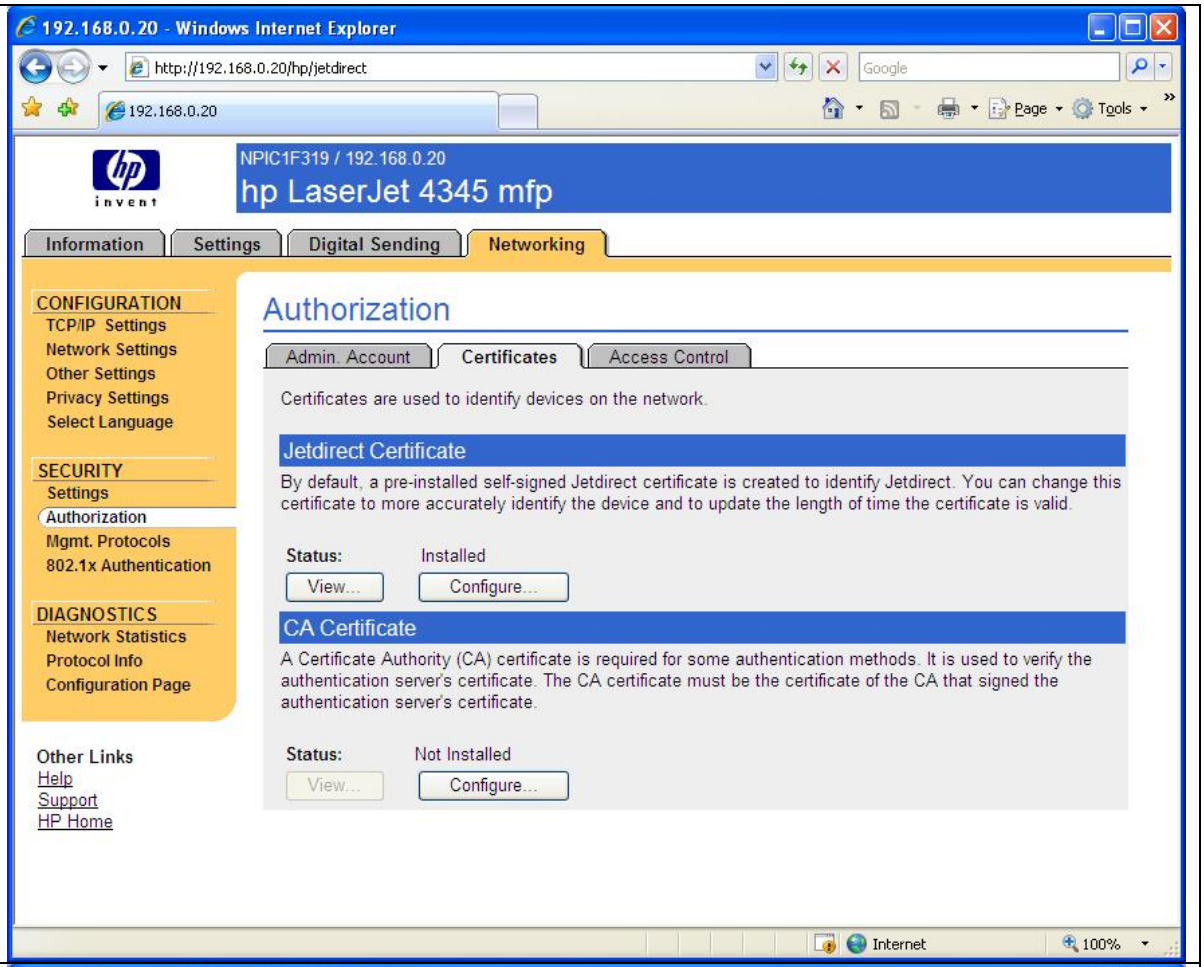


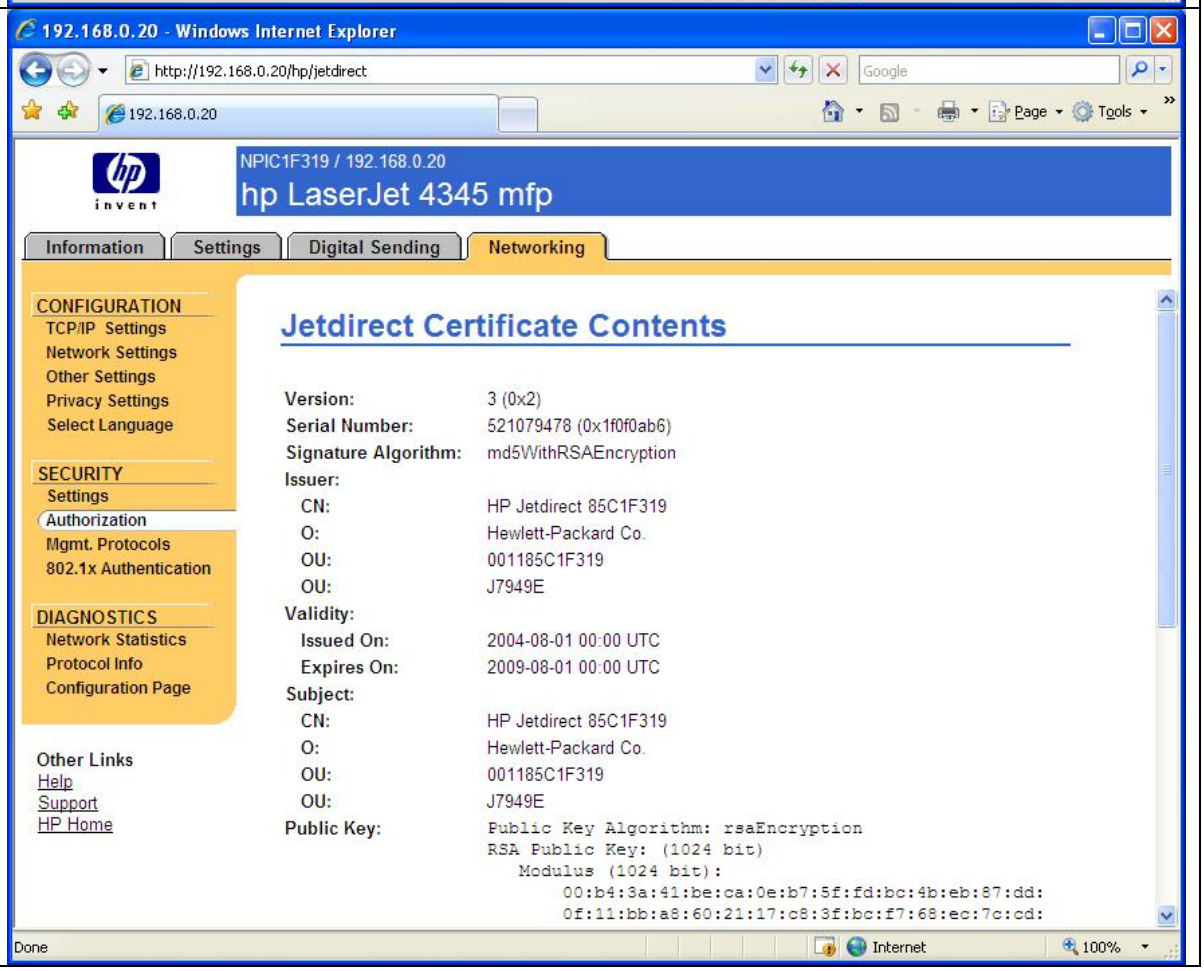
Figure 30 - Network Diagram

A pretty basic setup! The XP client is going to open a browser and talk to the 4345MFP. In short, the XP machine will be an SSL client and the 4345MFP will be an SSL server. In order to get SSL working properly, we are going to need to assign a certificate to the 4345MFP so that it can verify its identity correctly and pass all those checks that the client has to do. We'll use regular HTTP and go to the Jetdirect page where we can perform our certificate operations.

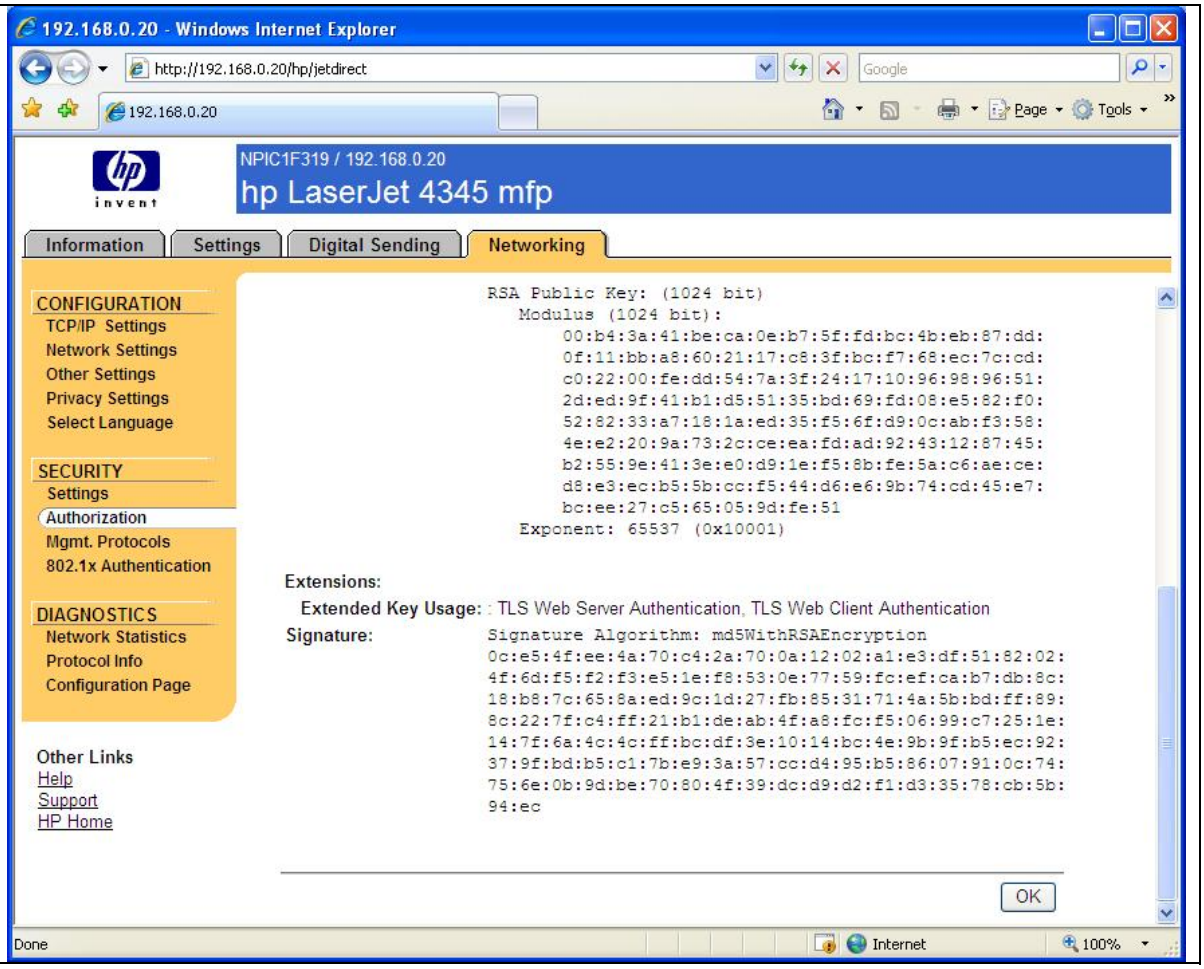
Every Jetdirect will create a self-signed certificate the first time it is powered on. Each Jetdirect has a unique self-signed certificate. For small environments, trusting the self-signed certificate (by storing the certificate on the client) may be all that is needed for security. We can take a look at this certificate by pressing "View…" under the heading "Jetdirect Certificate"

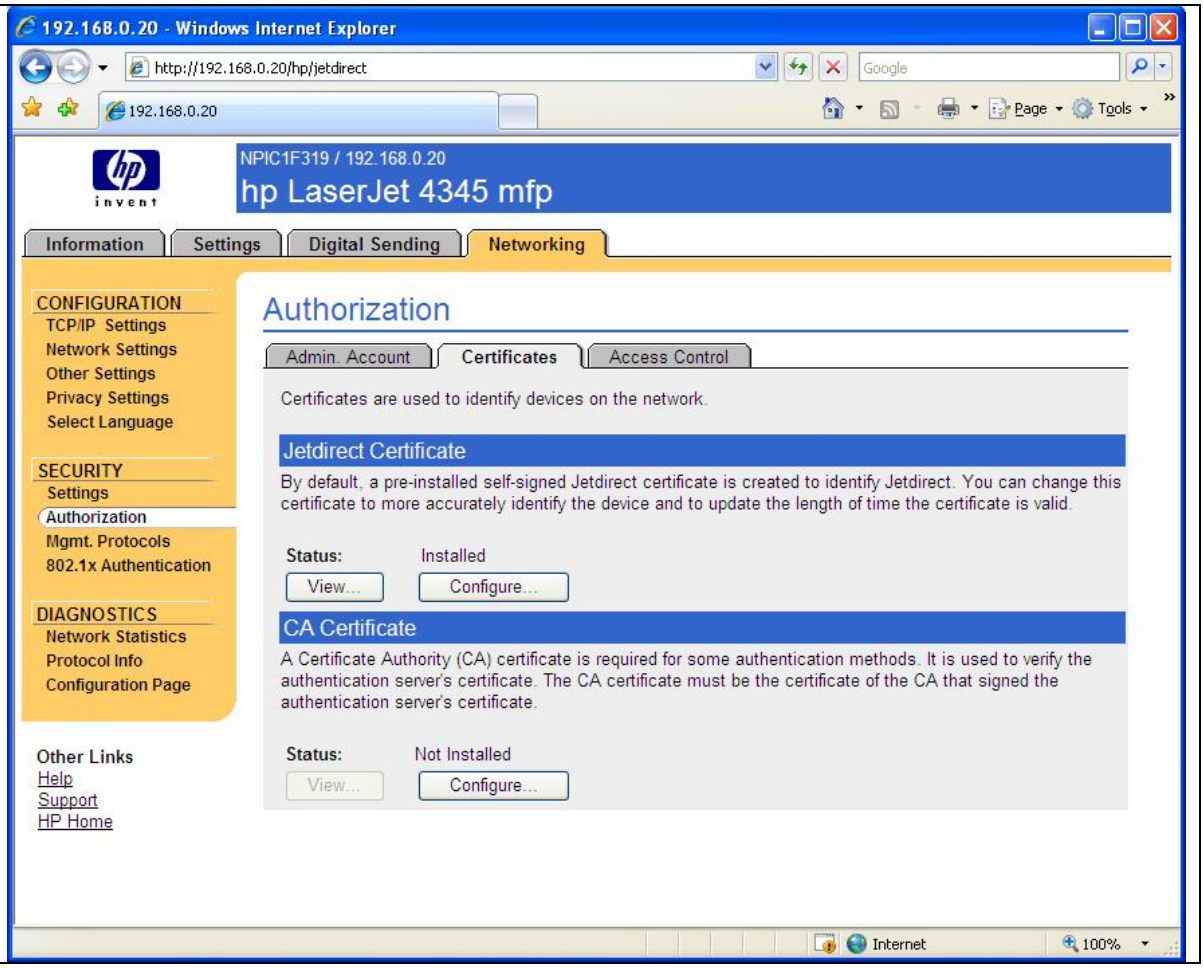


The subject and issuer names are the same – that is the first clue that it is a self-signed certificate. Because the self-signed is generated at first time power up, there are no DNS names or IP addresses associated with it.

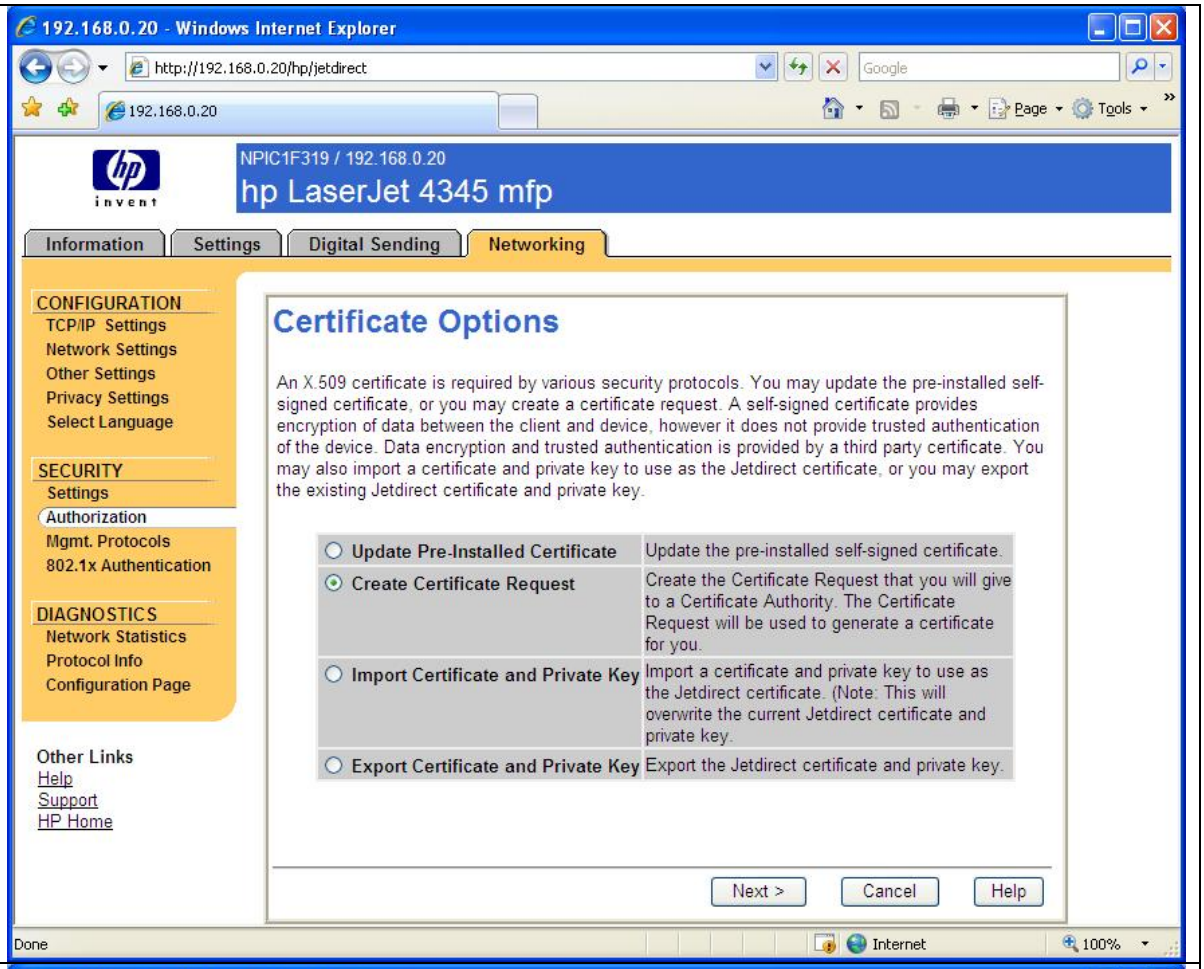


We see the RSA public key is 1024 bits for the self-signed certificate and that the certificate can be used for client and server authentication. We also see that the certificate has a signature – which means it has been signed (by itself in this case). Click OK and go back to the main screen.

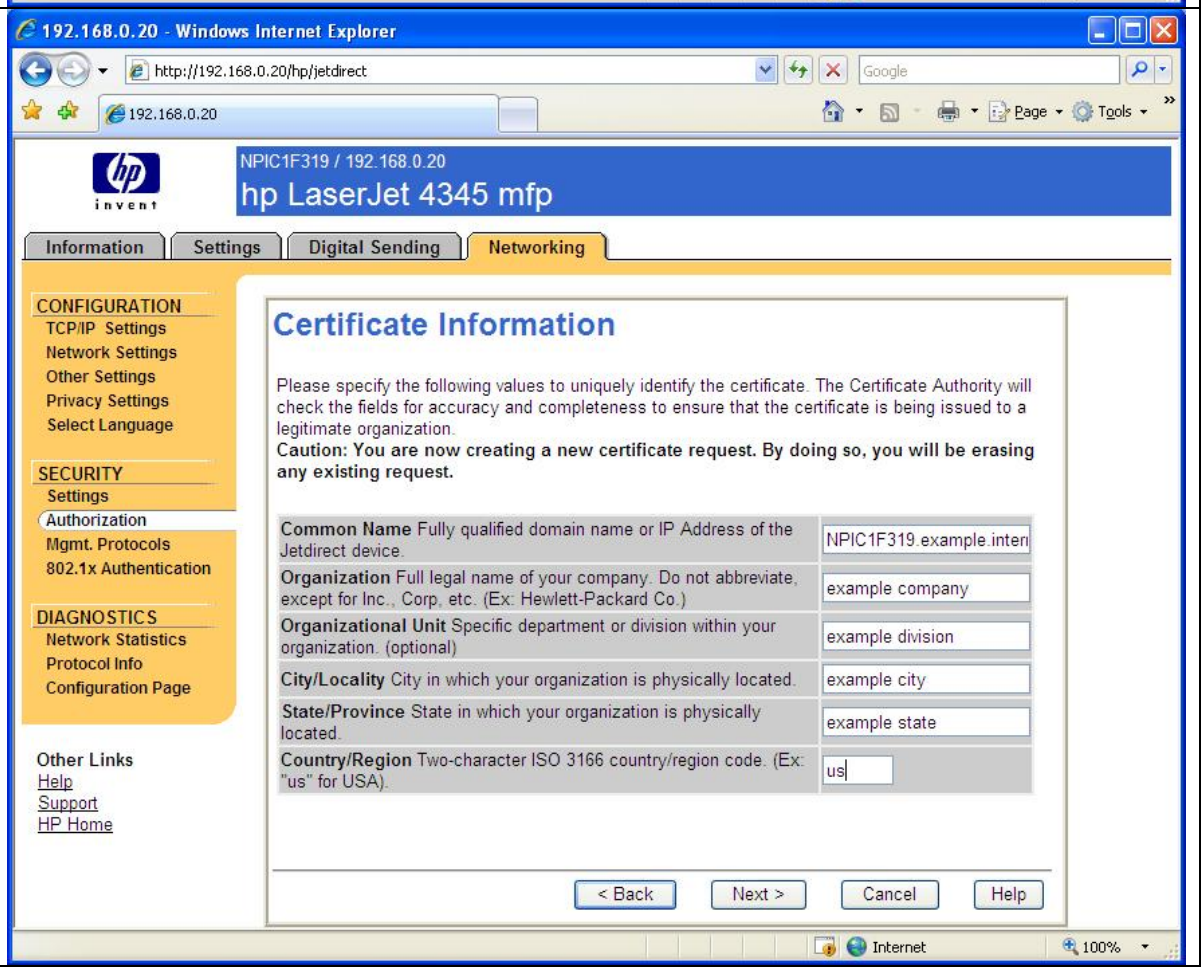


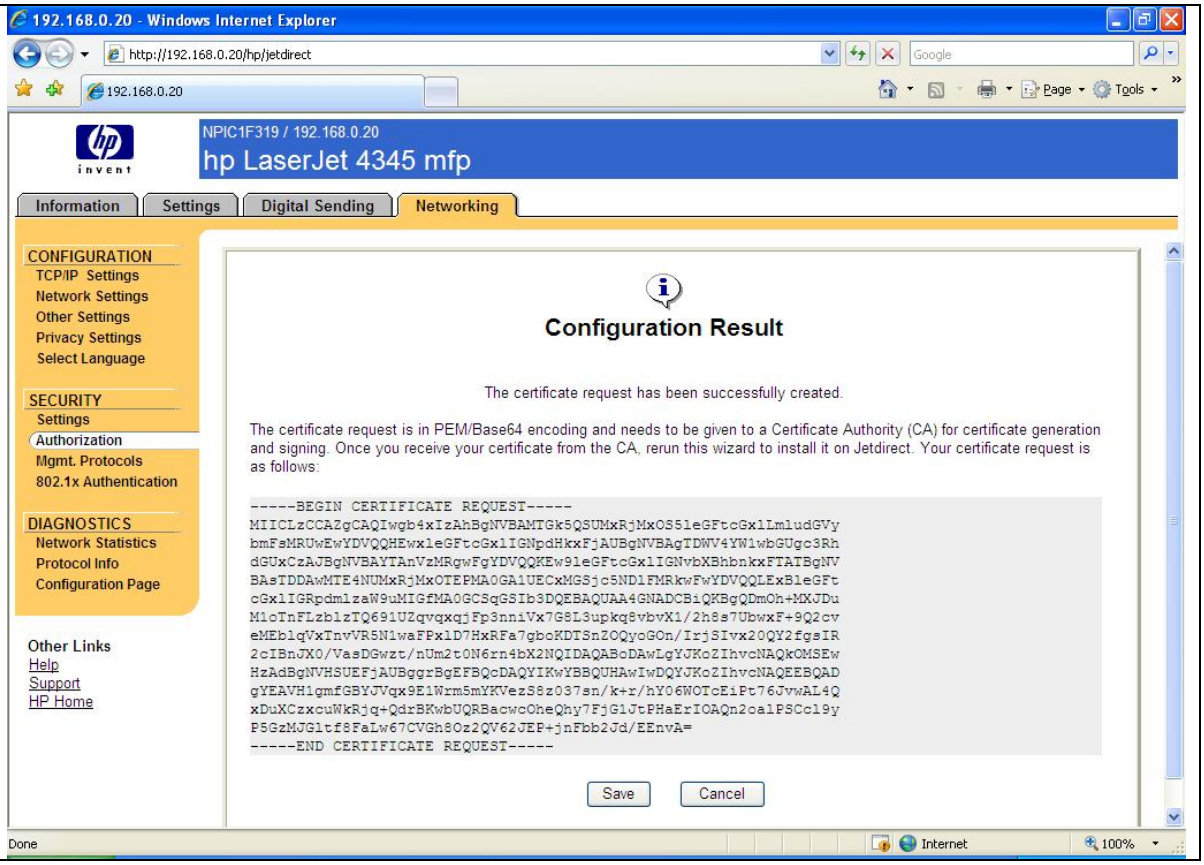
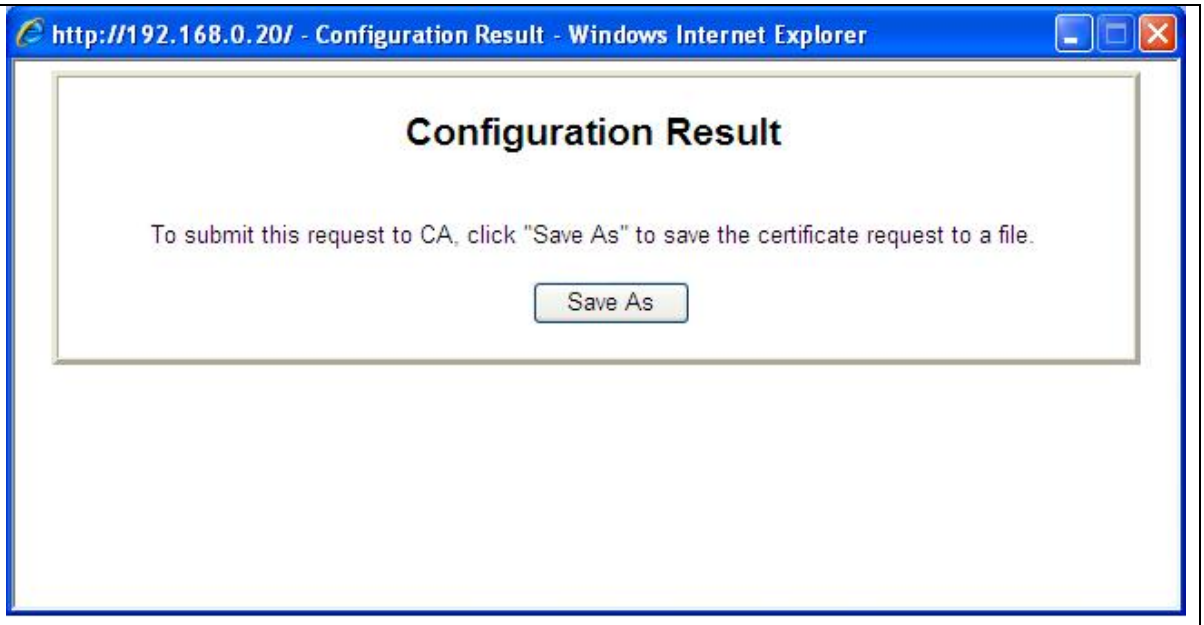
Under the heading "Jetdirect Certificate", press "Configure…"	

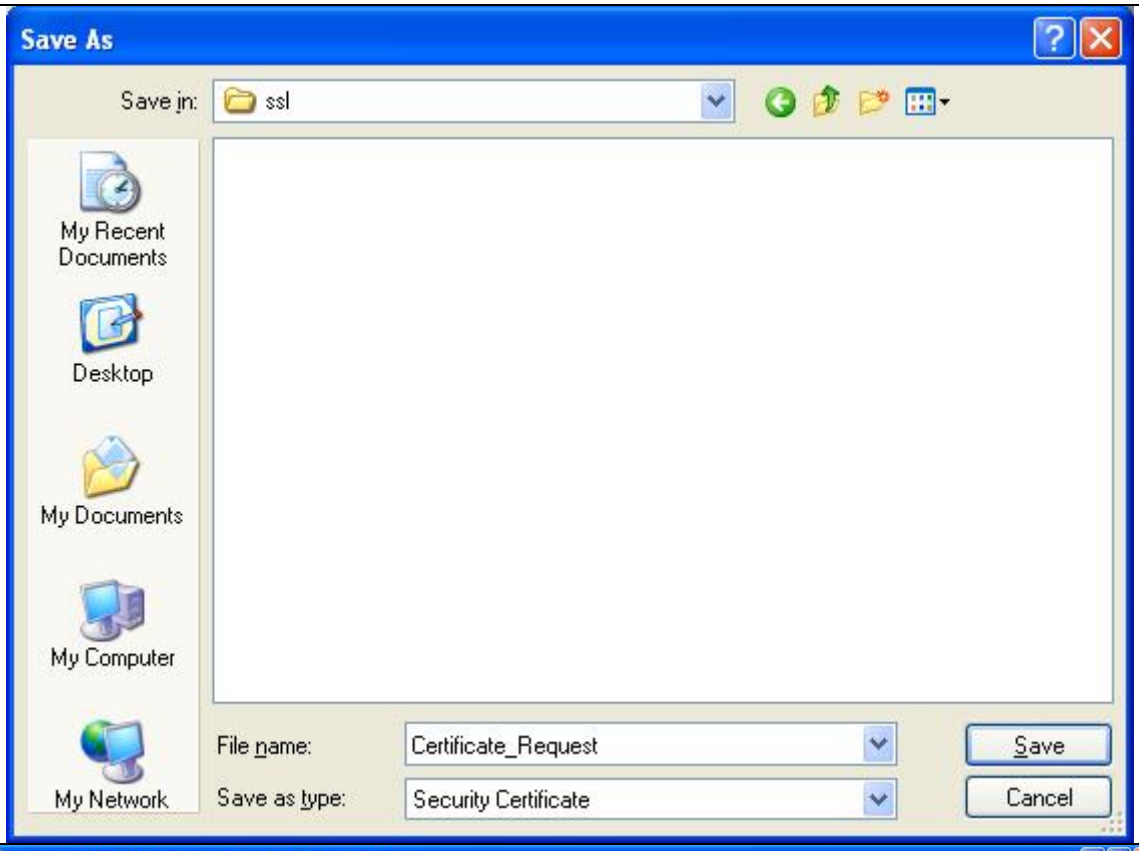
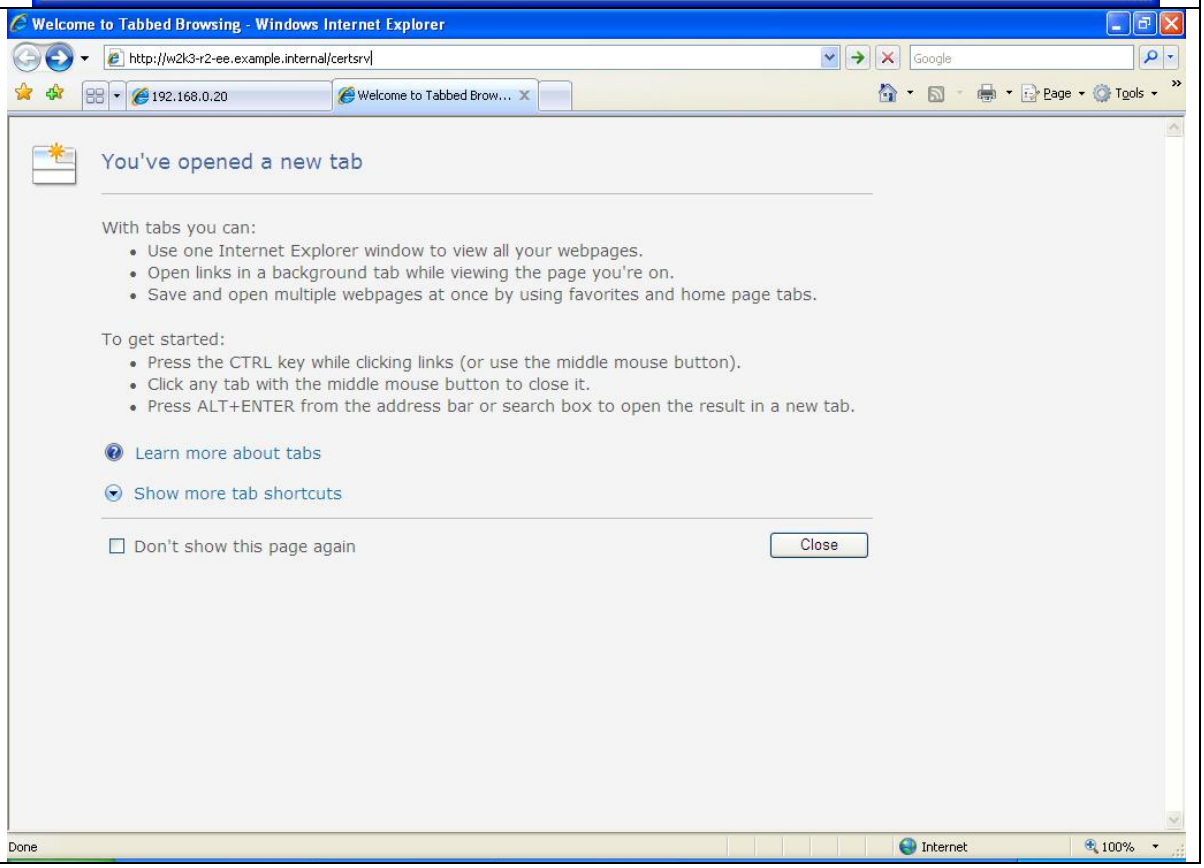
Select the radio button "Create Certificate Request". This will tell Jetdirect to create a public/private key pair and along with some more information that we be entered, generate a certificate request with the public that can be given to a CA. Jetdirect does not reveal the private key. Press "Next ->"

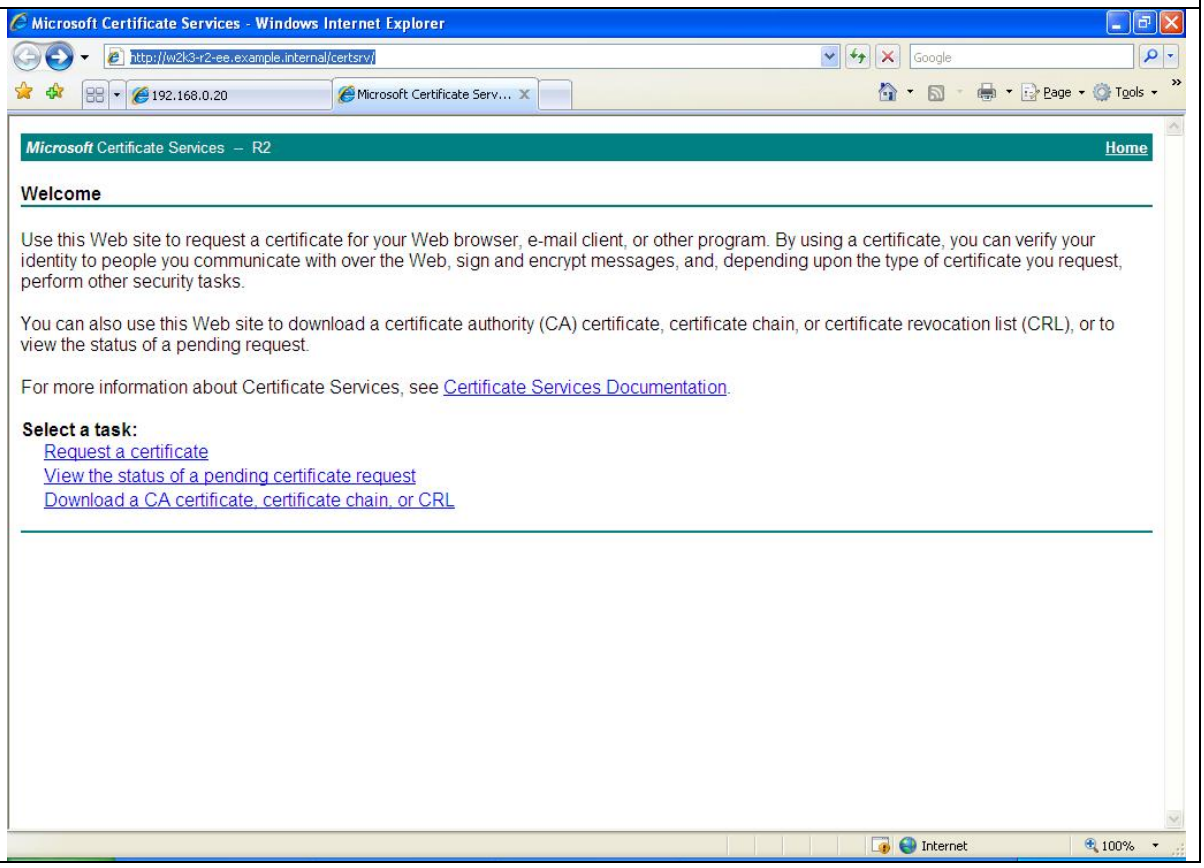


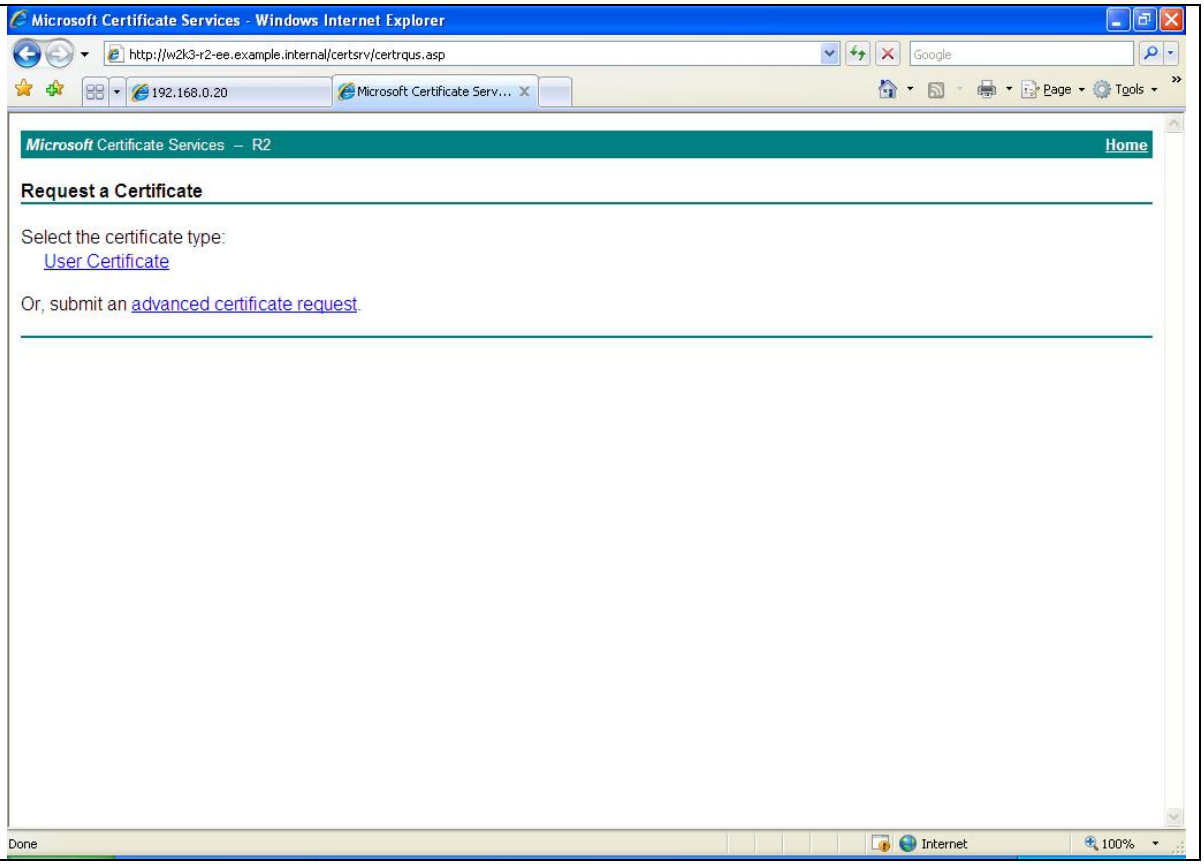
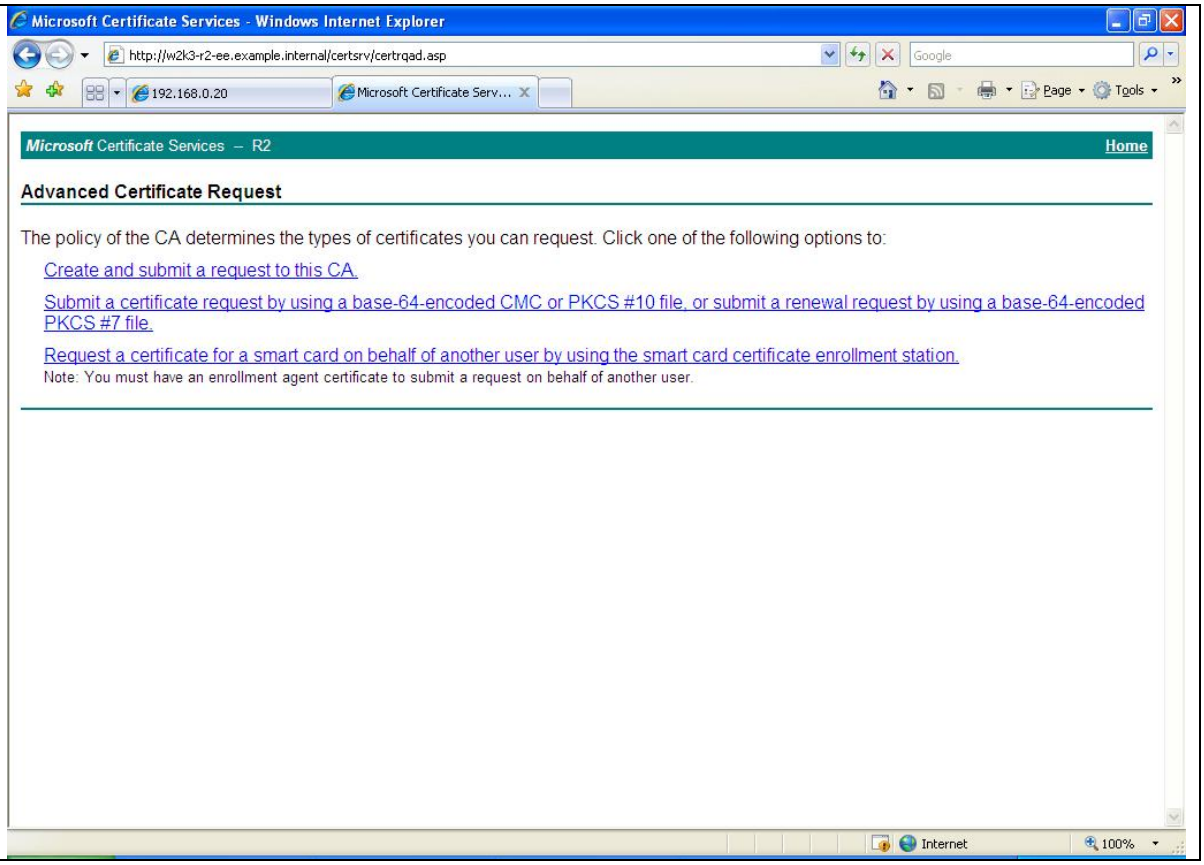
Here we enter details to properly identify the Jetdirect device. Each customer will have different values here. After entering in the values, press "Next->"



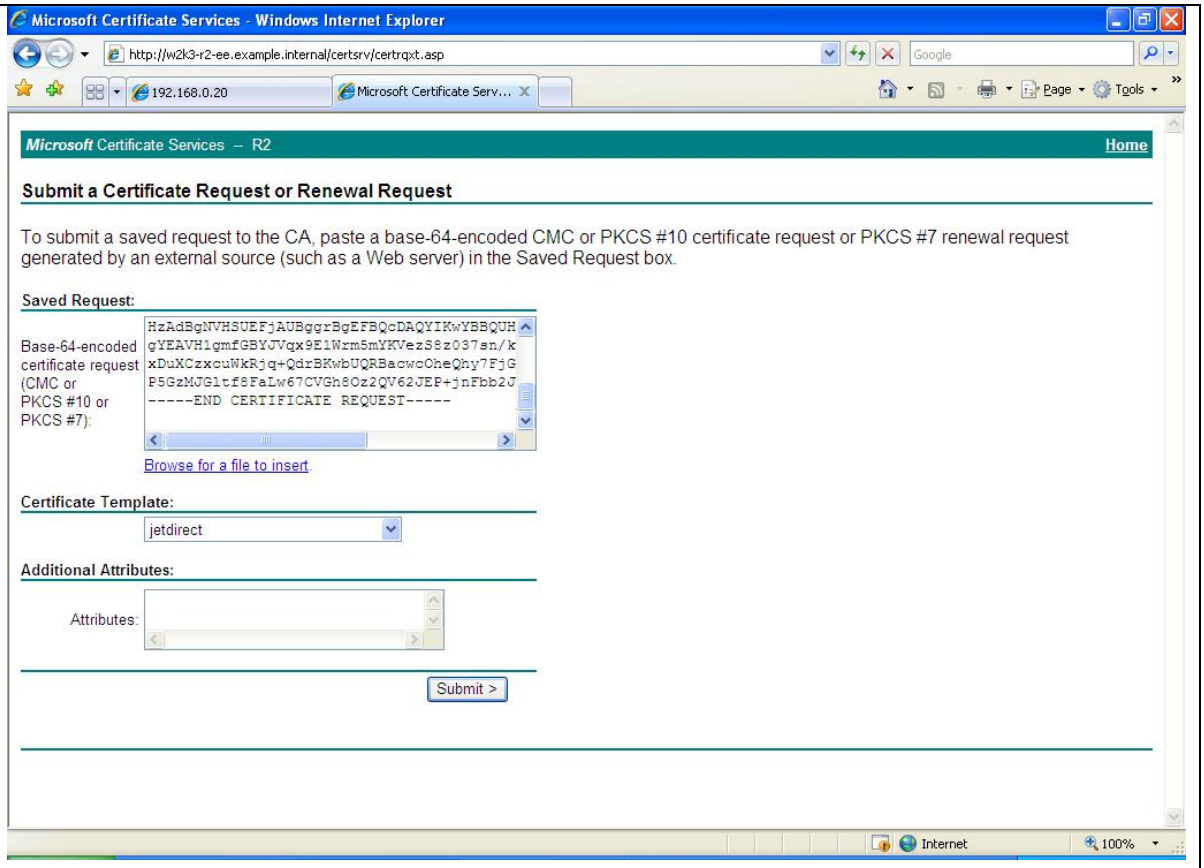
Here is the certificate request. We are going to want to store it. We can cut/paste it or click “Save”.	
Click “Save As”	

Store it in a directory on the client.	
Now we are going to bring up R2's CA web server.	

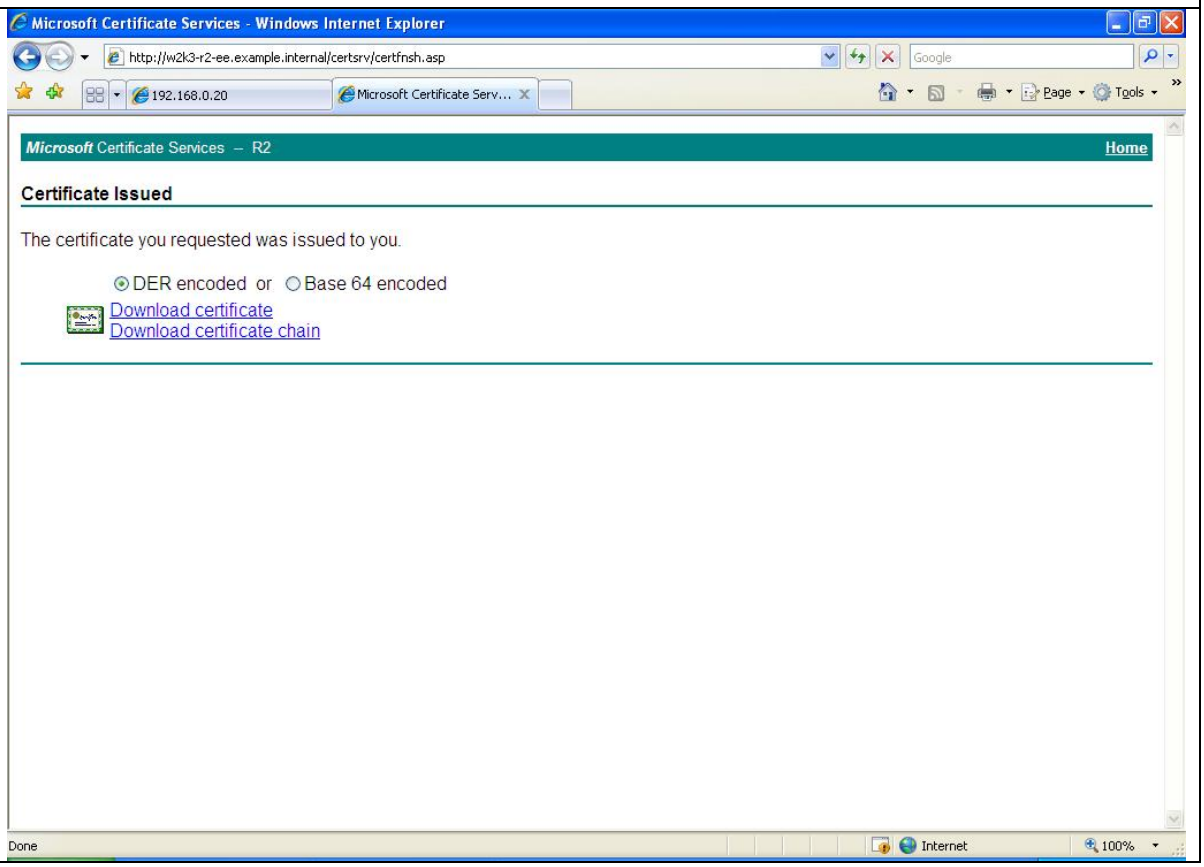
Enter the credentials that will allow a certificate to be issued.	
And here is the R2's CA web server. Let's click the link "Request a Certificate"	

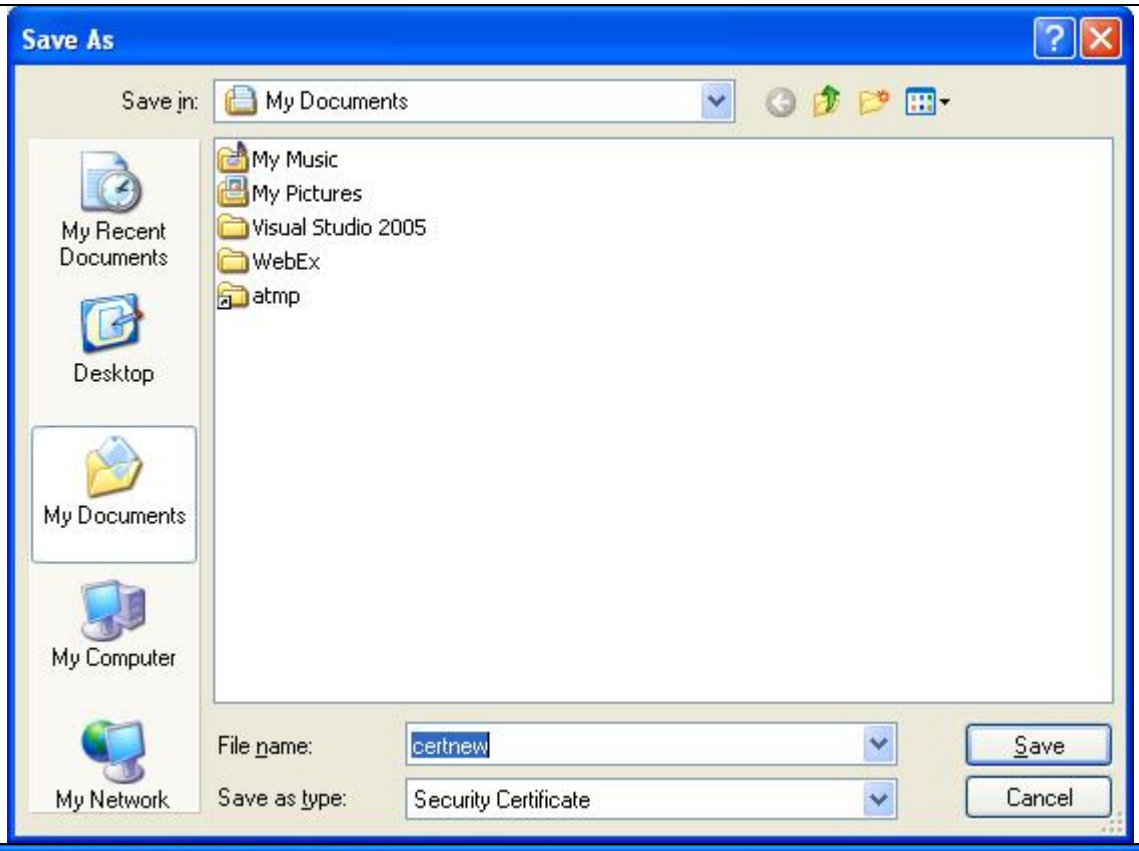
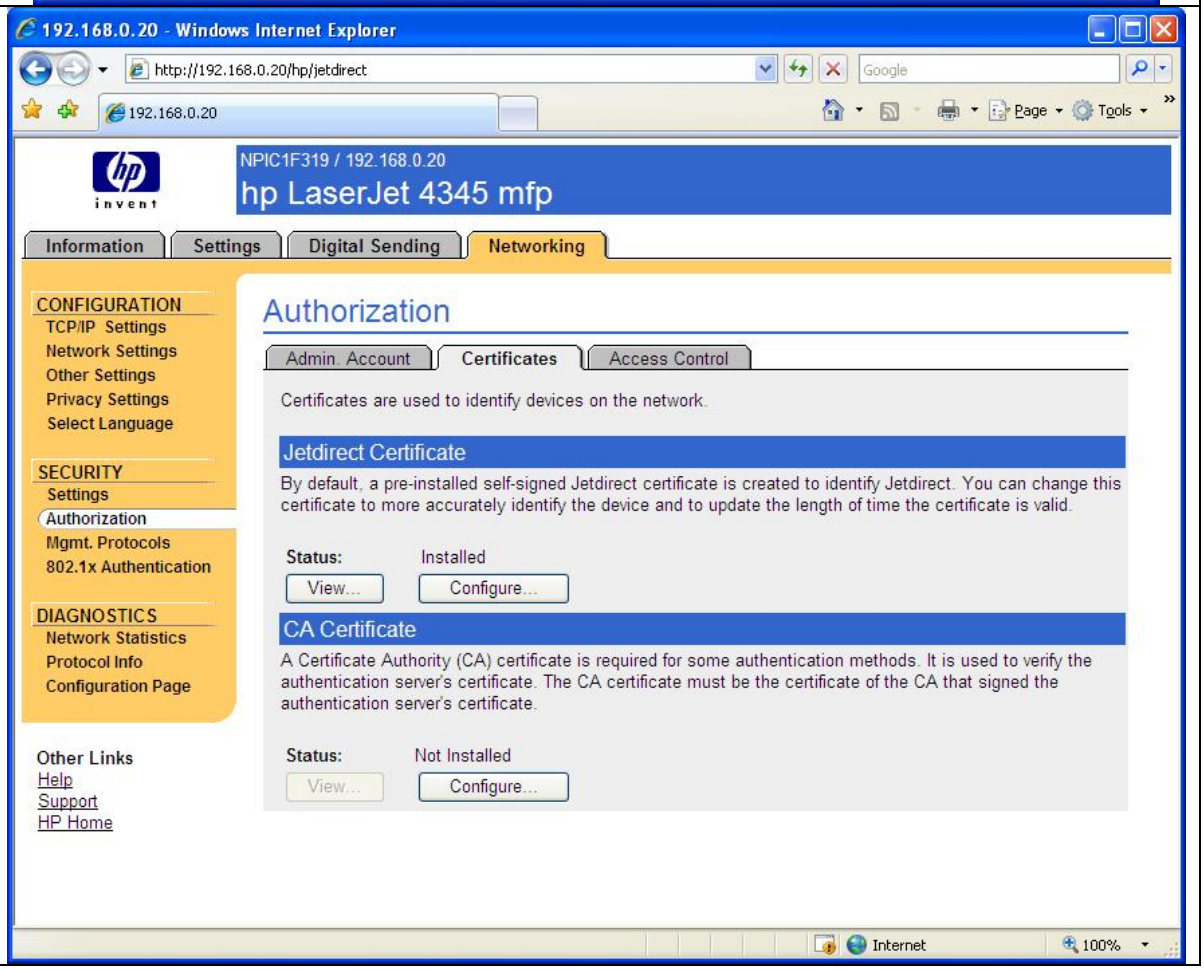
Click "advanced certificate request"	
Select the second link "Submit a certificate request…."	

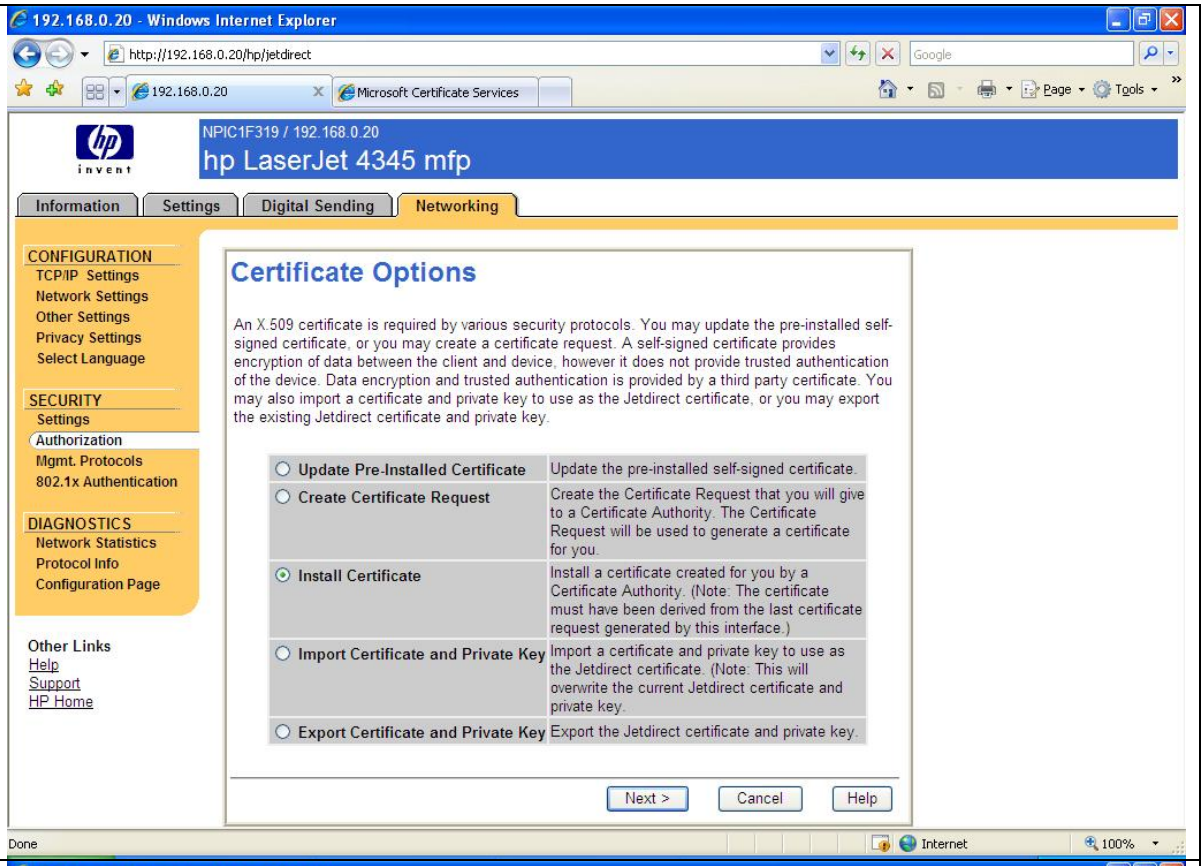
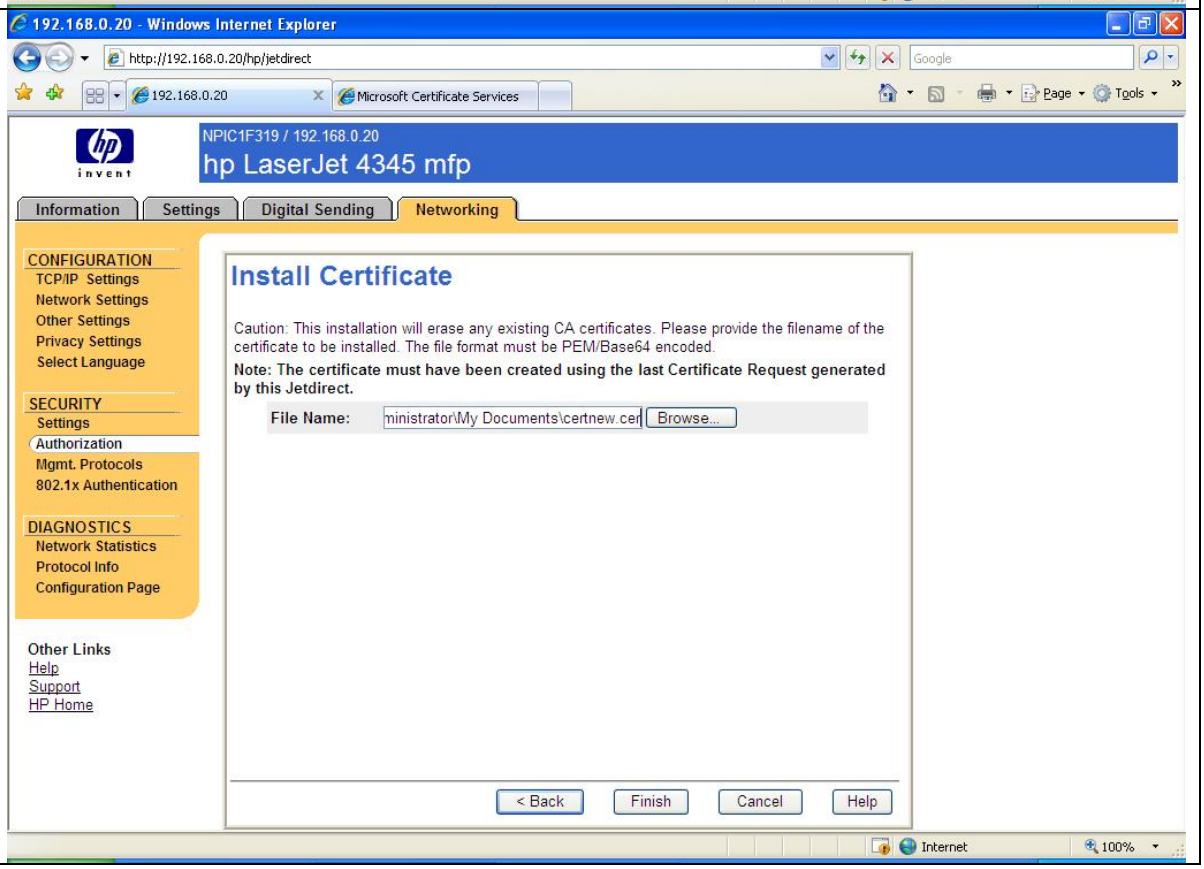
We cut and paste the certificate request from Jetdirect into the box provided. We select a certificate template. This template is basically a "cookie cutter" for how to create a specific type of certificate. We have a template called "jetdirect" which has already been created. The only thing it really specifies is that the certificate can be used for Client and Server authentication. Click "Submit".

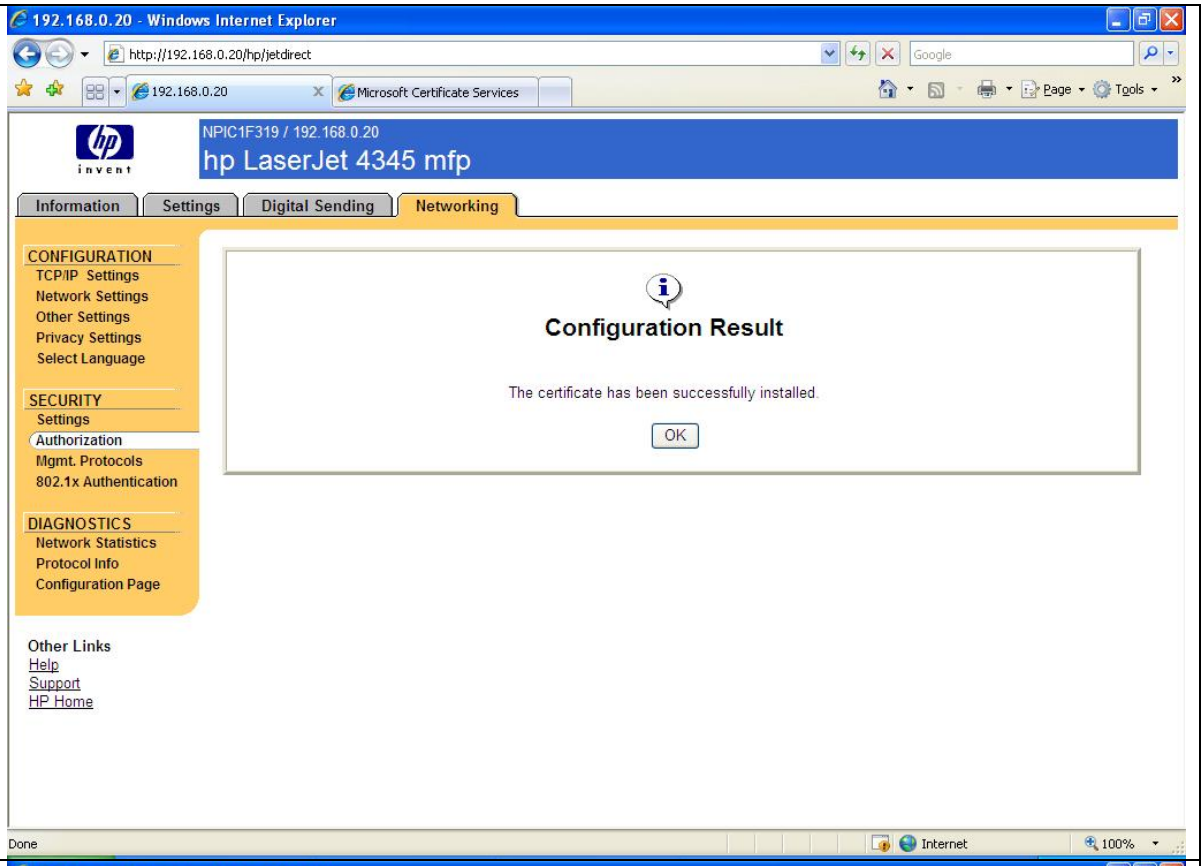
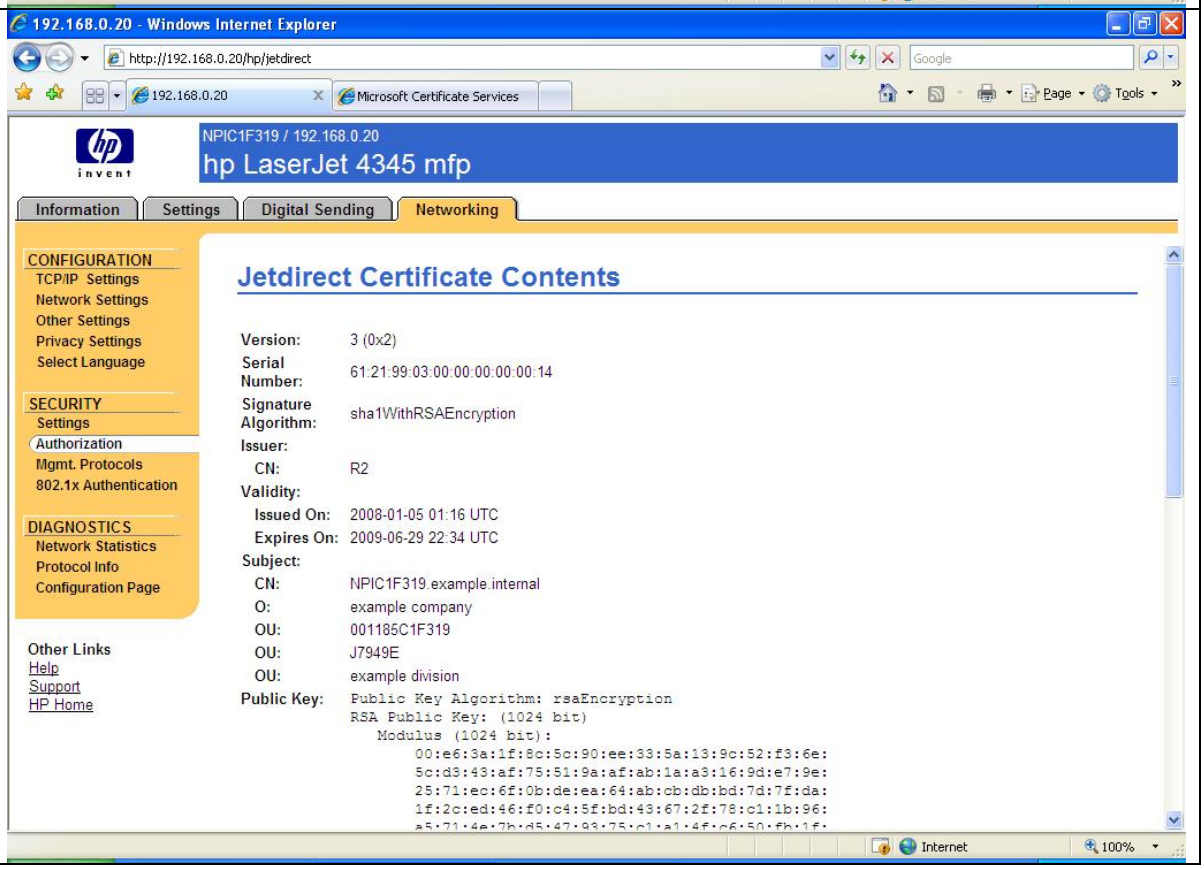


Click "Download certificate". DER encoding is fine.

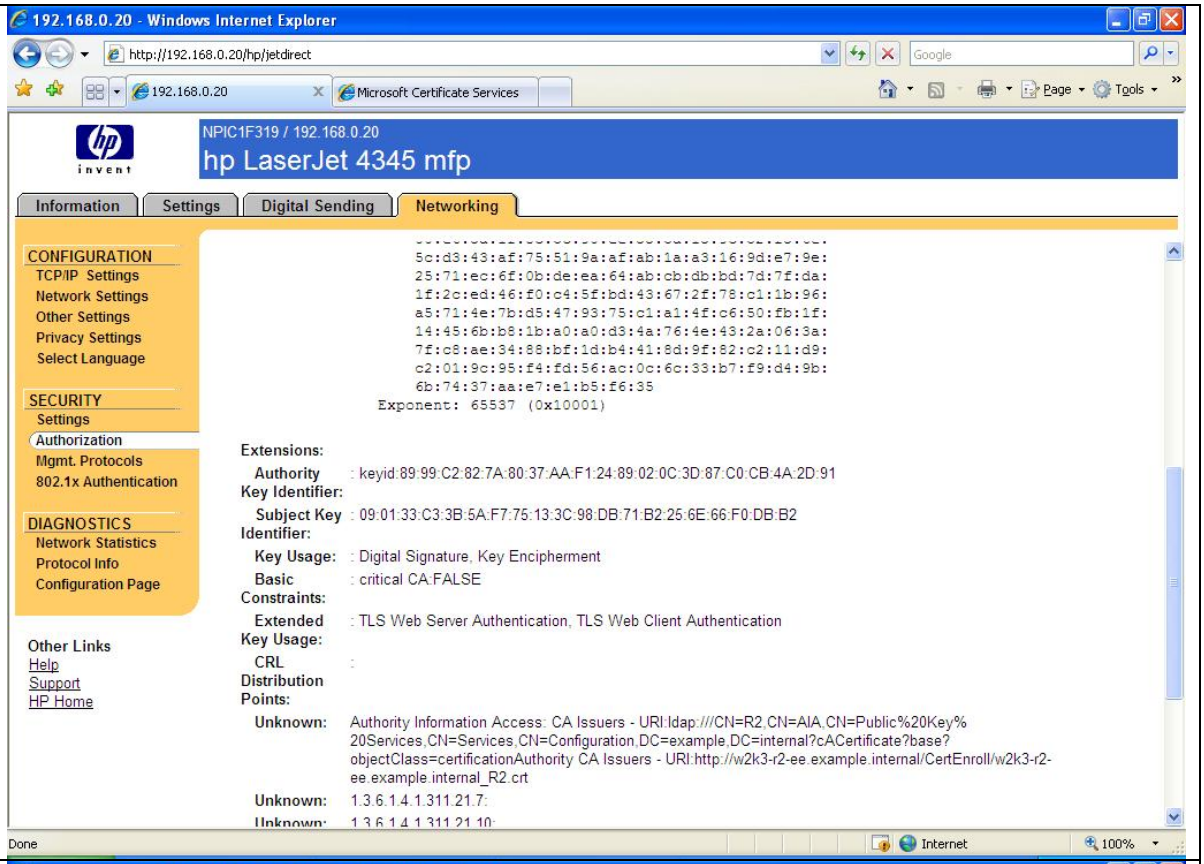


Save it.	
Bring up the certificate wizard on Jetdirect again by pressing "Configure…"	

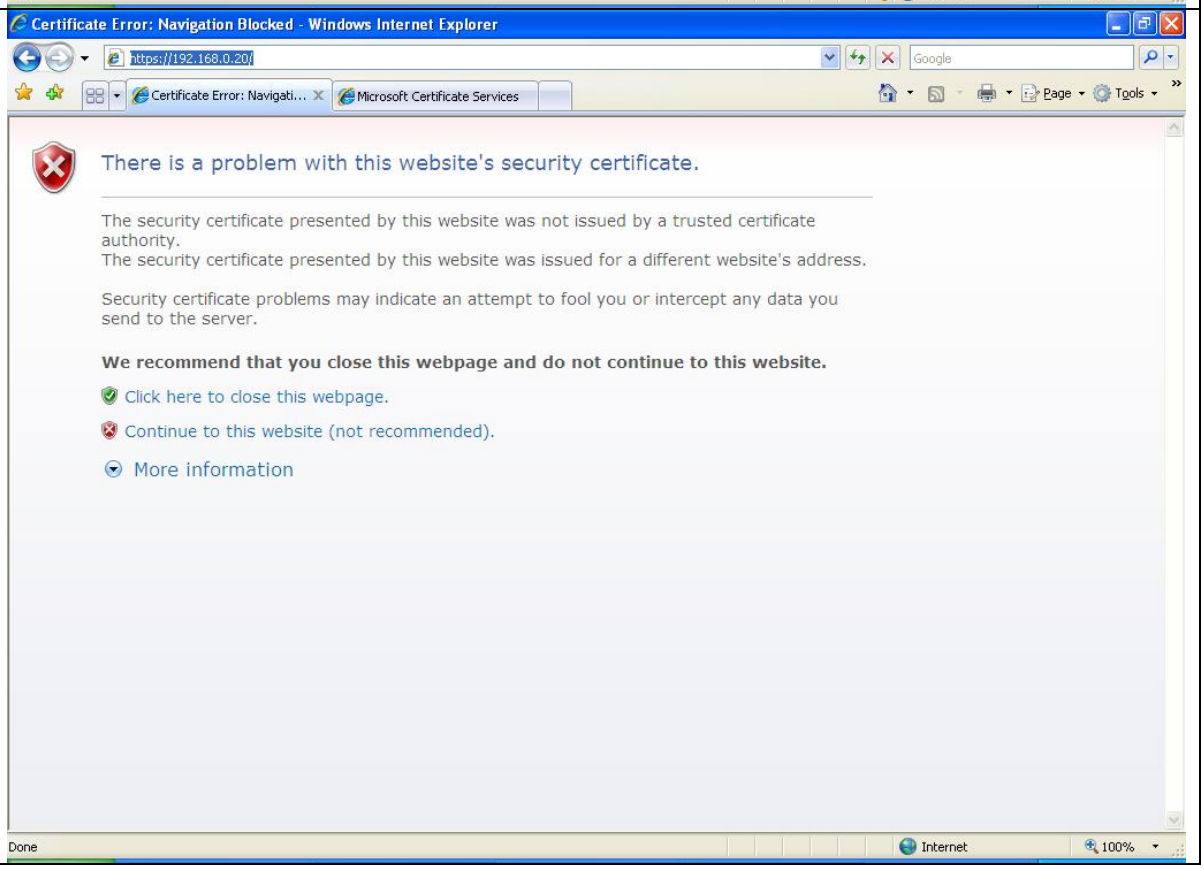
Now we select "Install Certificate" and click "Next"	
Point it to the file obtained from the R2 CA. Click "Finish"	

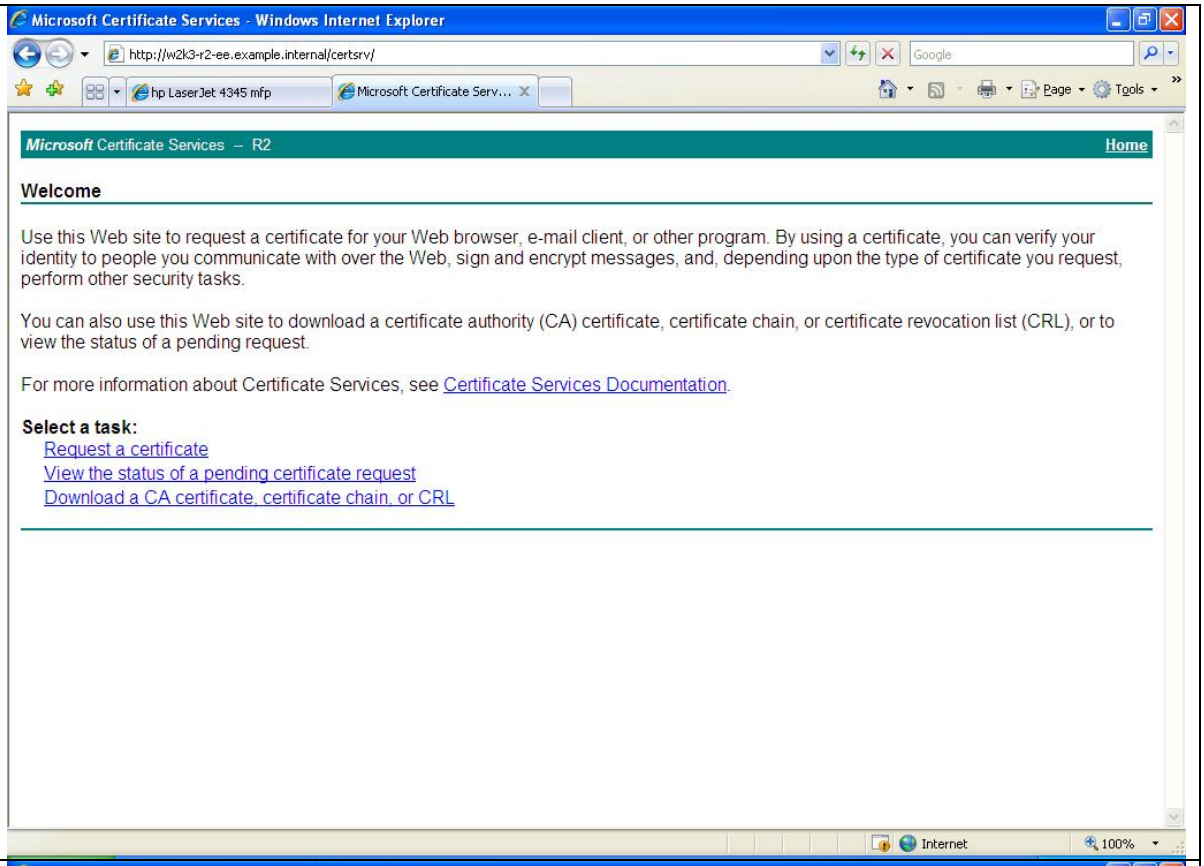
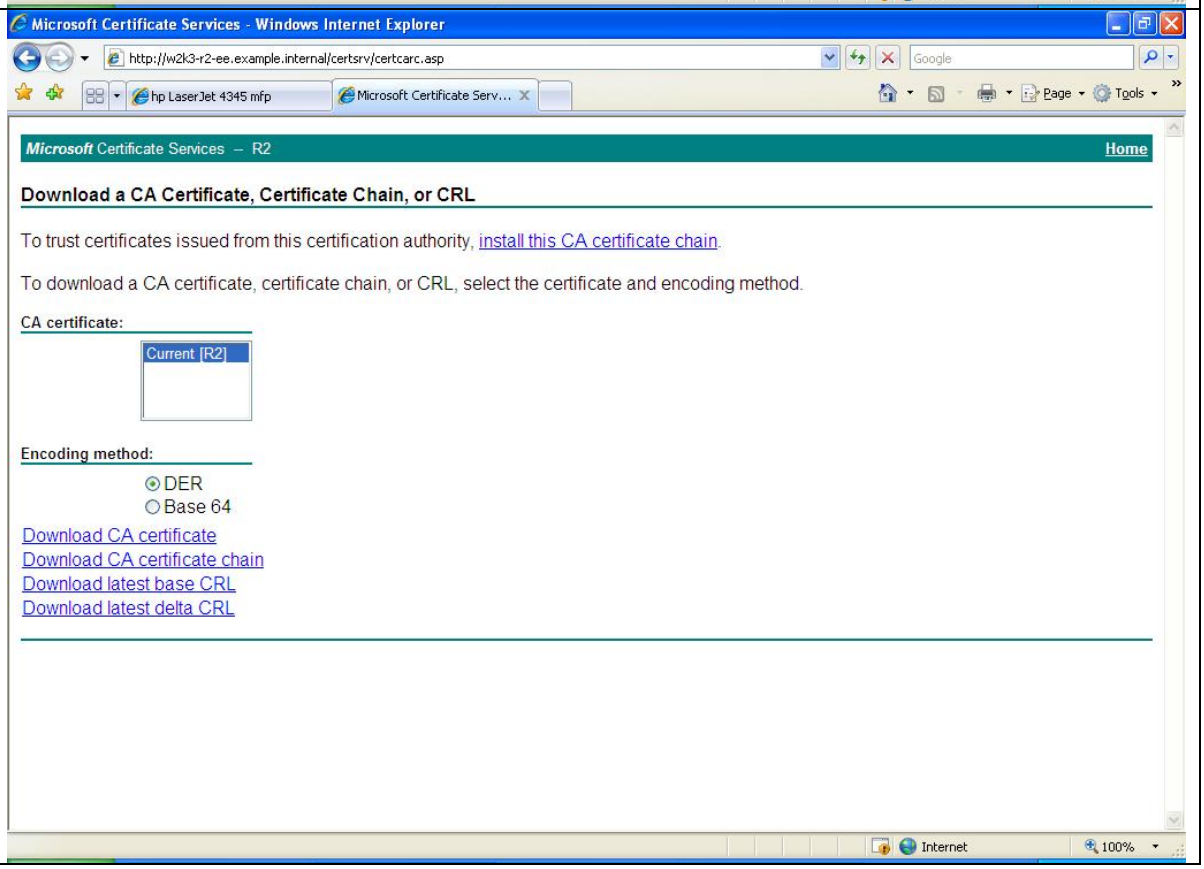
Cool – it worked. Click "OK"	
Now – let's view the contents of this certificate. We can see that the issuer is R2. We also see the Subject CN. This name will be important later on.	

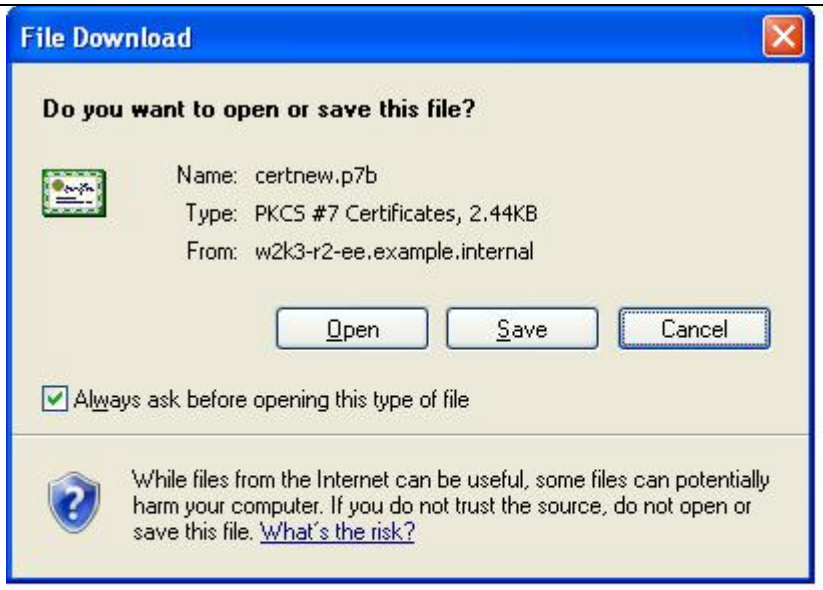
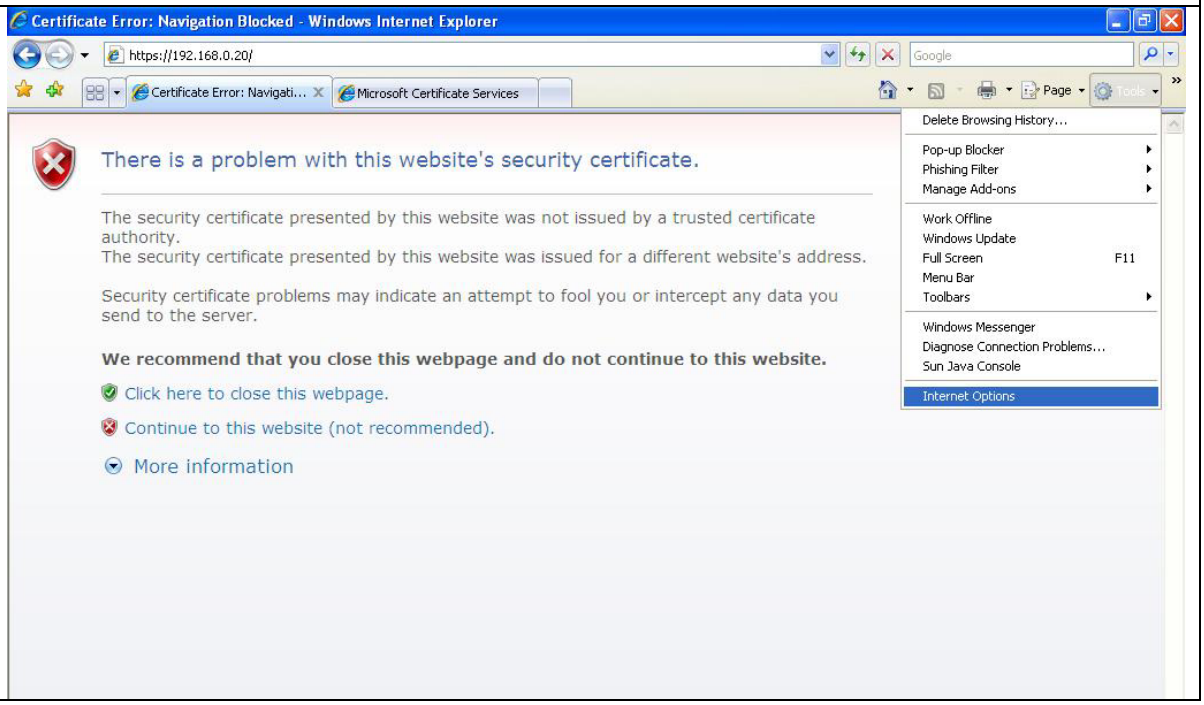
We see we have some CRL distribution points in the certificate as well – remember that. Also see that we can do Web Server and Web Client authentication.



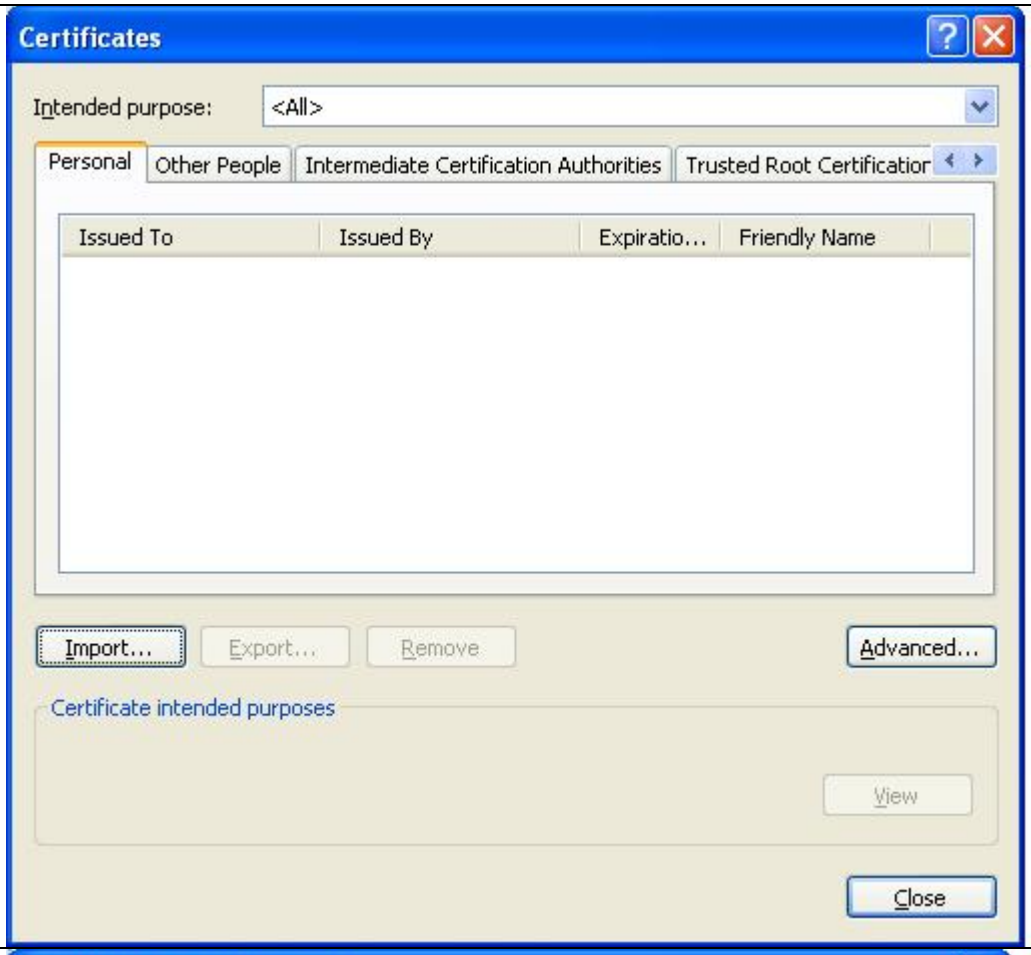
Let's use HTTPS. Everything should be fine right? Wrong! The client has failed its server certificate checks. Why? It says that the Security Certificate was not issued by a trusted certificate authority. The browser's certificate store must not know about our R2 and RootCA certificate authorities. Let's correct that. First, we need to go back to R2's CA web server.

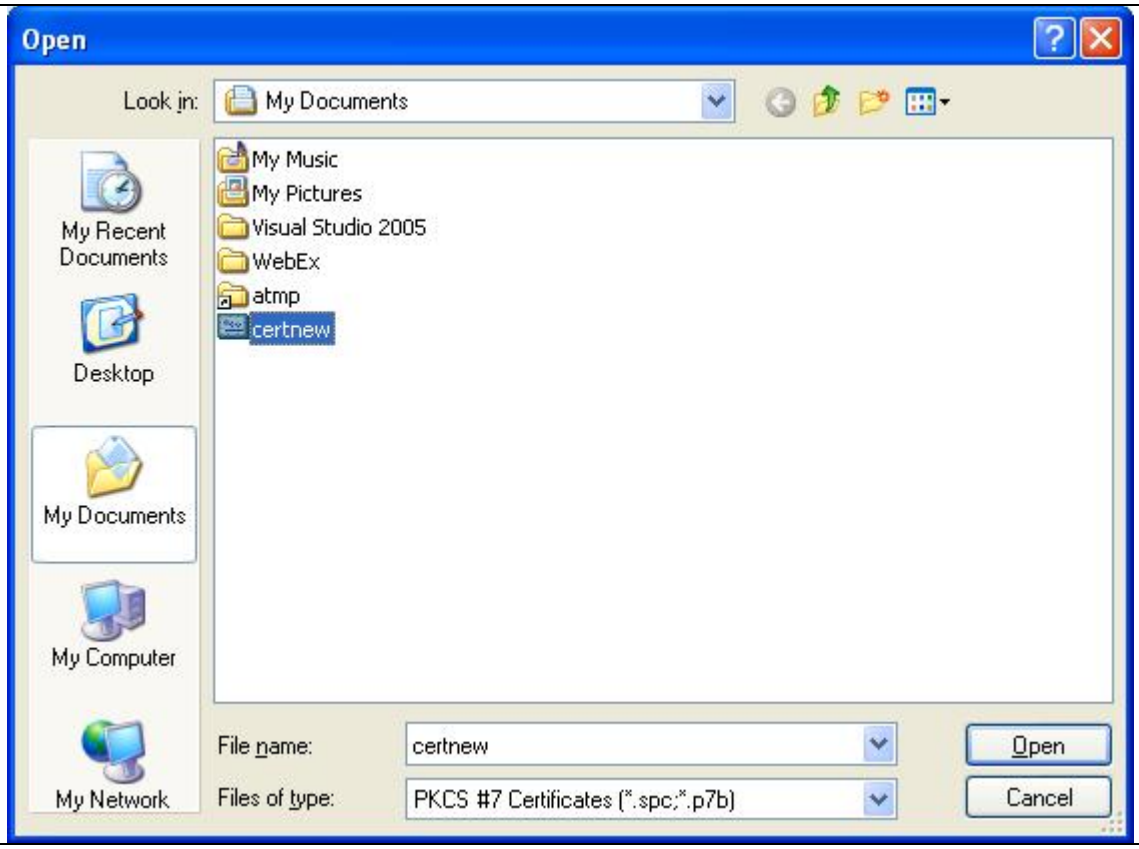
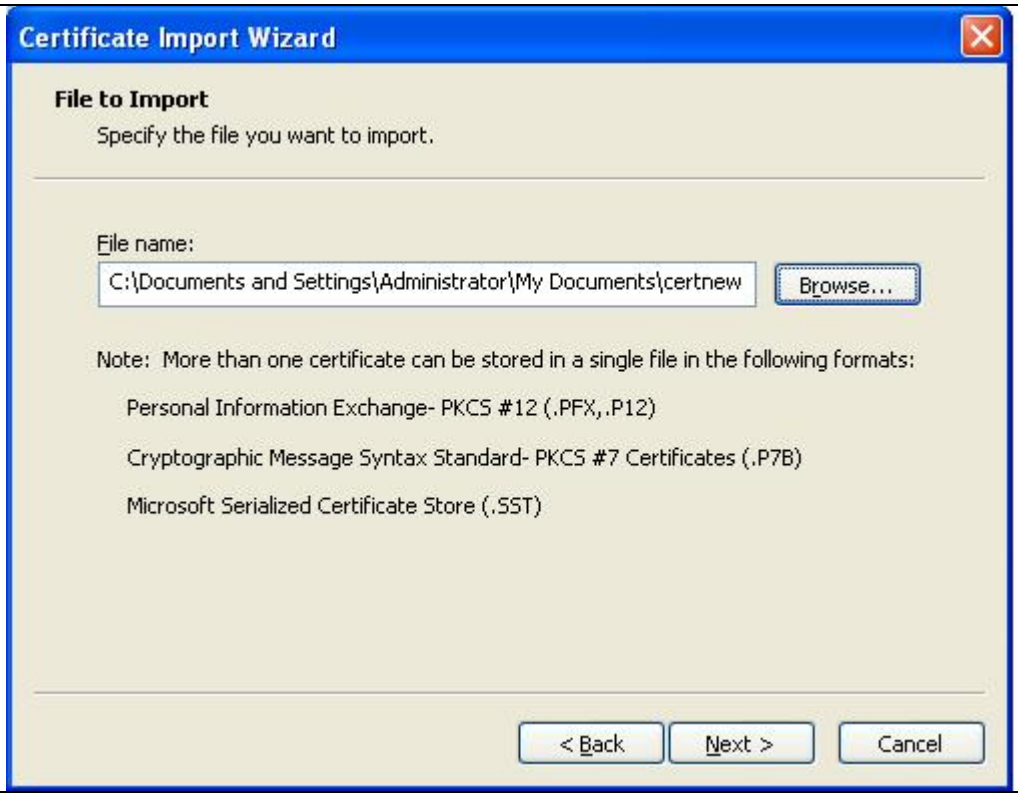


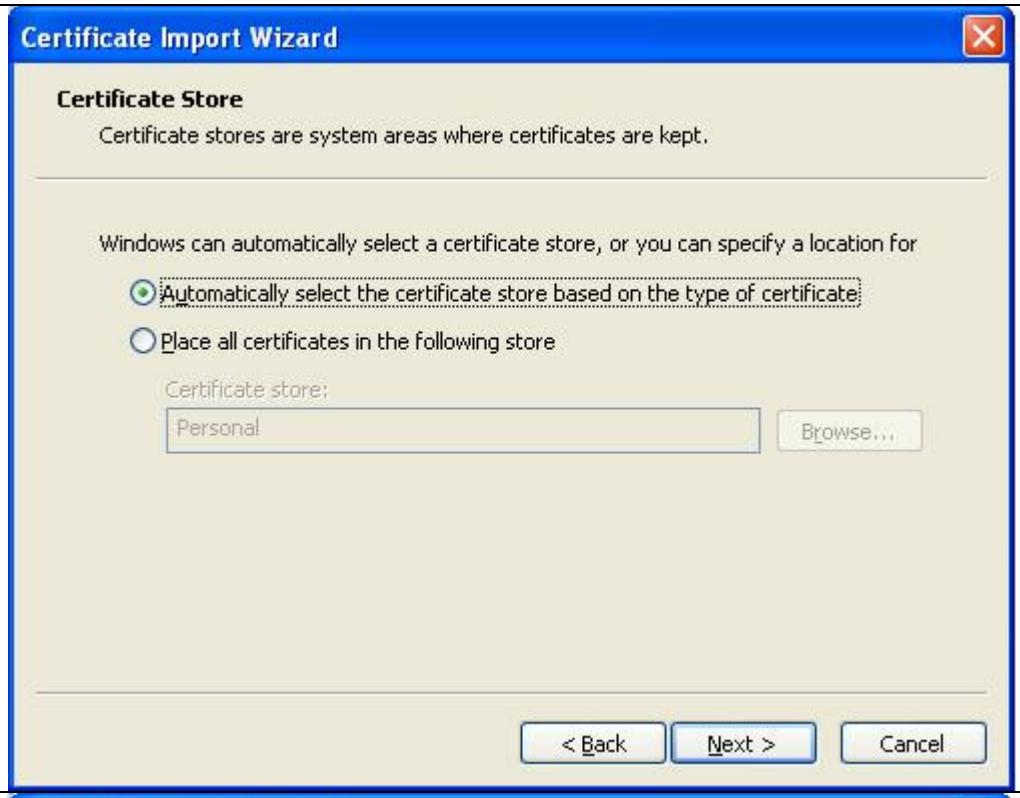
Click "Download a CA certificate, certificate chain, or CRL".	
Select "Download CA Certificate Chain". This file will have both R2 and RootCA's public key certificate.	

Save it.	
Go to "Tools" and click "Internet Options".	

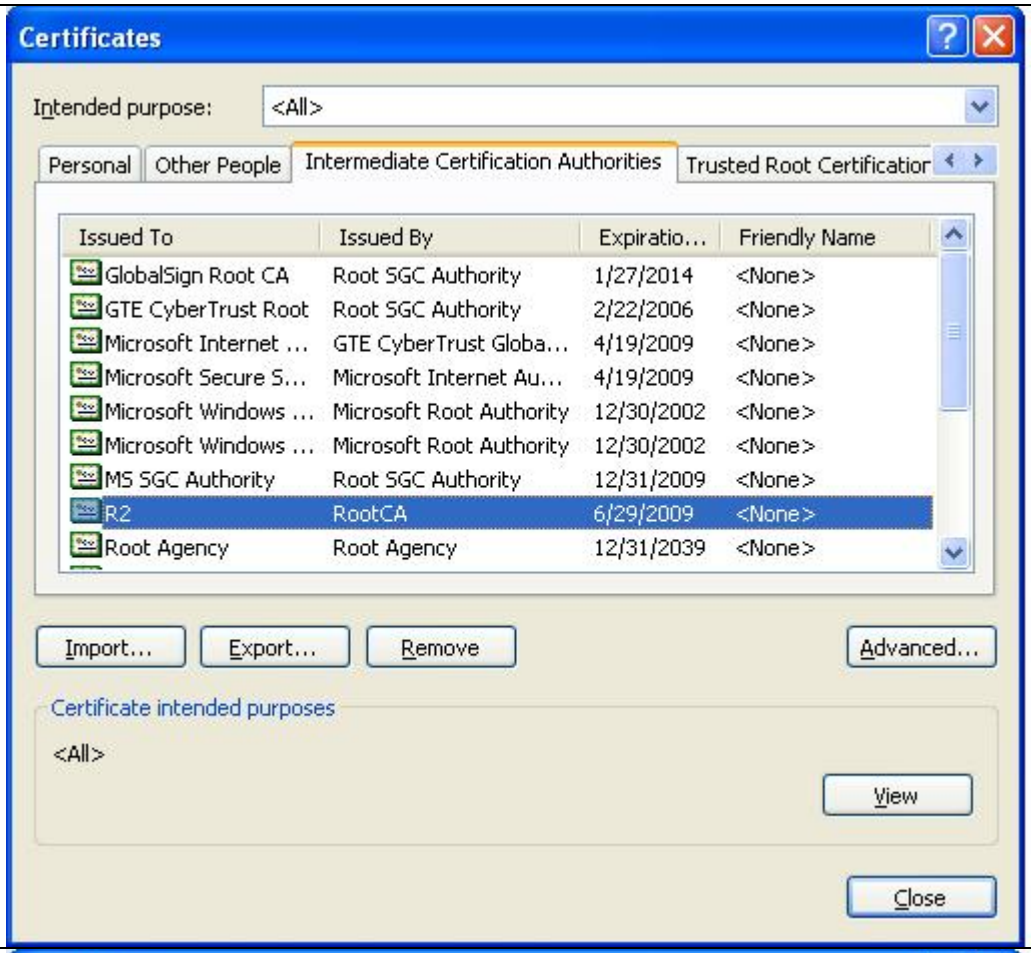
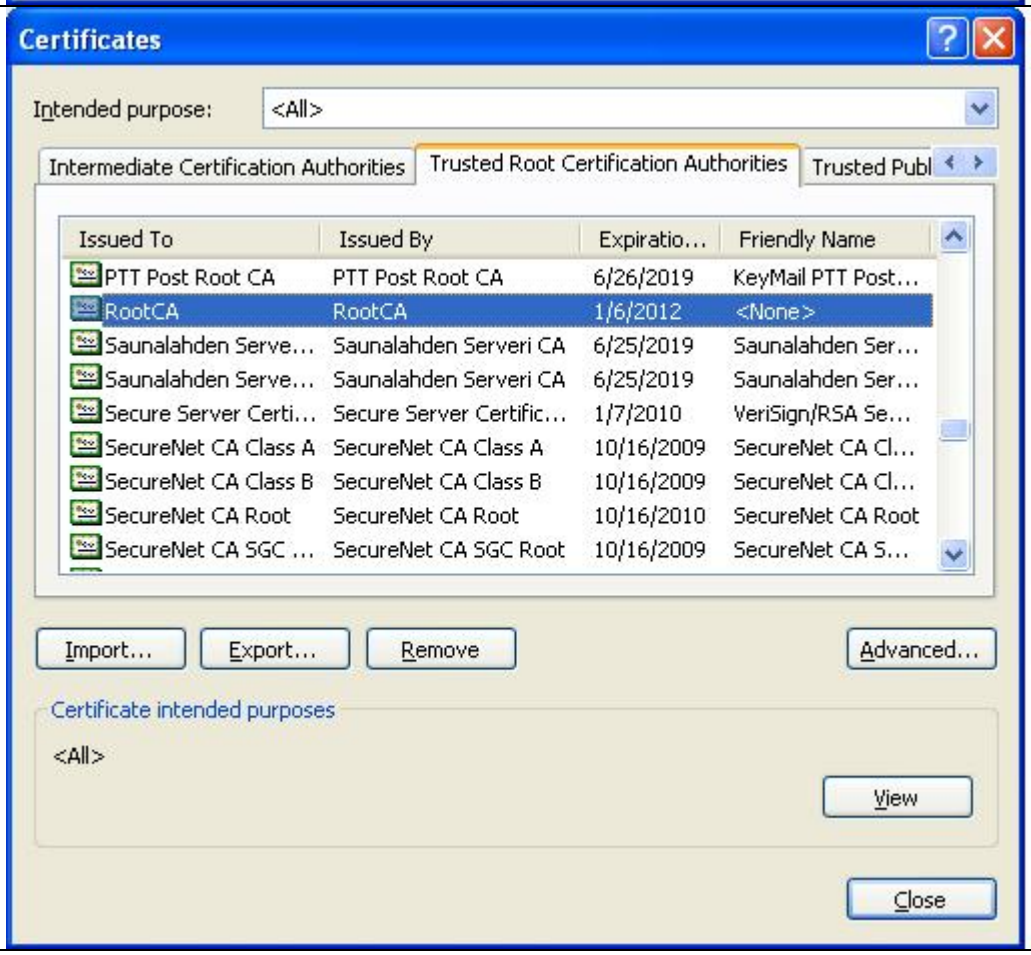
Click “Certificates”	

Click “Import…”	
Click “Next”	

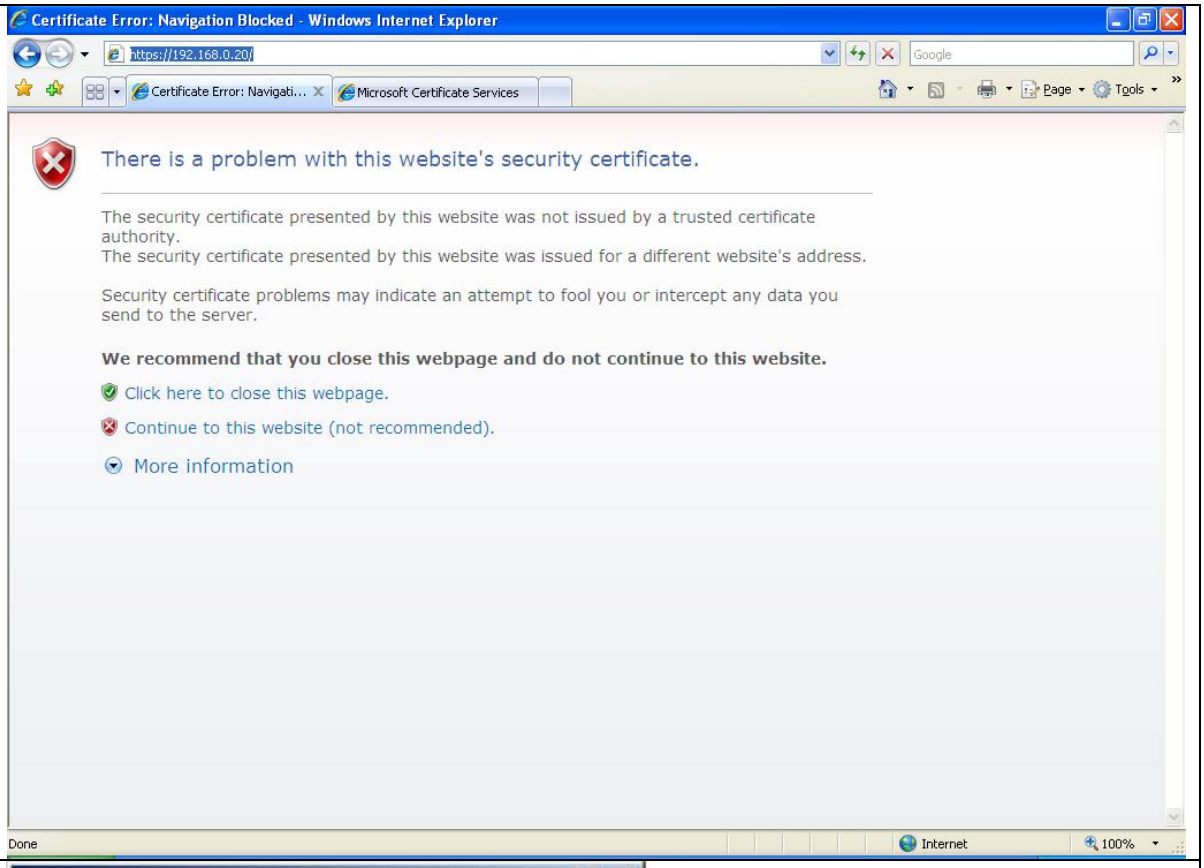
Select the file	
Click "Next"	

Select "Automatically select the certificate store…." Click "Next"	
Click "Finish"	

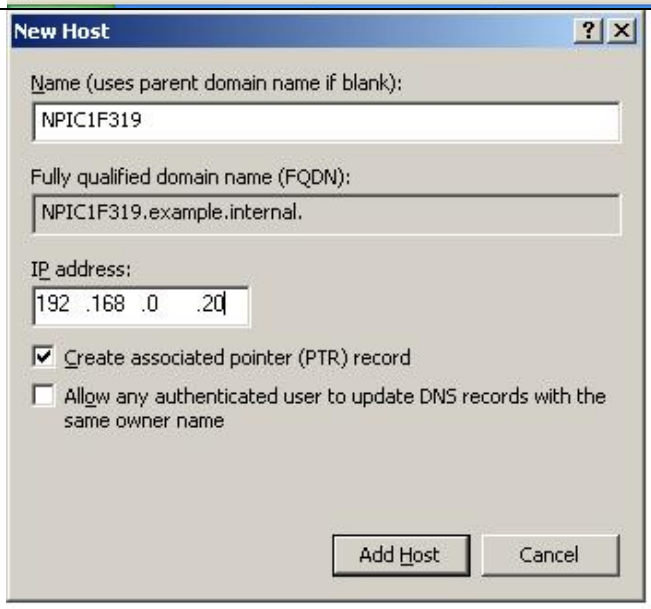
Press "Yes". Note the Security Warning message. Installing a CA public key certificate as a trusted Root CA is a big deal. You need to be very sure this is what you want to do.	
Click "OK"	

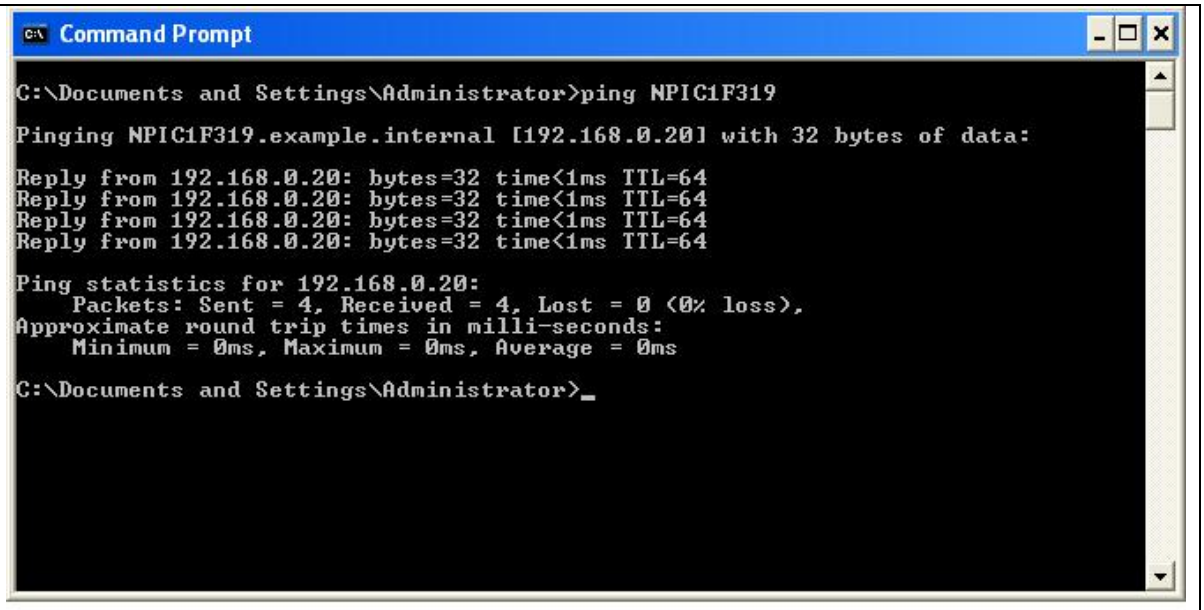
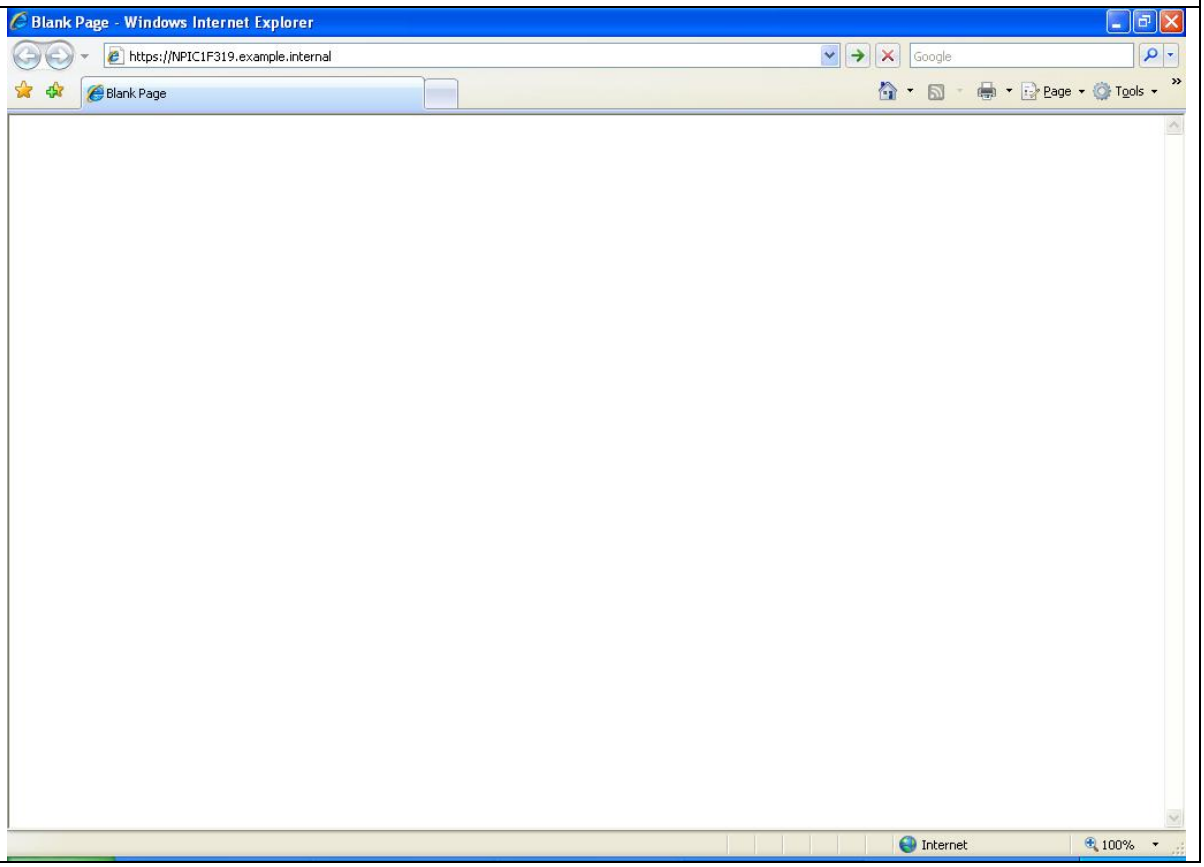
Select the tab "Intermediate Certification Authorities" and we can see that R2's public key certificate has been installed. Yea!	
Click the tab "Trusted Root Certification Authorities" and we see RootCA has been installed. Yea!	

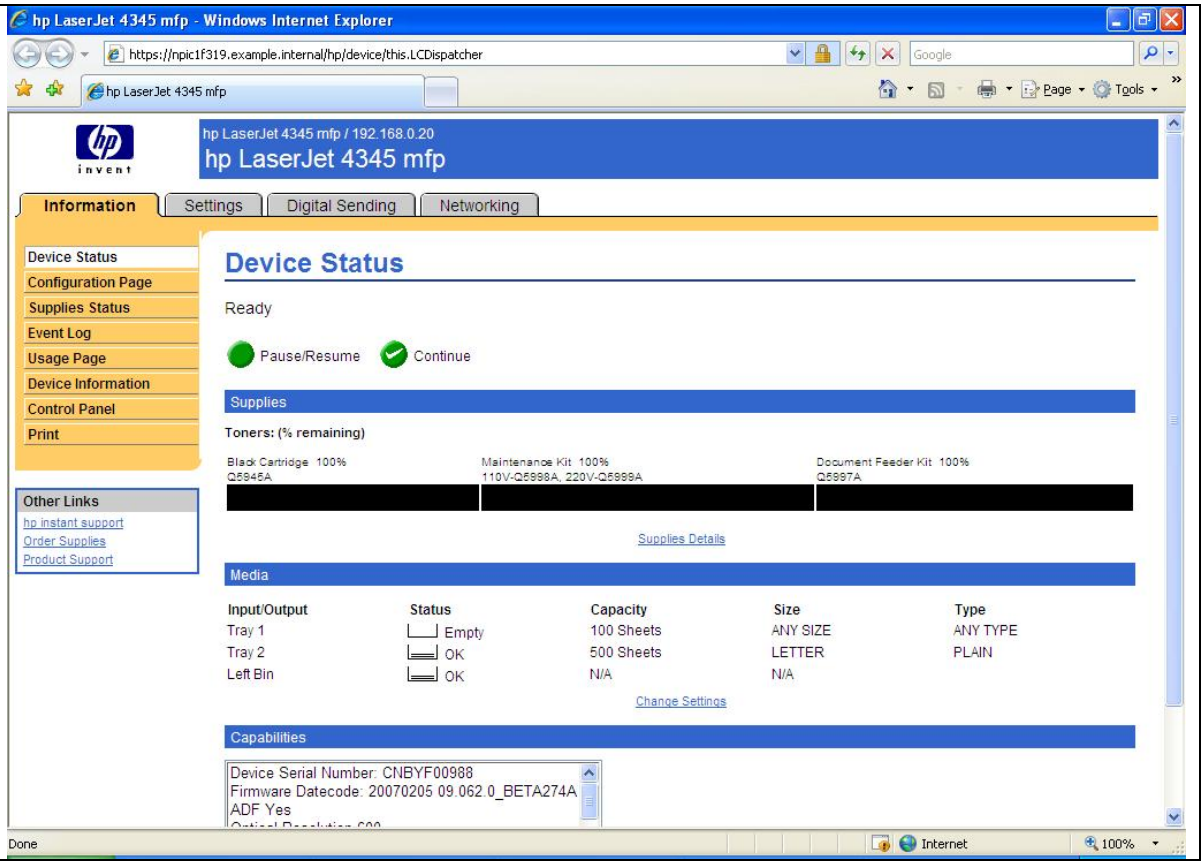
Now we go back to the web page and still get an error!! No!! The problem is that we have a name mismatch. We are using the IP address in the URL and the IP address is no where to be found in the certificate. We need to use the name that the certificate has. Time to create a DNS entry for the printer.



Here is our DNS entry which matches the Subject CN in the certificate.



We ping it just to be sure. Looks good.	
We go back to the web browser and enter the name instead of the IP address.	

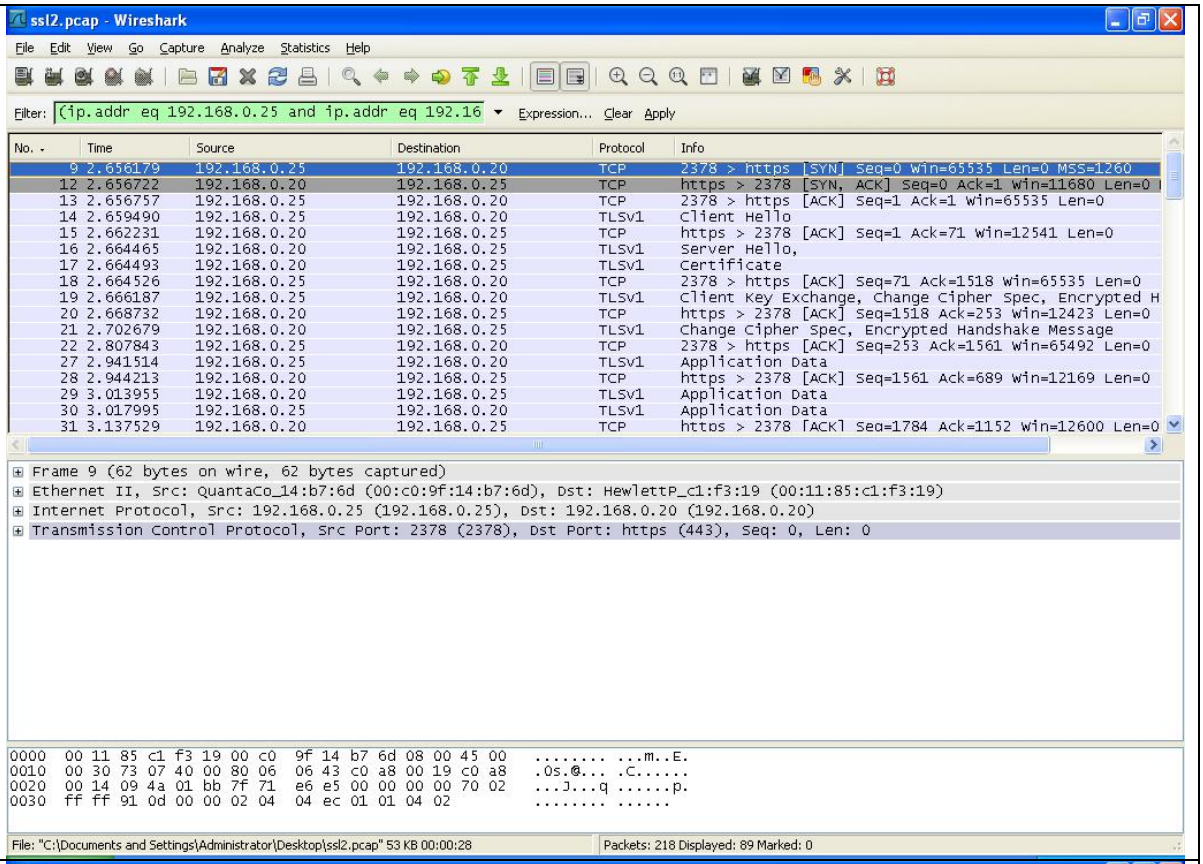
Everything worked!	

Now SSL/TLS is working for HP Jetdirect just like it would work for an Internet secure shopping experience.

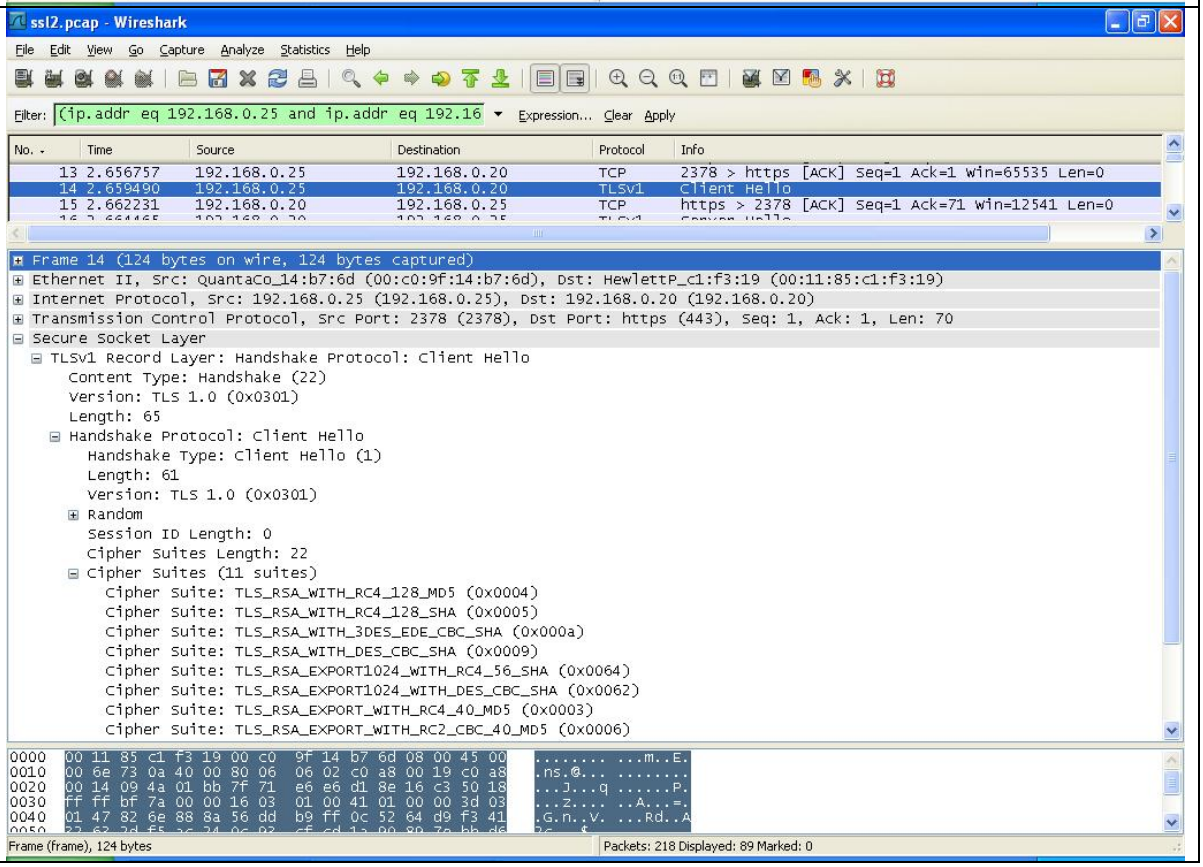
A Detailed Look at the SSL/TLS Connection

Good stuff so far! Let's bring up Wireshark and see what was actually happening on the wire during the successful https connection.

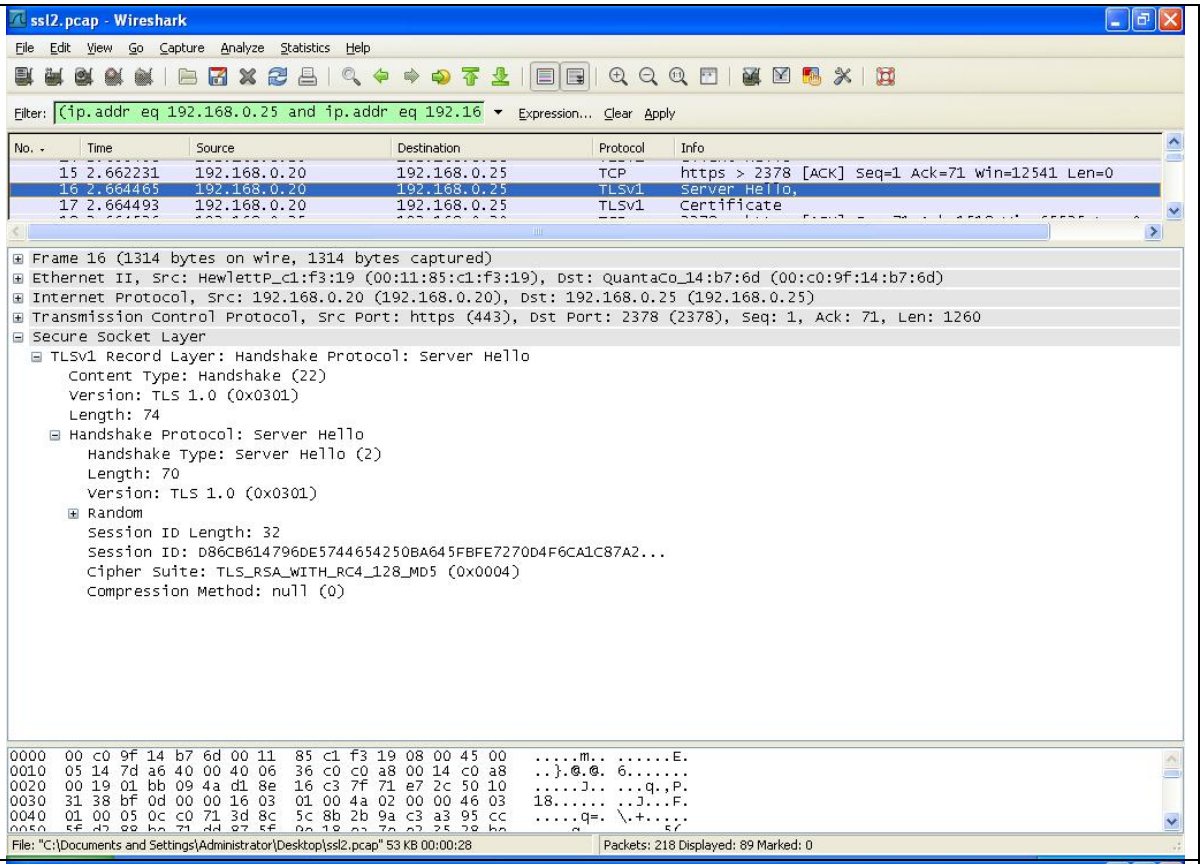
We see the TCP connection is established to "https" or TCP port 443. The client is 192.168.0.25 and the web server is 192.168.0.20.



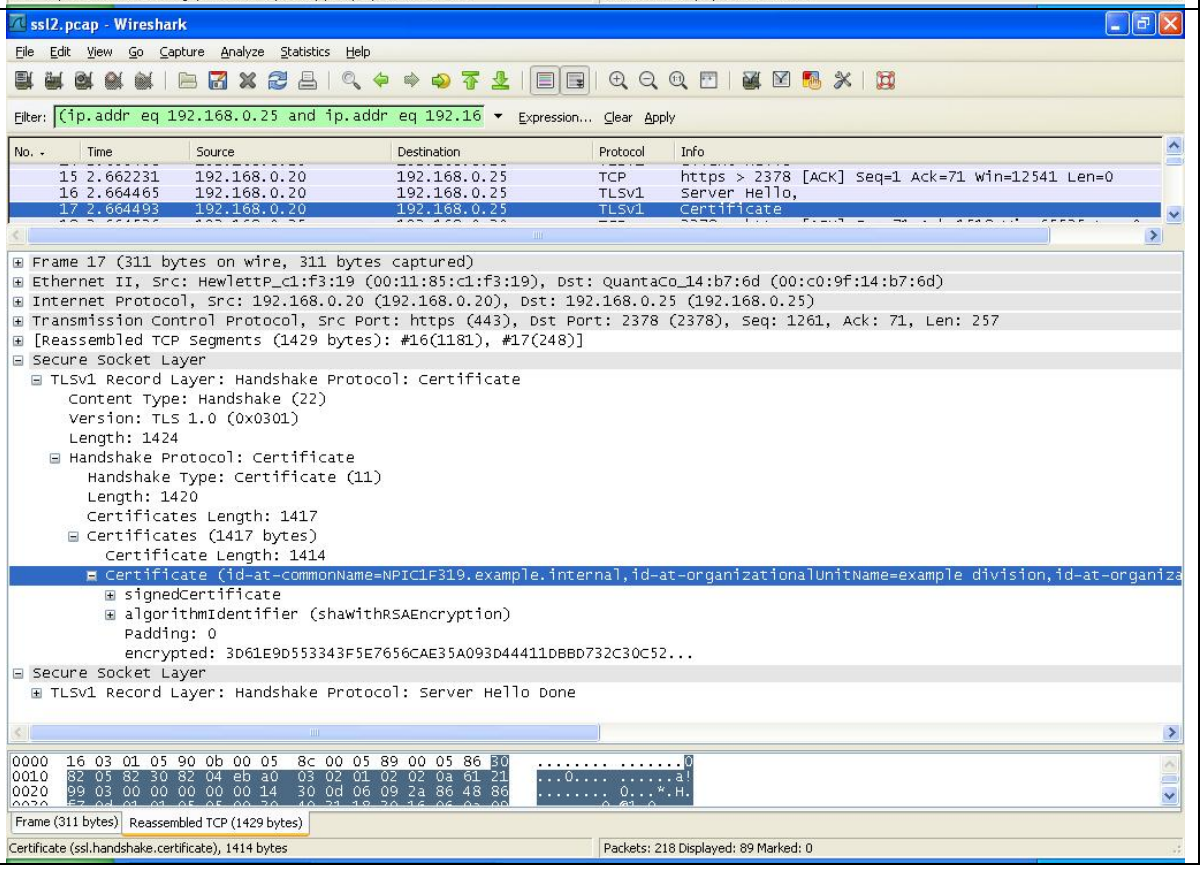
We see the SSL "Client Hello" message. Notice the detail. TLSv1 "Record Layer" and "Handshake Protocol". Based upon our previous discussion, we know the client is going to send a random number and the cipher suites it supports.



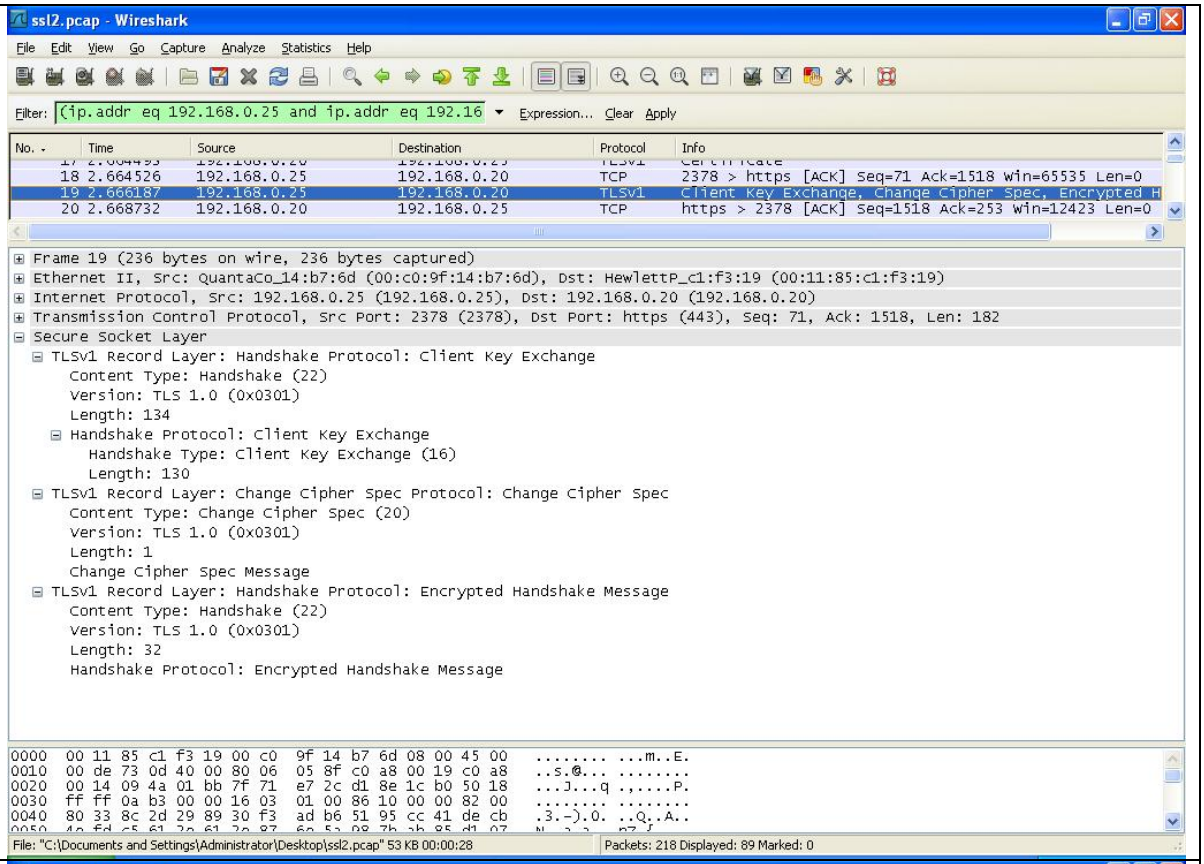
Now let's look at the server hello. Here we see a random number and the cipher suite selected to be used: TLS RSA WITH RC4 128 MD5



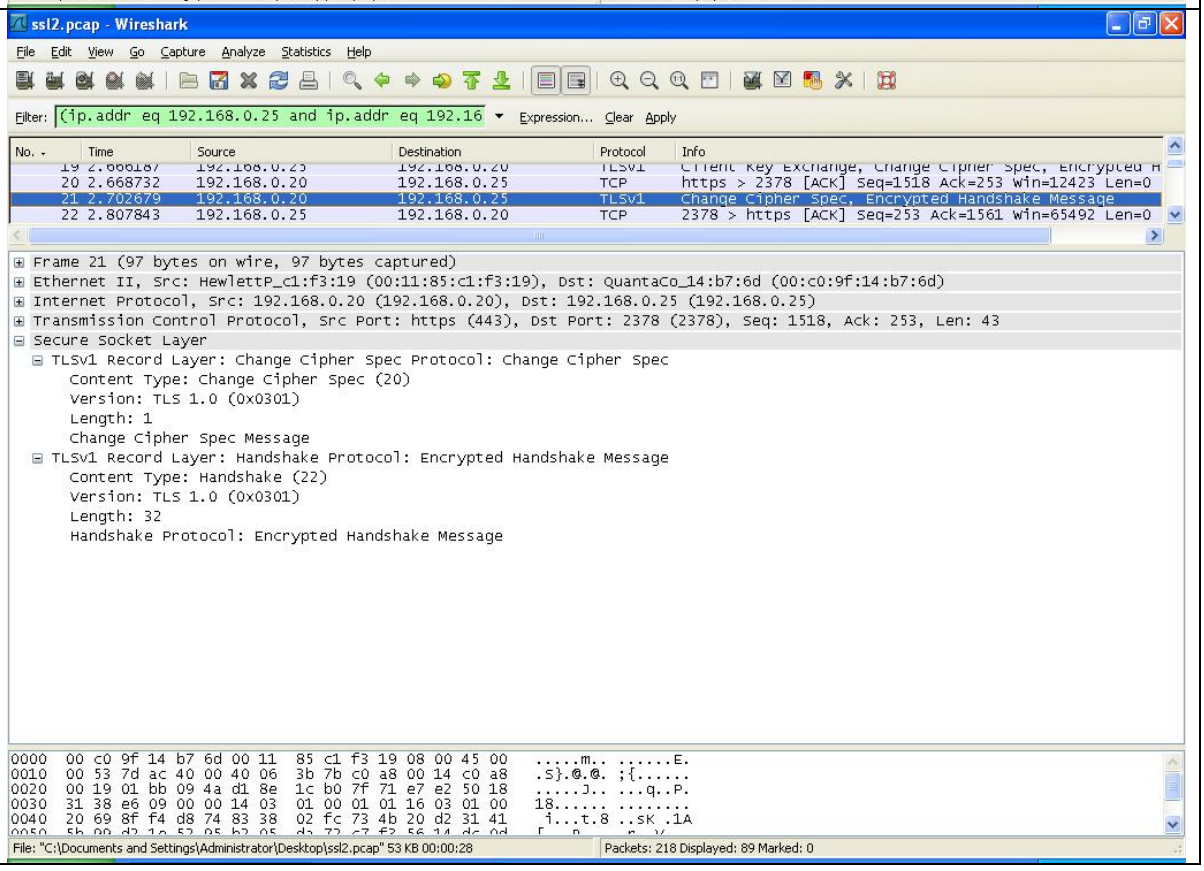
We see the server's certificate. We can tell from the common name that it is the one we just assigned Jetdirect previously. This packet also contains the "Server Hello Done" message.



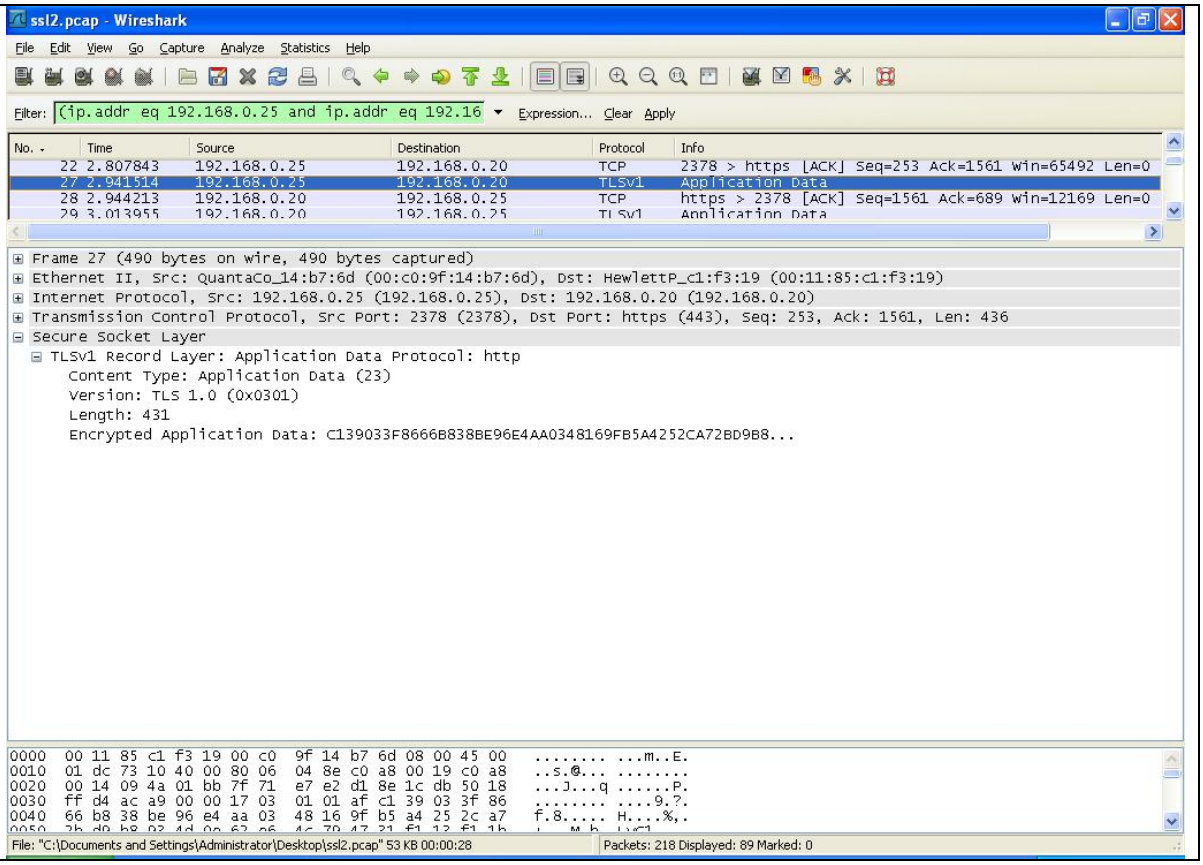
Here the client is sending over the pre_master_secret encrypted with the server's public key. It is also letting the server know it is changing keys to the ones derived from the master_secret



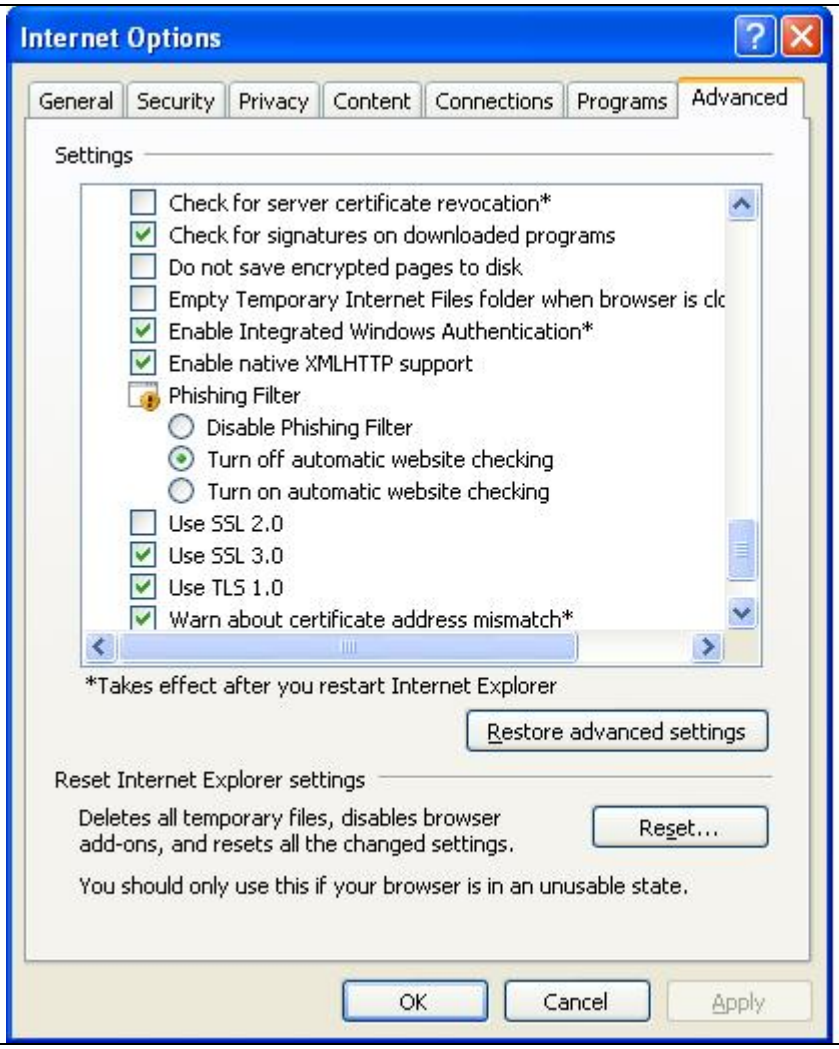
Same info coming from the server this time.

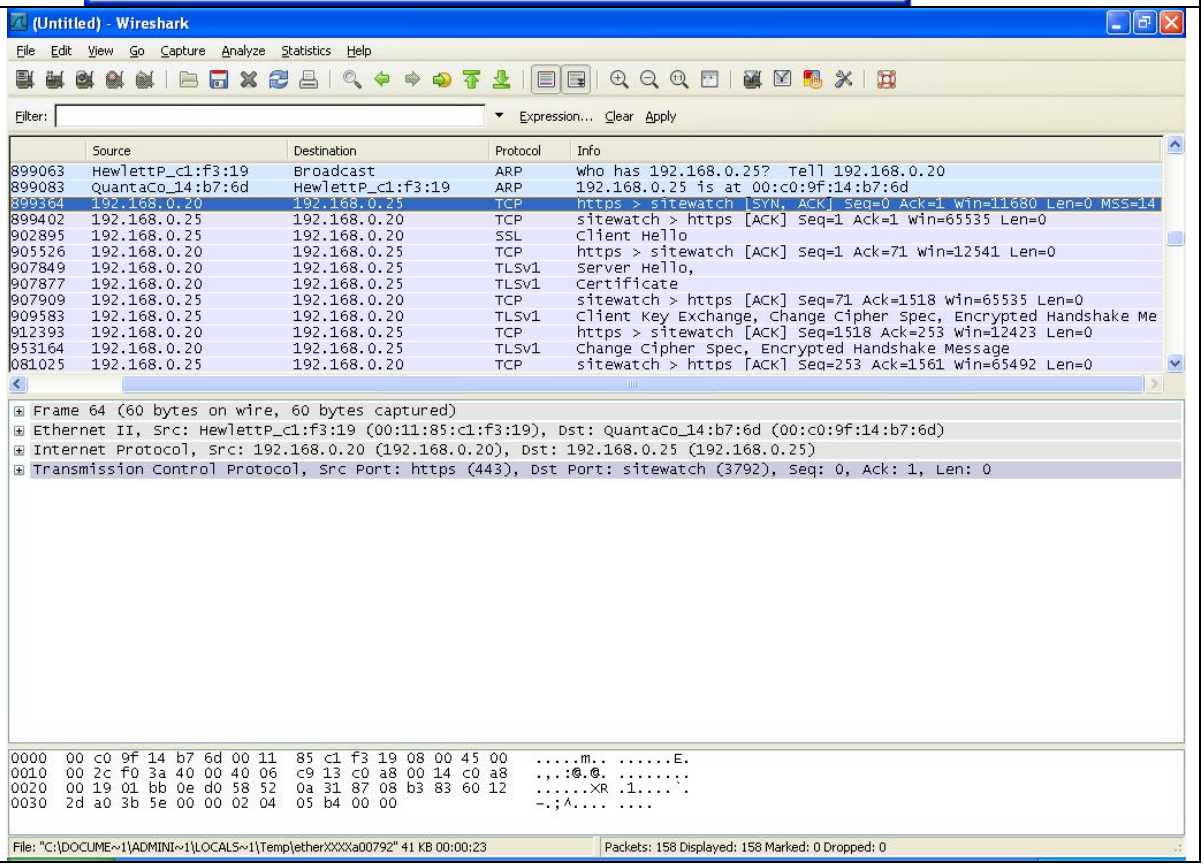


Now we have actual client data – this is probably the initial HTTP request encapsulated in SSL/TLS.

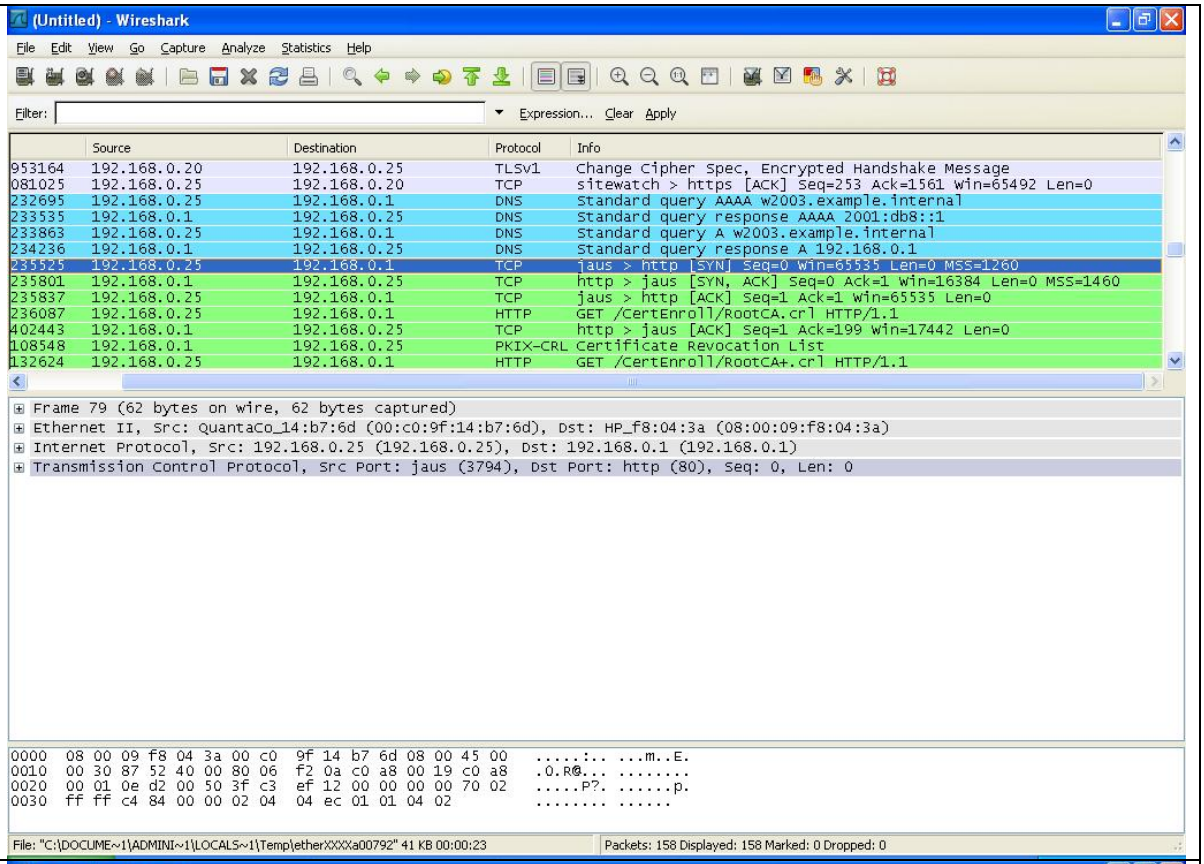


There was one check that wasn't done – the CRL. This check wasn't done because it is disabled by default. Going into the "Internet Options" under the Tools menu for IE7, we then click the Advanced tab and look what we find.

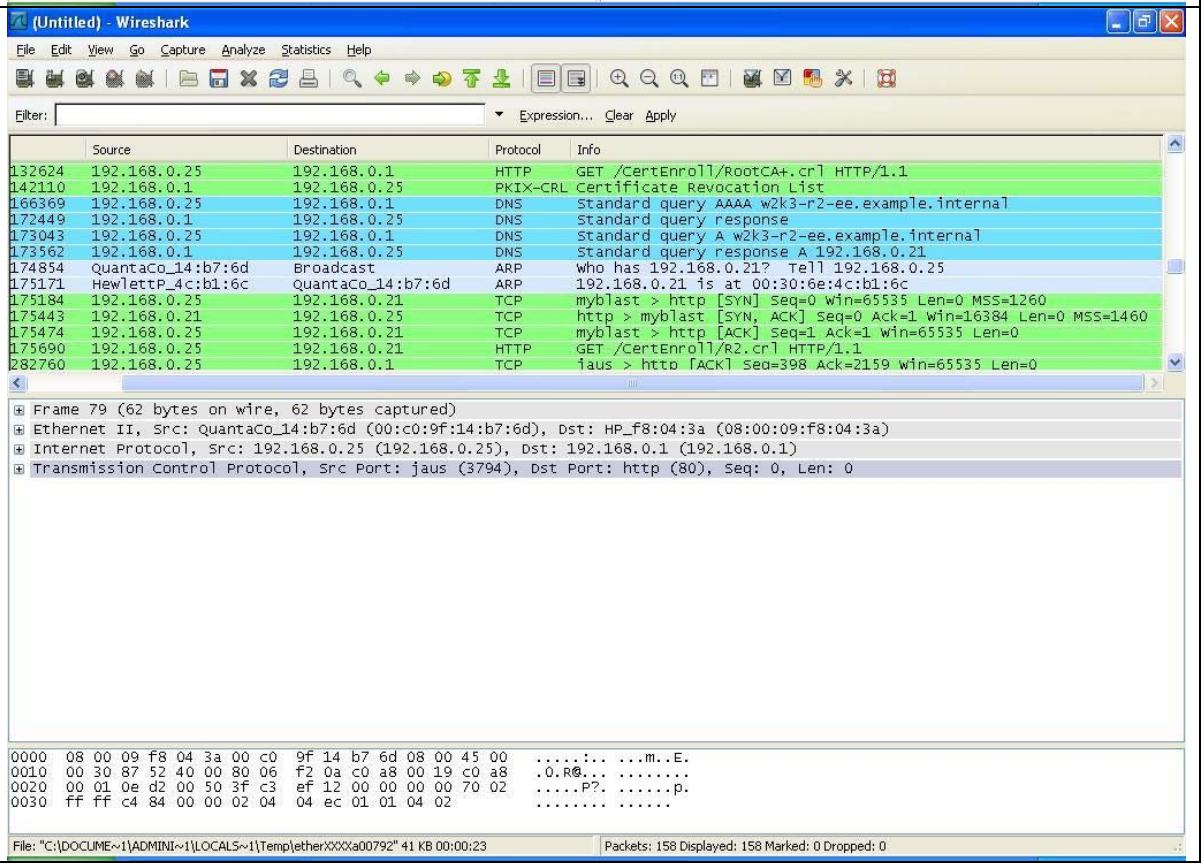
Check for server certificate revocation is not selected.	

Let's select it and restart IE7.	
Here is another HTTPS connection to the same LJ4345MFP. Everything looks the same so far	

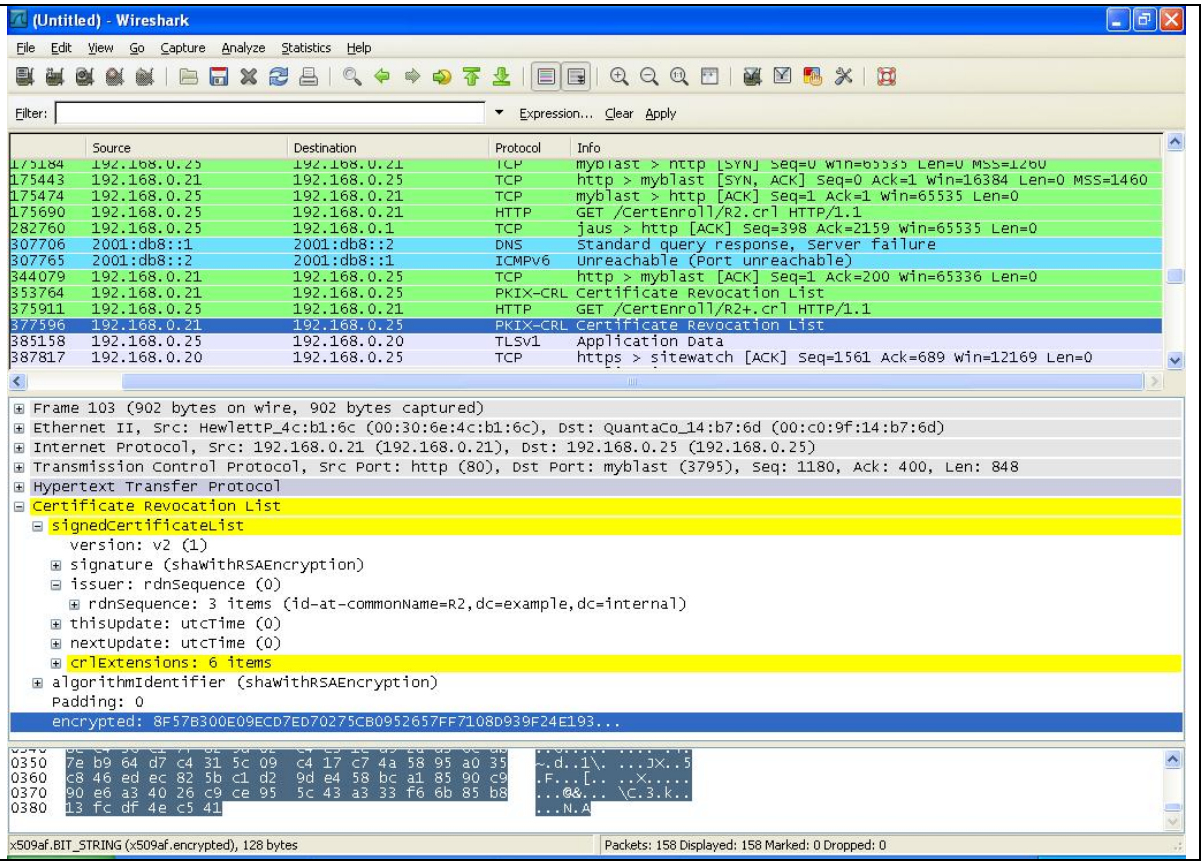
Here we go – looks like before any application data is sent, the CRL is check using http. This one is going to the RootCA



Another CRL request to R2.



Here is the content of the CRL – encrypted of course.

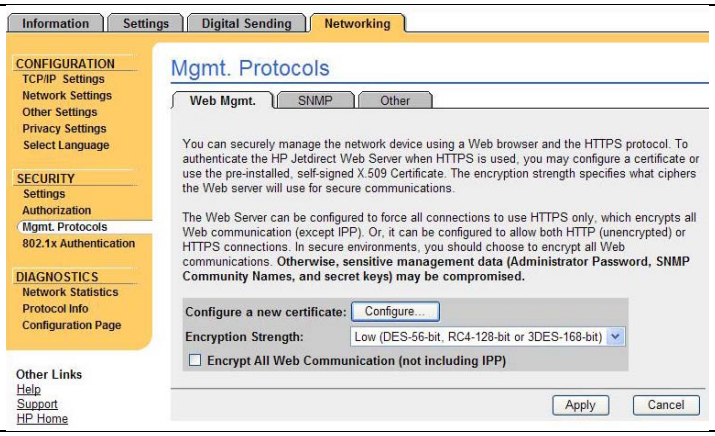


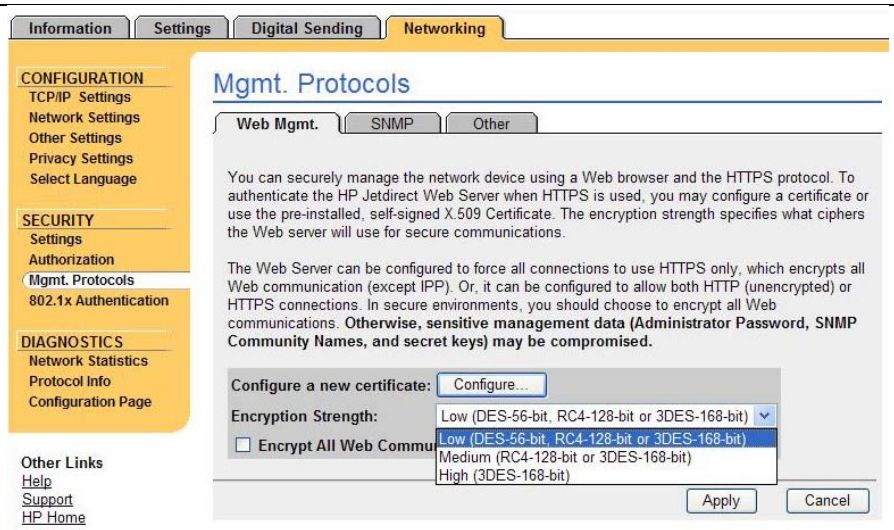
A performance hit would occur when CRLs are checked. That is probably why it isn't checked by default.

SSL/TLS Server Settings

HP Jetdirect has a couple of useful settings to control how SSL/TLS clients connect to it. Let's have a look.

There are three main settings for the SSL/TLS server. One is the Certificate and we've covered that. The next one is the checkbox "Encrypt All Web Communication". When that is checked, Jetdirect will redirect HTTP requests to use HTTPS so that HTTPS is effectively forced to be used.



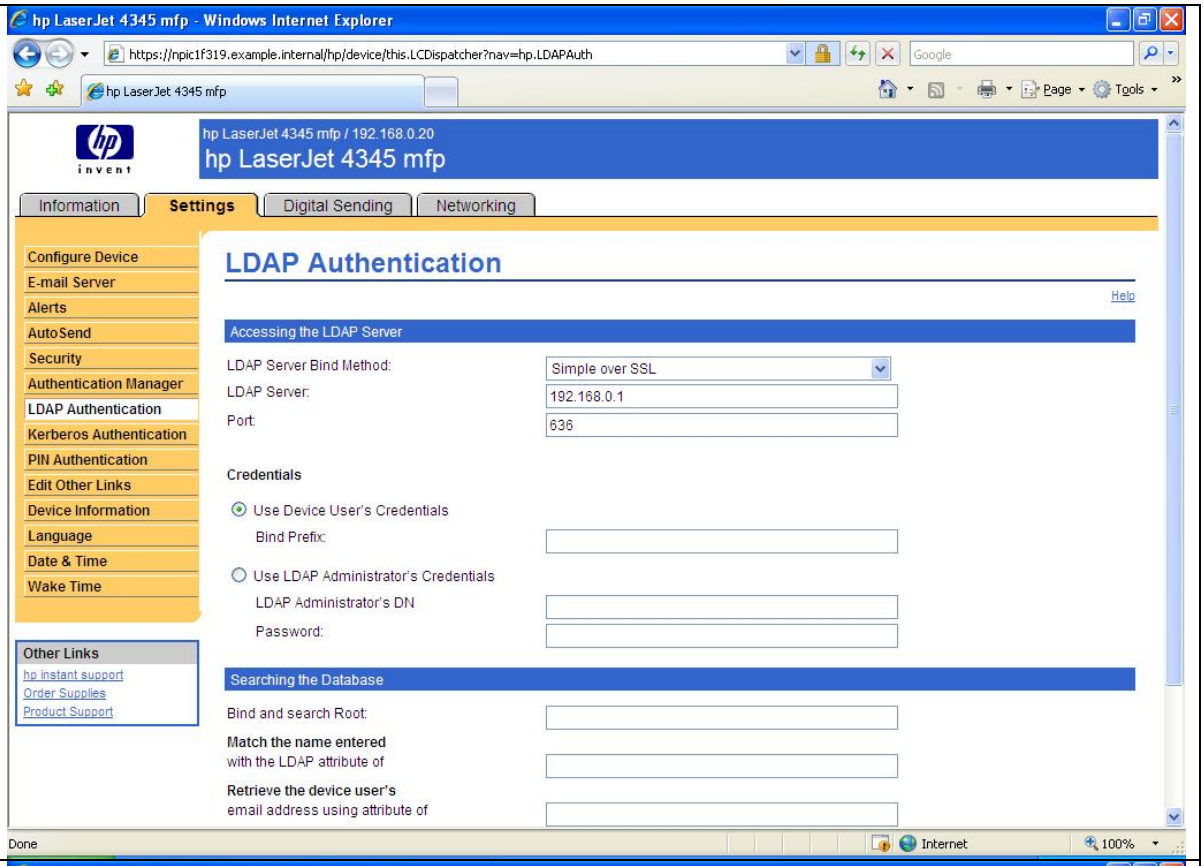
The setting "Encryption Strength" controls the cipher suites that Jetdirect will select from a client request. The default setting is "Low" which is a bit misleading – it really means that all cipher suites that Jetdirect supports can be used including ciphers that aren't considered as secure anymore. If the client can only support DES, Jetdirect will still accept it. However, if the client offers DES and other cipher suites, Jetdirect will prefer higher security ciphers when presented with the choice. Setting it to "Medium" means that the client must offer RC4 or 3DES or the SSL/TLS connection won't be established.	

Well, looks like we've covered all that is necessary when HP Jetdirect acts as an SSL/TLS server. But wait, there's more! HP Jetdirect can also act as an SSL/TLS client when used by certain applications on a printer or MFP. The most popular one is LDAP over SSL/TLS. Now, what was formerly important to the SSL/TLS client (e.g., browser) becomes important for HP Jetdirect. Let's look at what happens here.

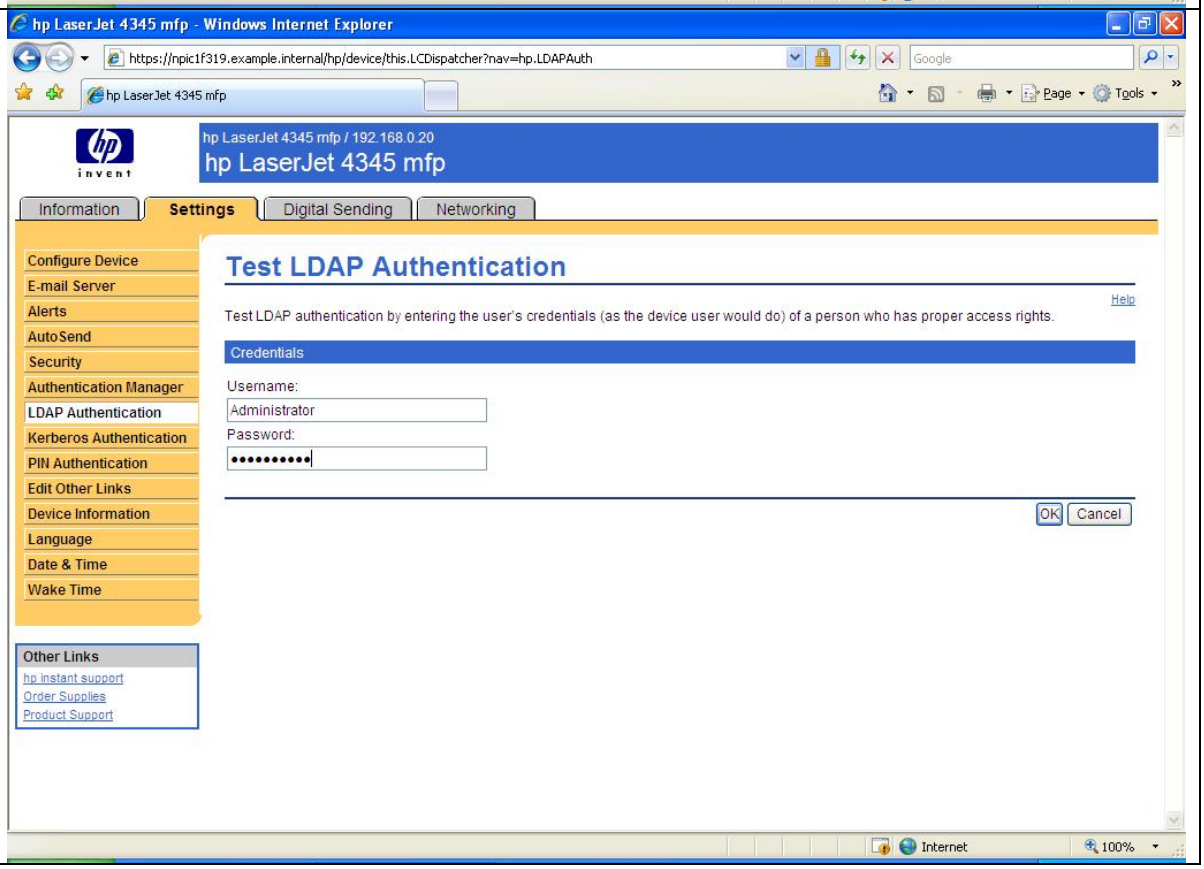
HP Jetdirect as an SSL/TLS Client

The most common situation for HP Jetdirect to act as an SSL/TLS client is when the MFP is going to use LDAP over SSL. Keep in mind that the roles are reversed here. HP Jetdirect is going to initiate a connection and verify the server's certificate just like a web browser would. Let's set this up.

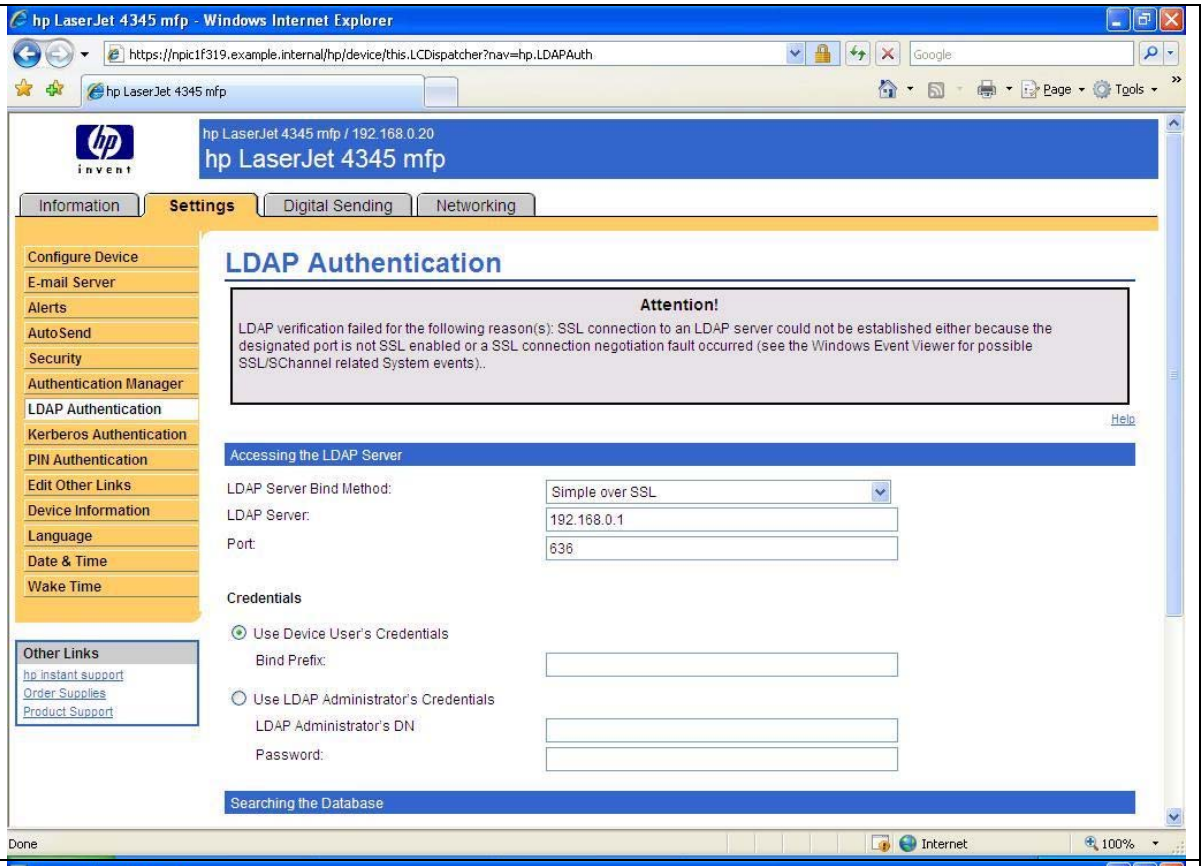
We are going to select Simple over SSL as the LDAP server bind method and use the IP address of 192.168.0.1, which is our LDAP server. We then scroll to the bottom and hit the "Test" button (not shown).



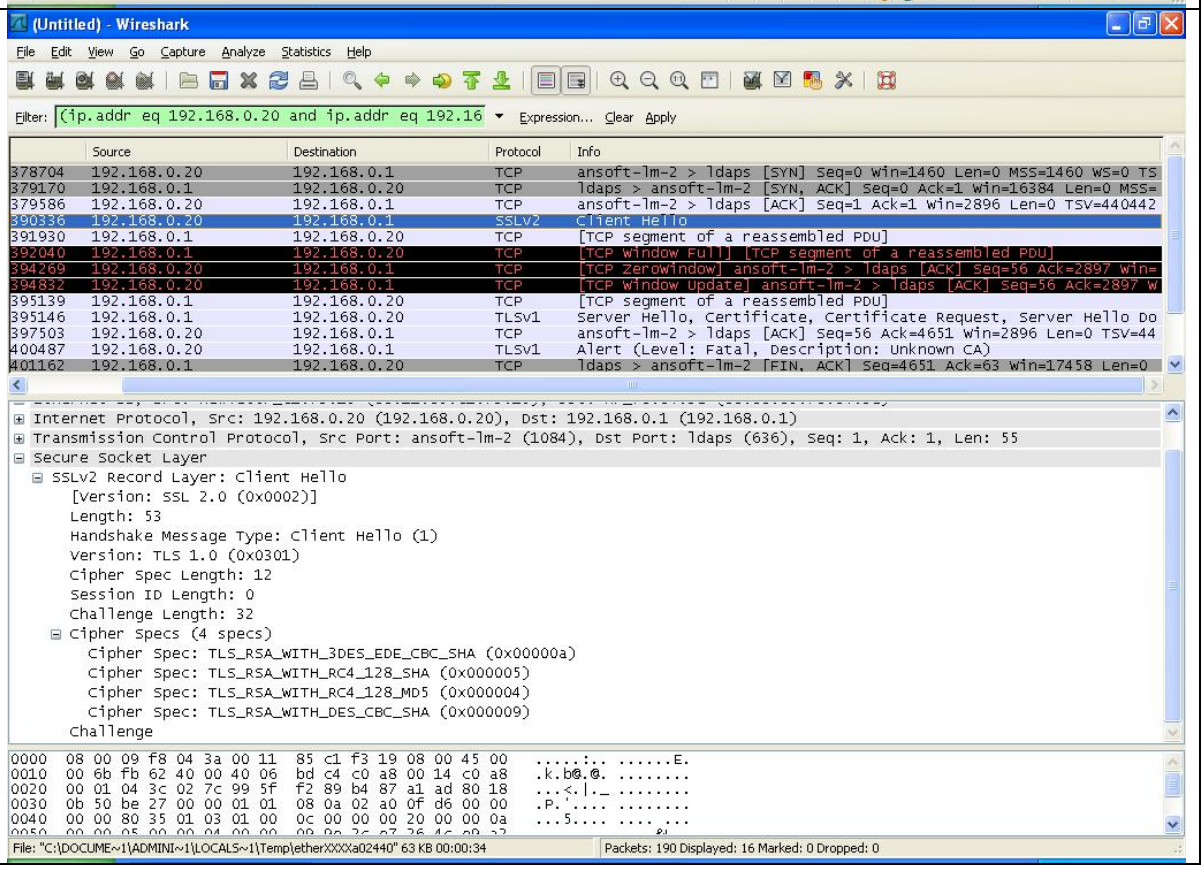
We are asked for credentials and we provide them and hit OK.



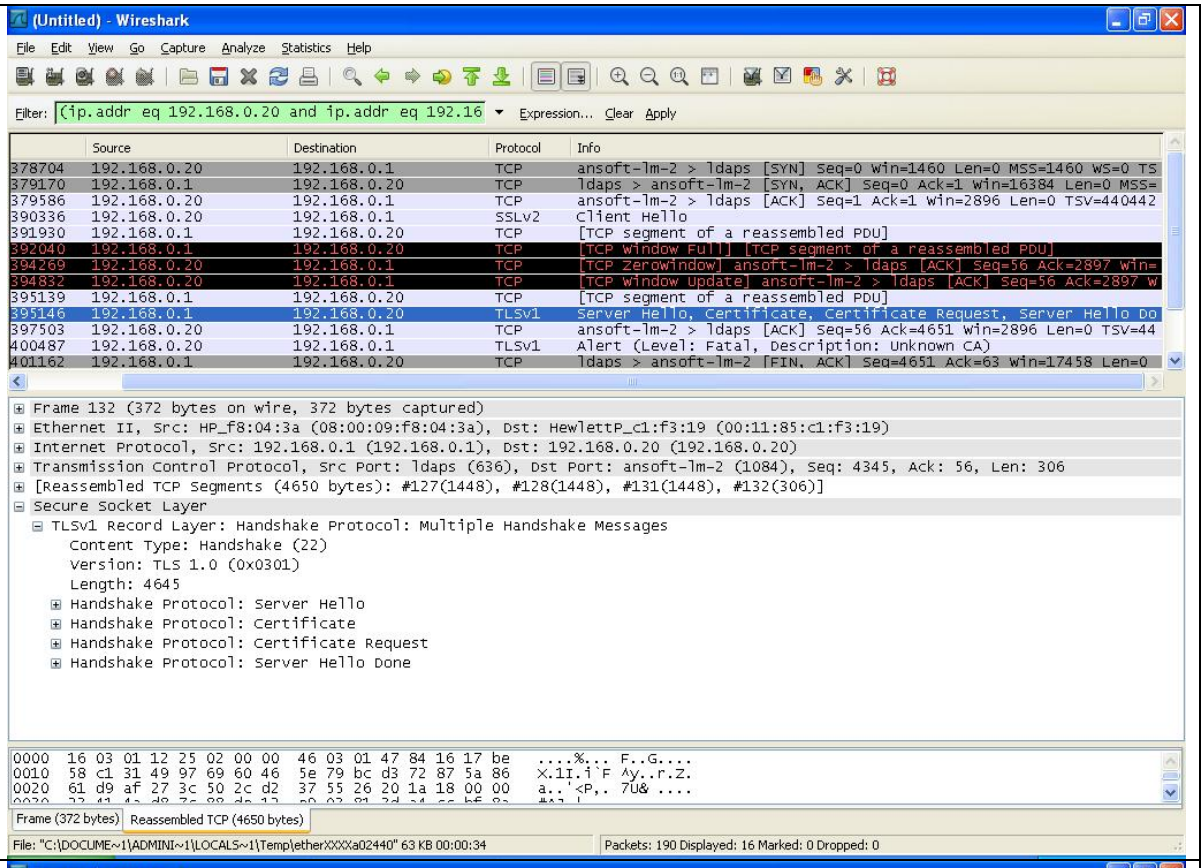
Error message – it didn't work. Let's look at a trace.



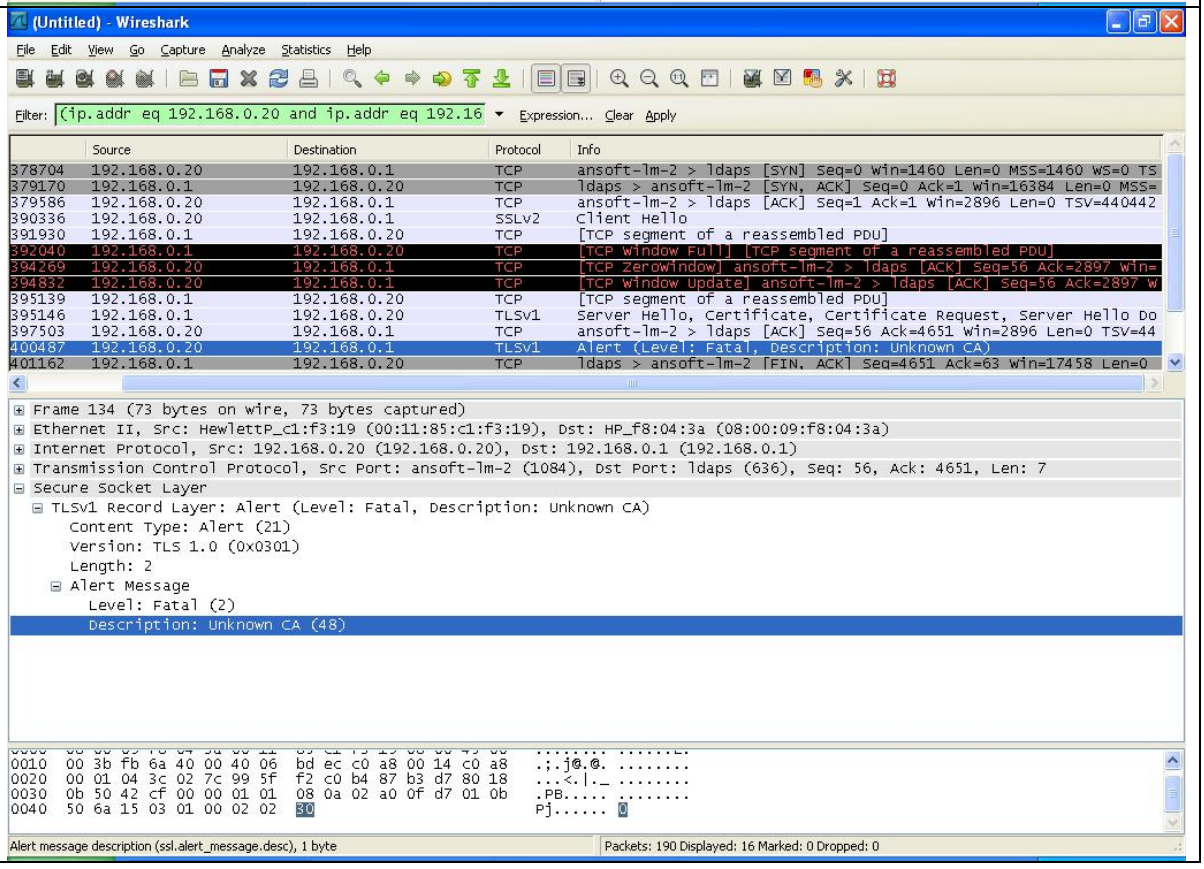
Here we see Jetdirect taking on the role of the client. It initiates the connection and sends the Client Hello.

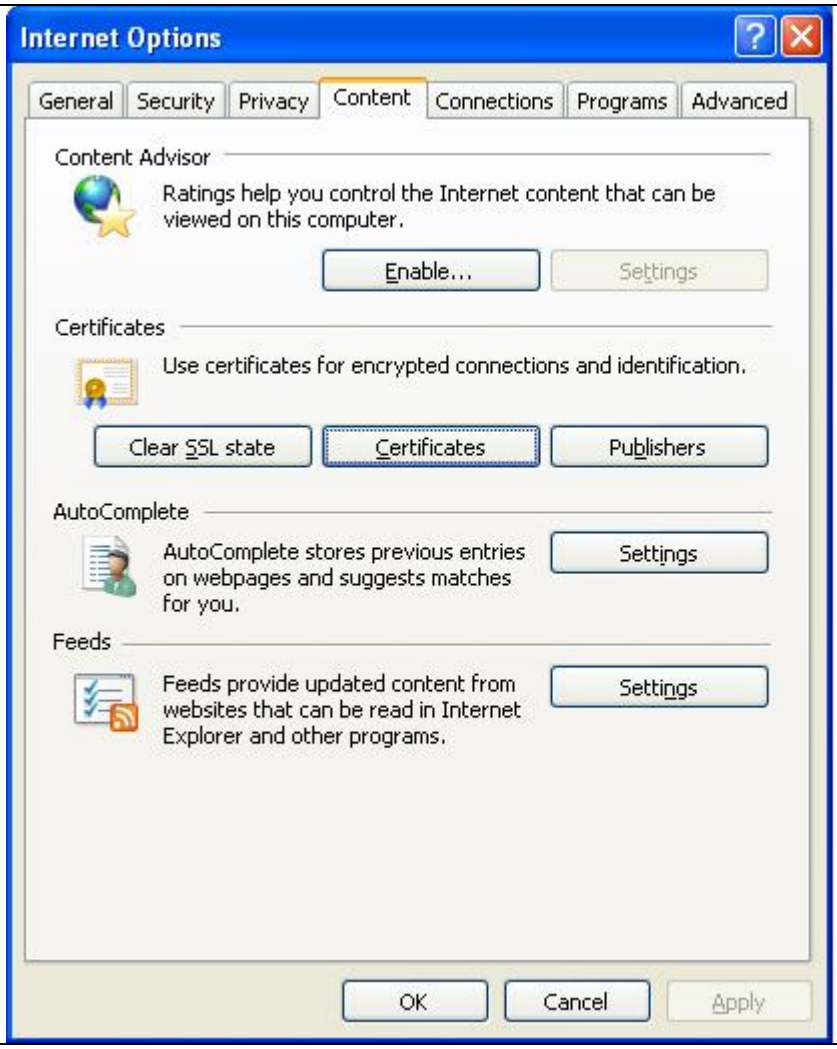


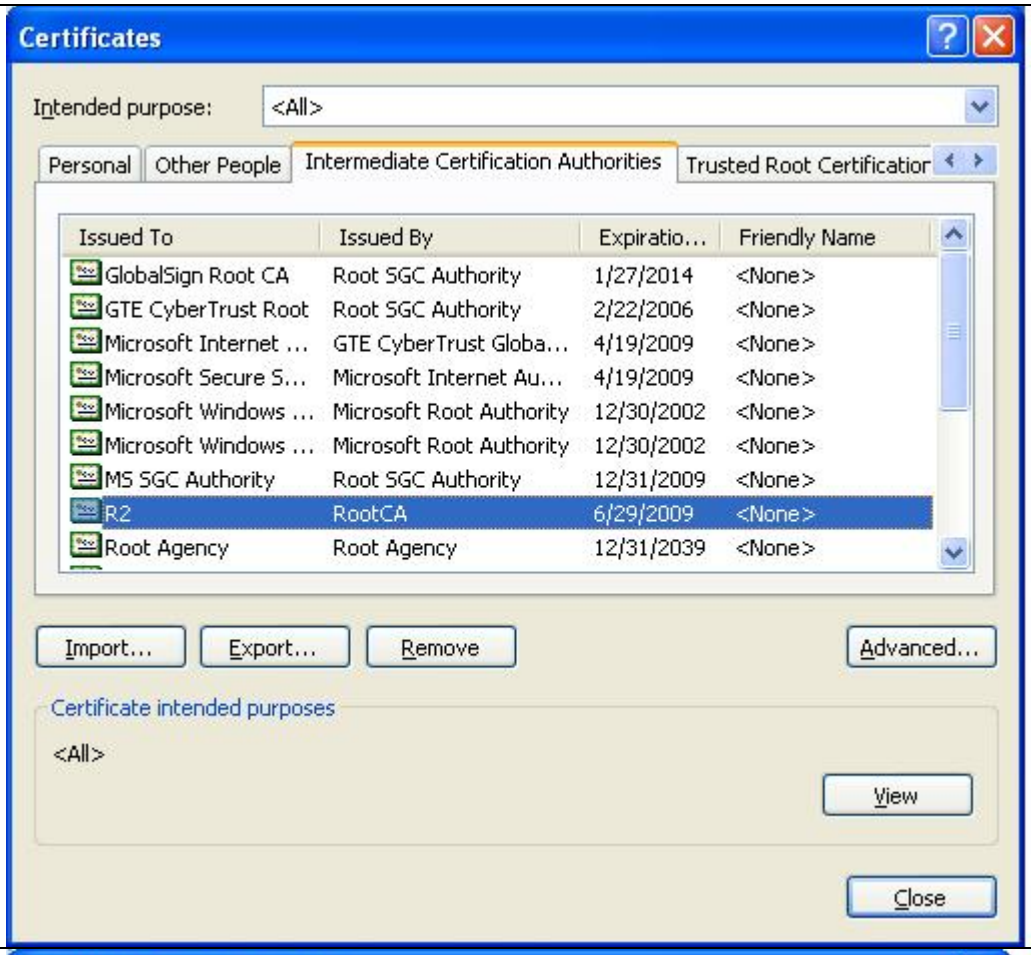
The server responds. There is a new message here – one we haven't talked about. The Certificate Request. The server is indicating to Jetdirect that it must send it certificate to the server to be validated. We are in good shape because Jetdirect already stored a certificate capable of doing Client and Server authentication. That is a good thing!

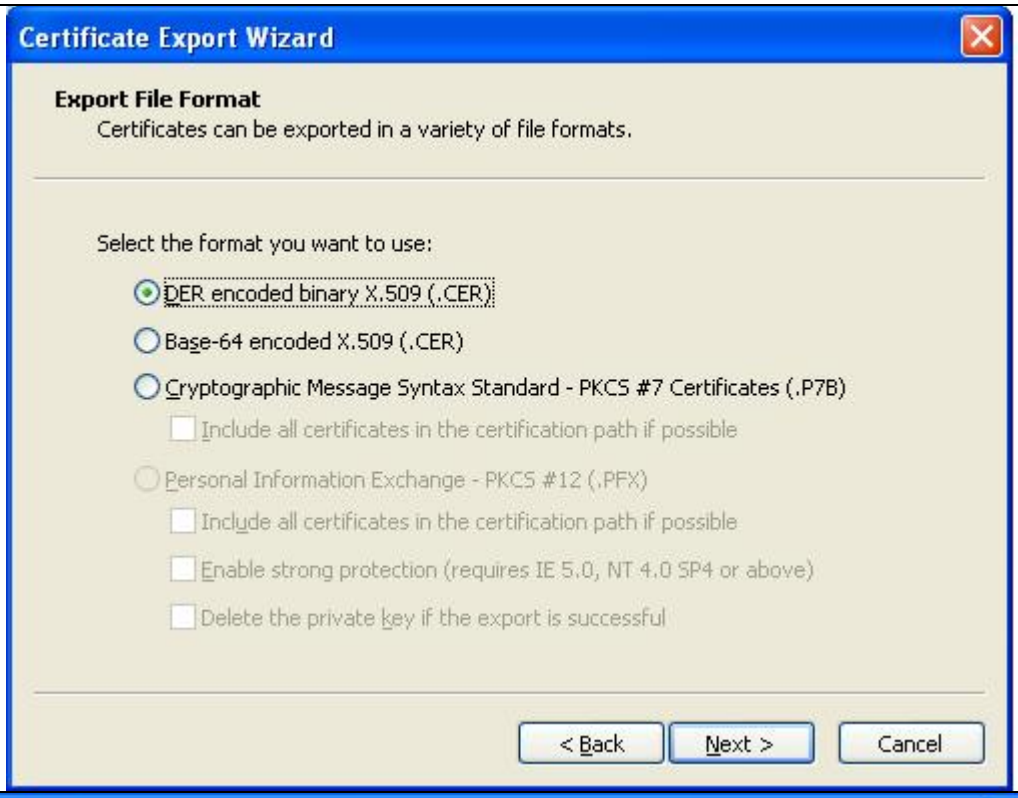
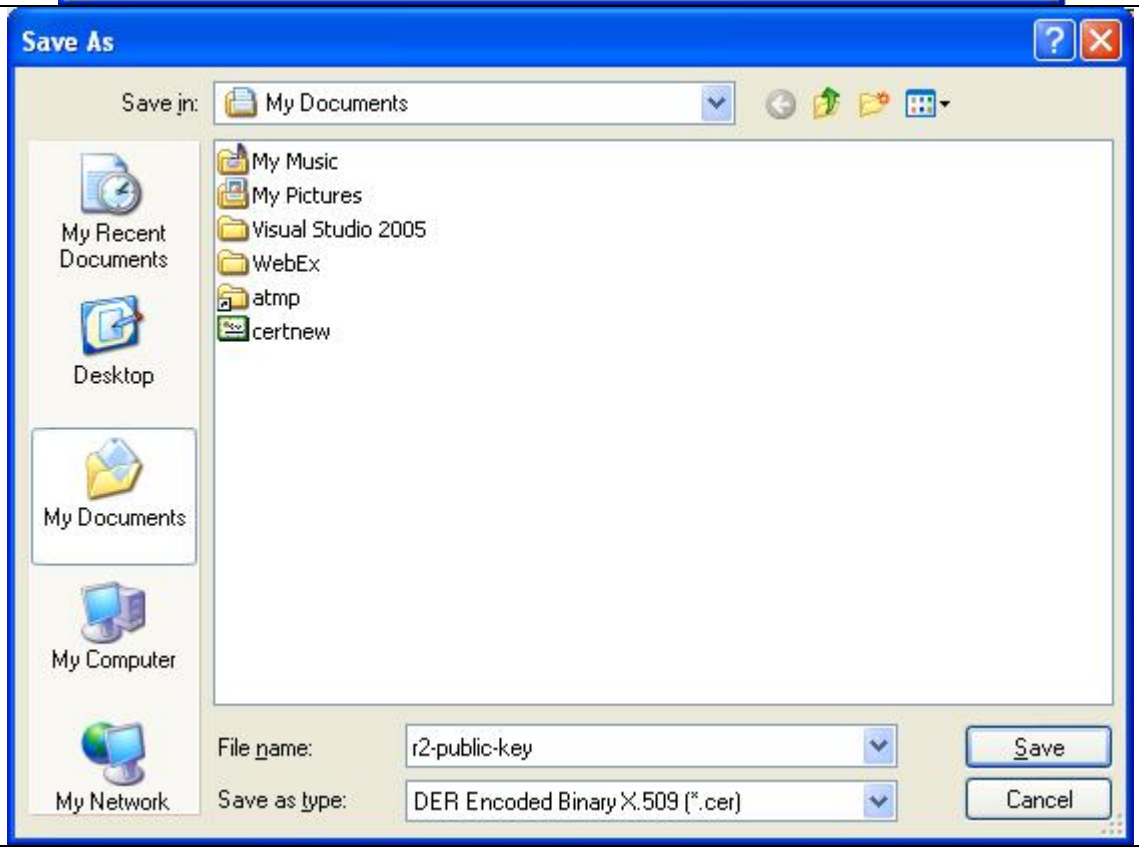


Here we have a TLS Alert indicating Unknown CA. Jetdirect is sending this message to the server. This means that Jetdirect performed some checks on the server certificate and couldn't verify that it was trusted. In short – Jetdirect needs a CA certificate configured.

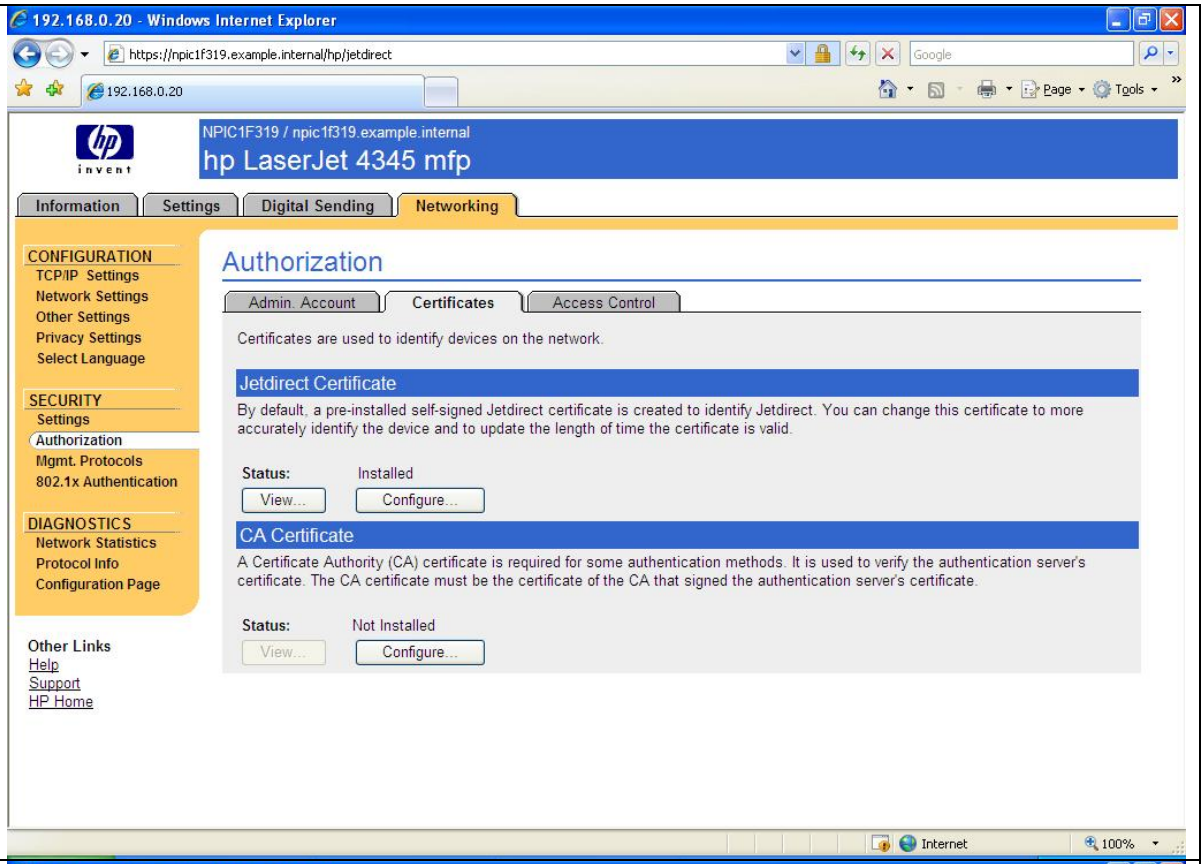
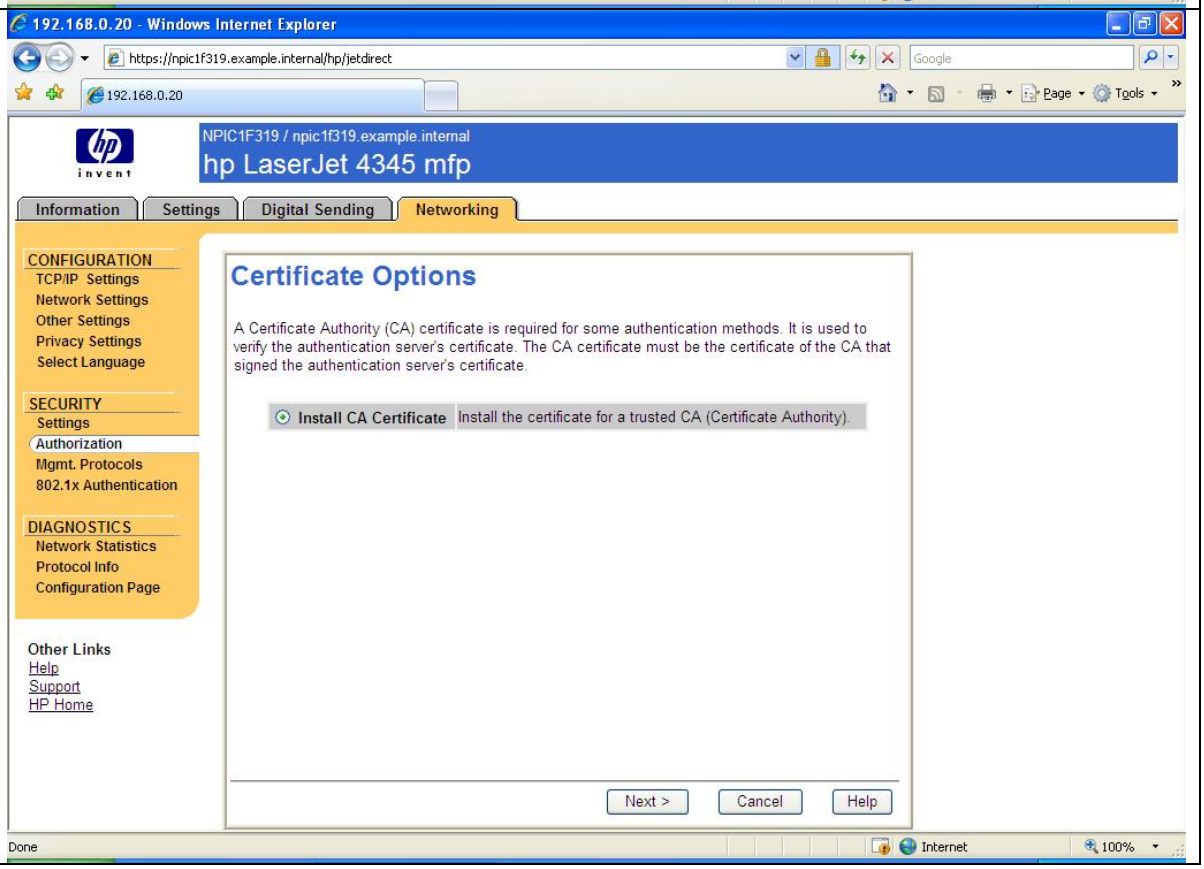


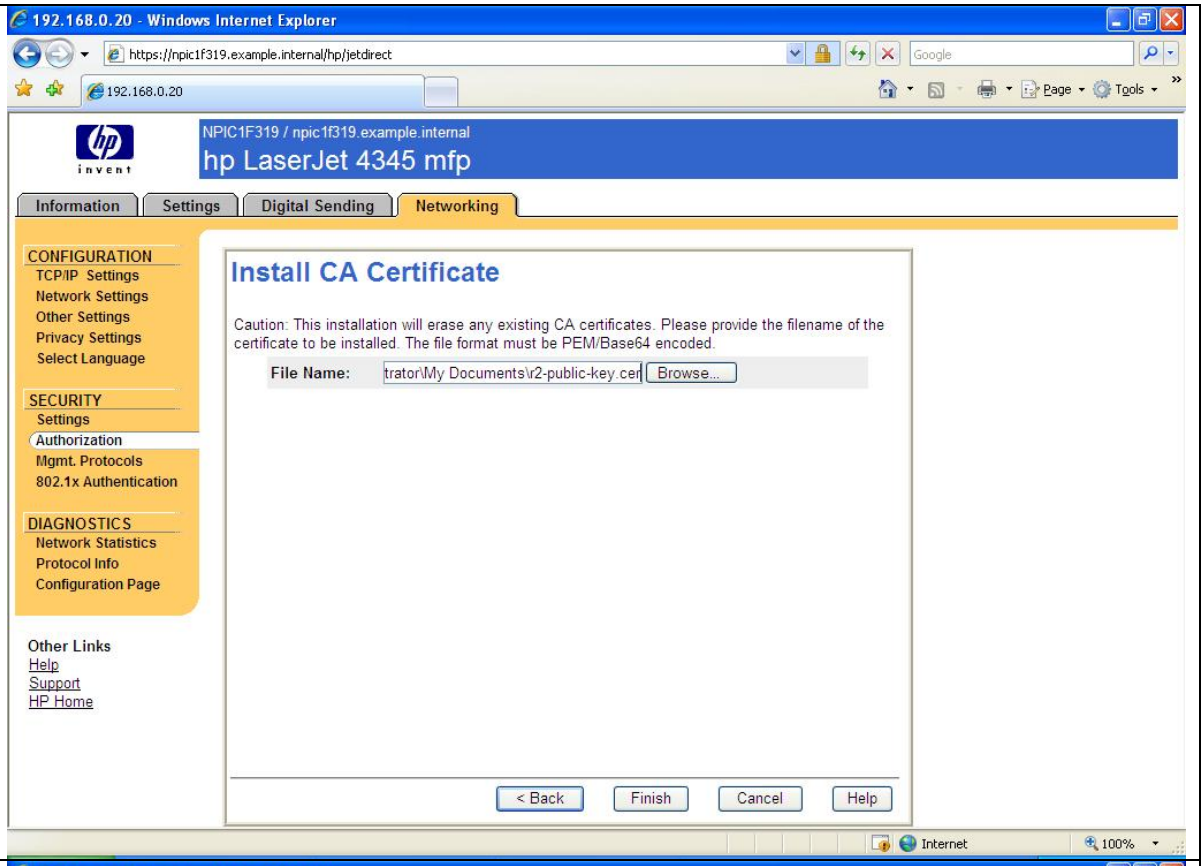
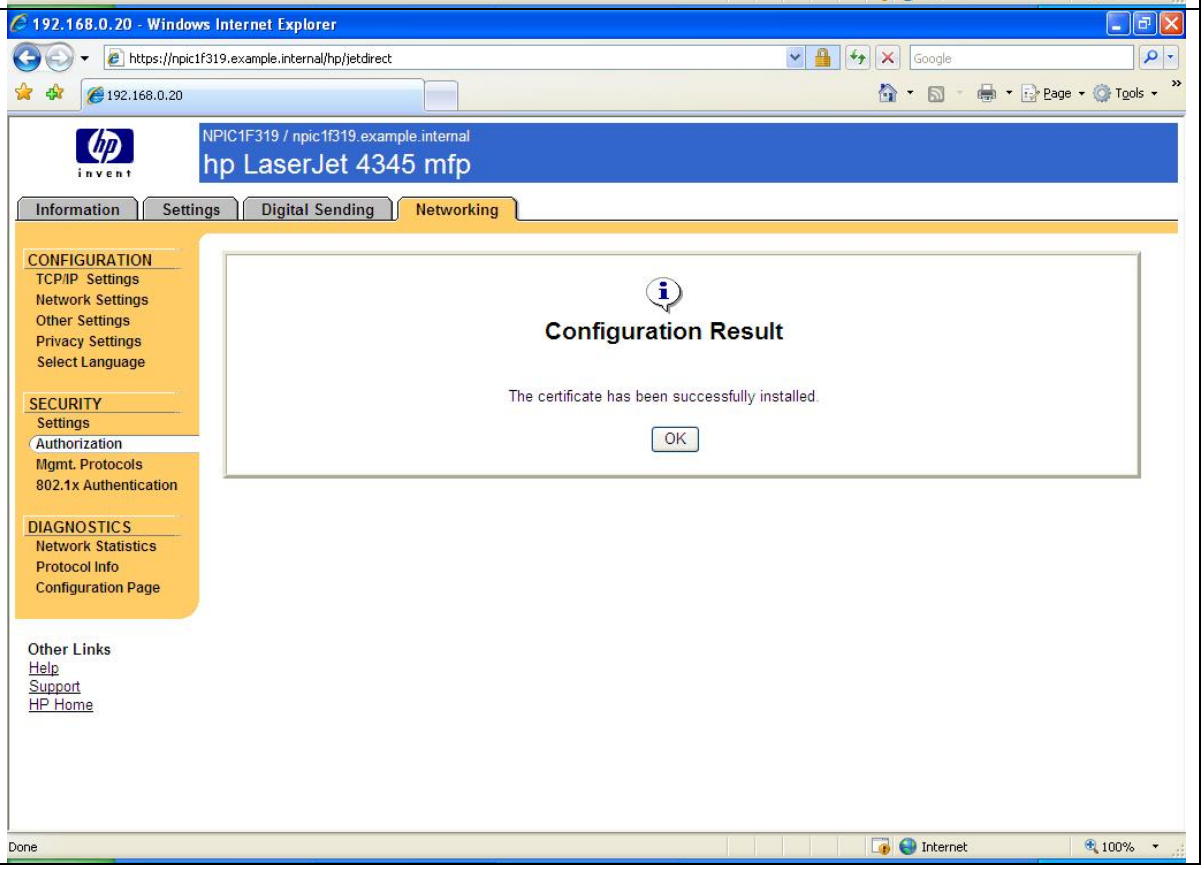
Because we have already stored the CA certificates in our browser's certificate store, we'll just export one and put it on Jetdirect. Let's take a look at it.	

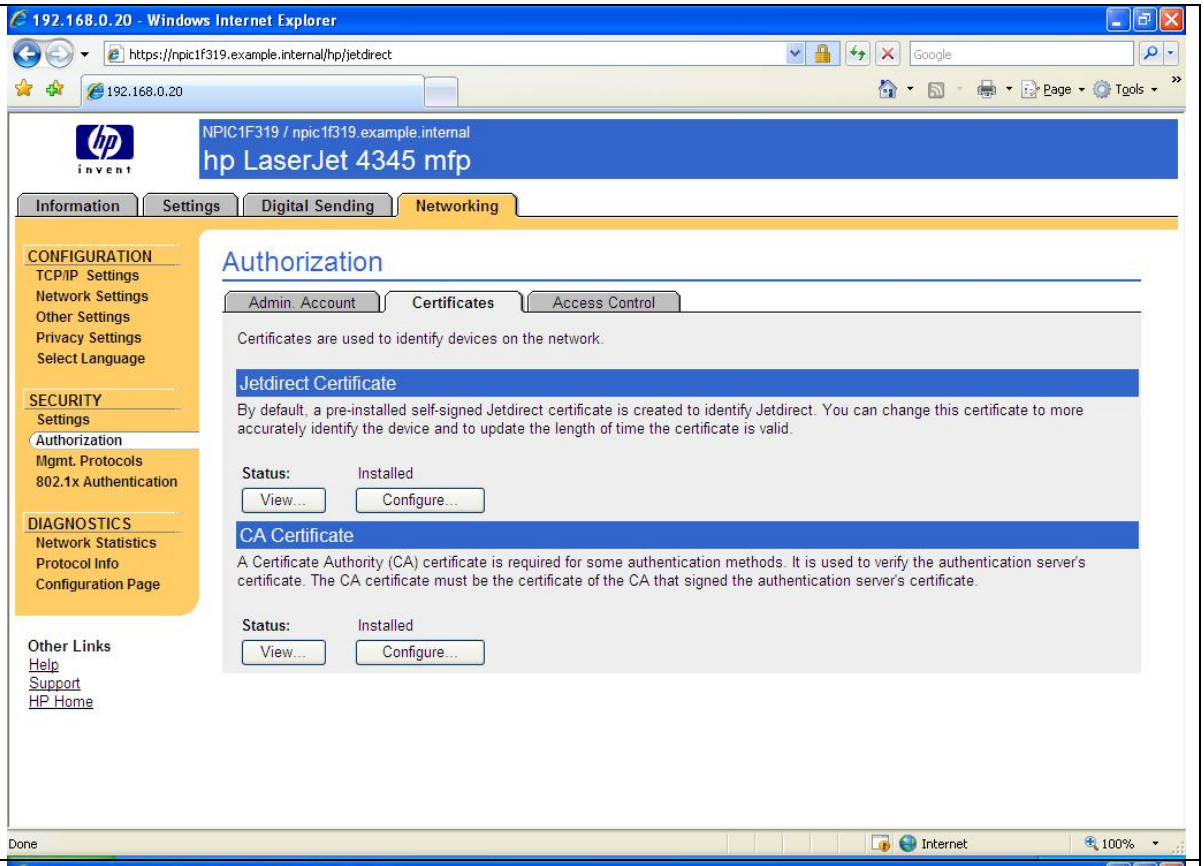
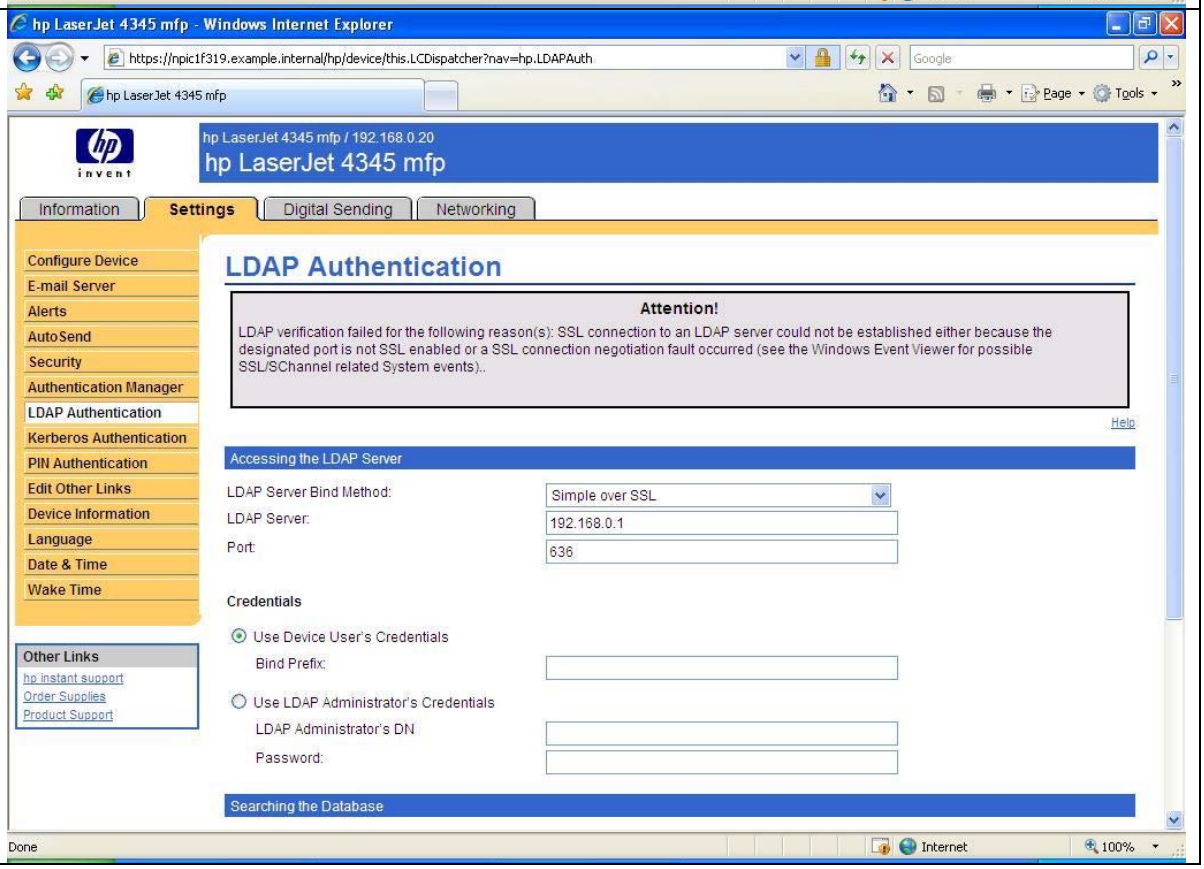
Select R2 and hit "Export…"	
Click Next	

Select DER. Click Next.	
Save it.	

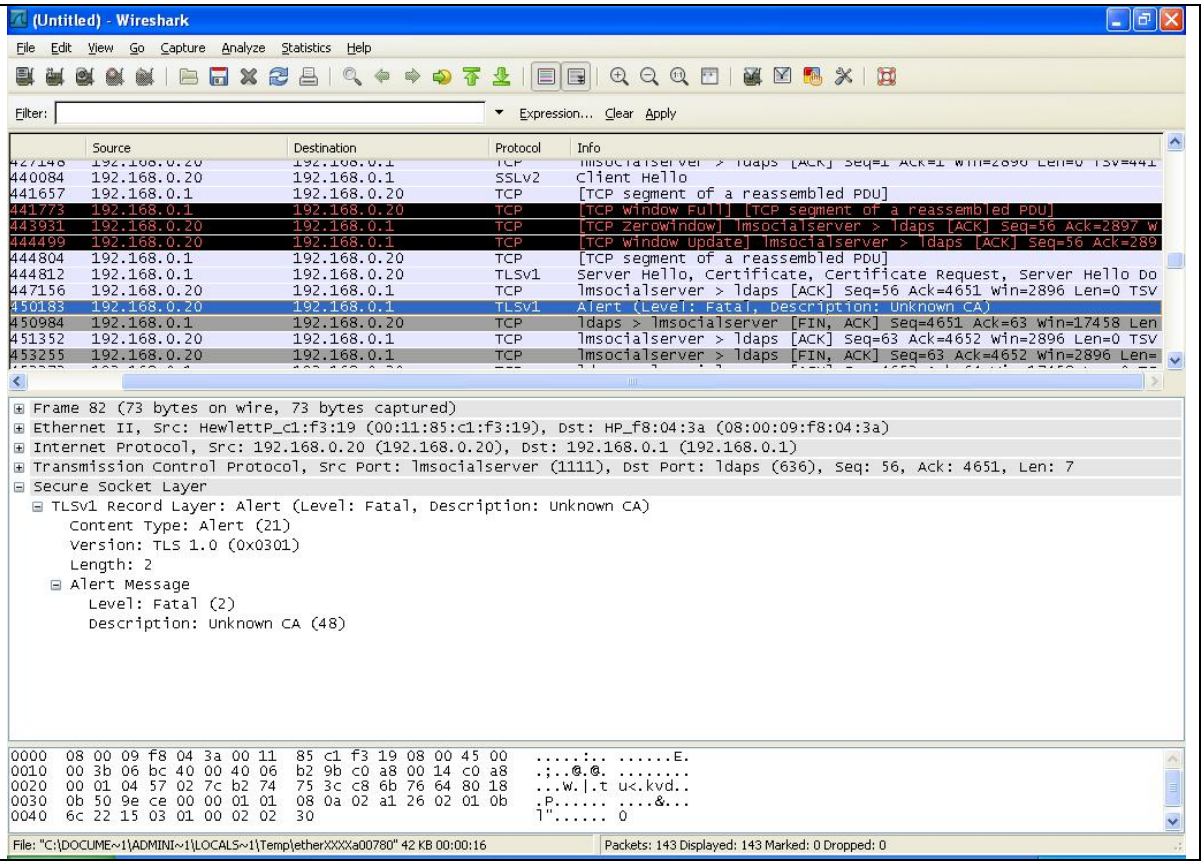
Save it.	
Click "Finish"	

Under the heading "CA Certificate", click "Configure"	
Select Install and click "Next"	

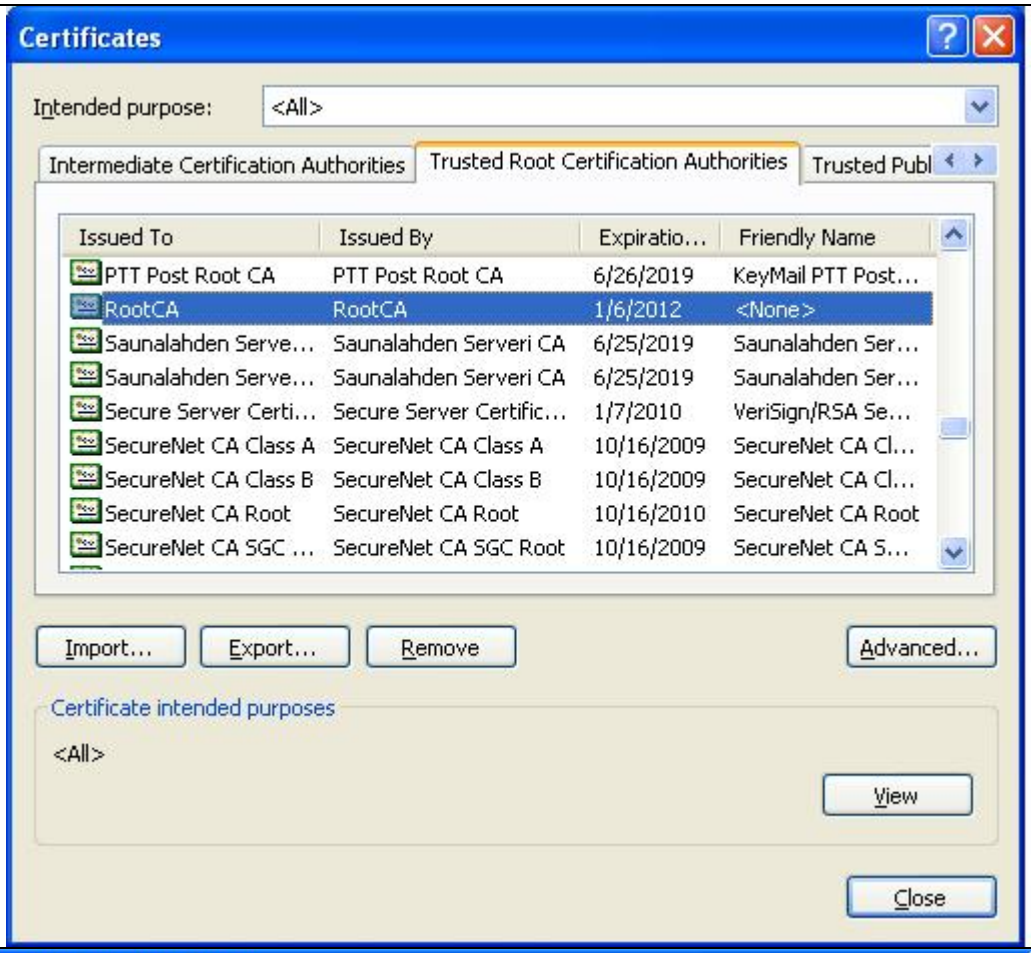
Select the file. Click Finish	
Click OK.	

The status for the CA Certificate is now "Installed"	
We try again and it still fails!	

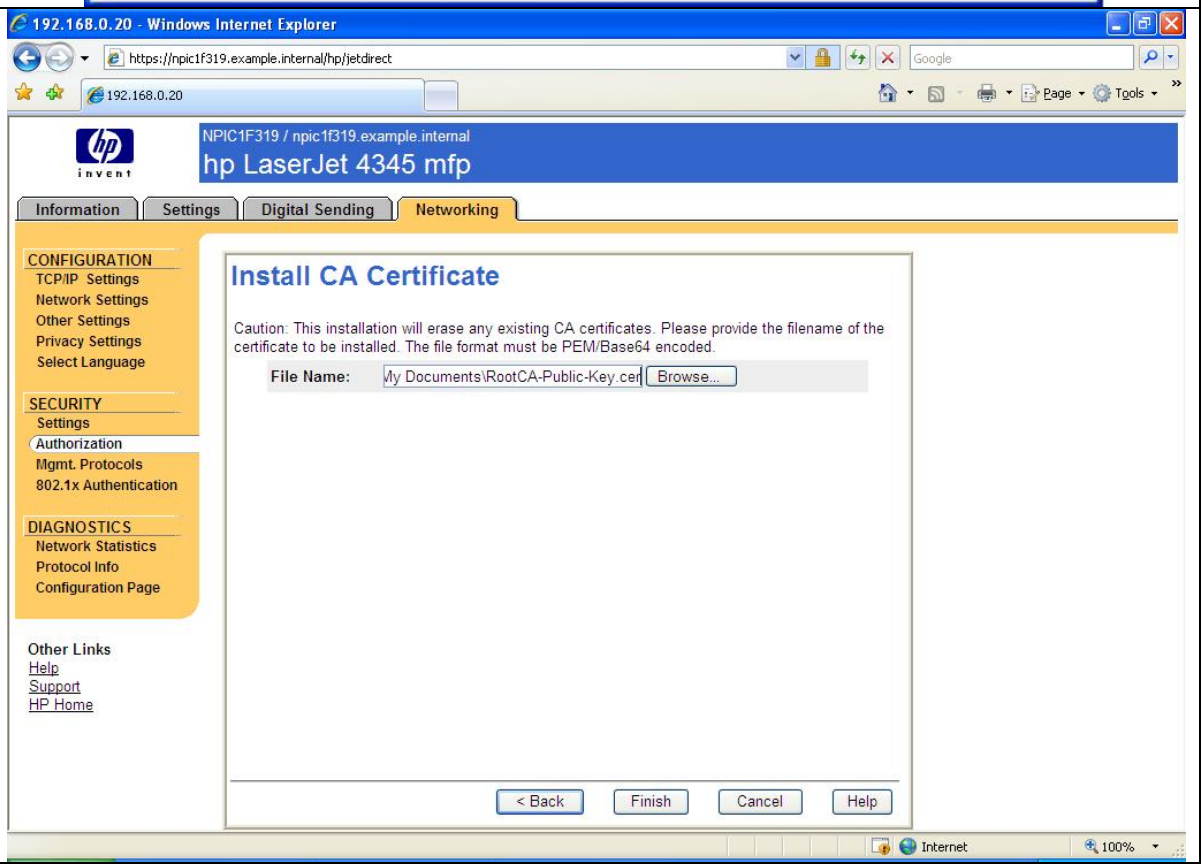
Same message. What did we do wrong?



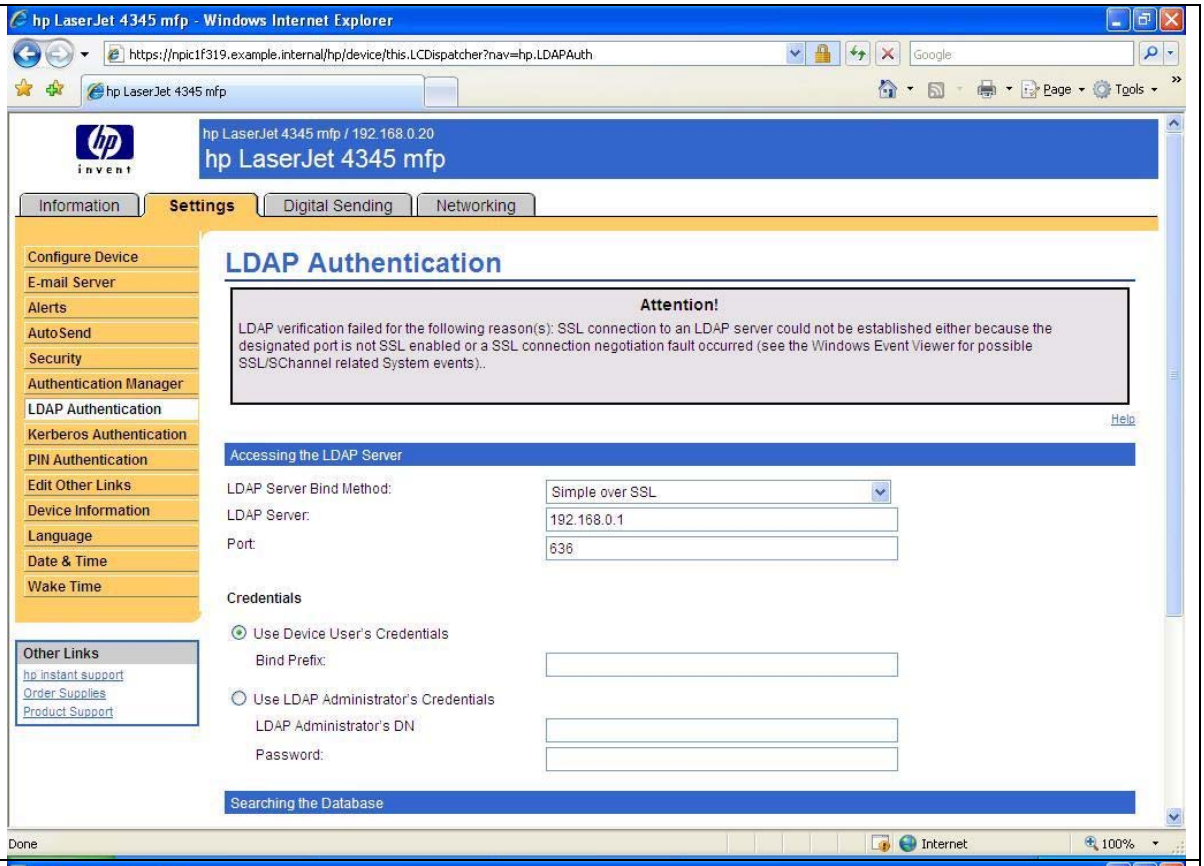
We need the ROOT CA.
Jetdirect cannot use Intermediate CAs.
Back to the certificate store and now let's export RootCA's public key certificate.



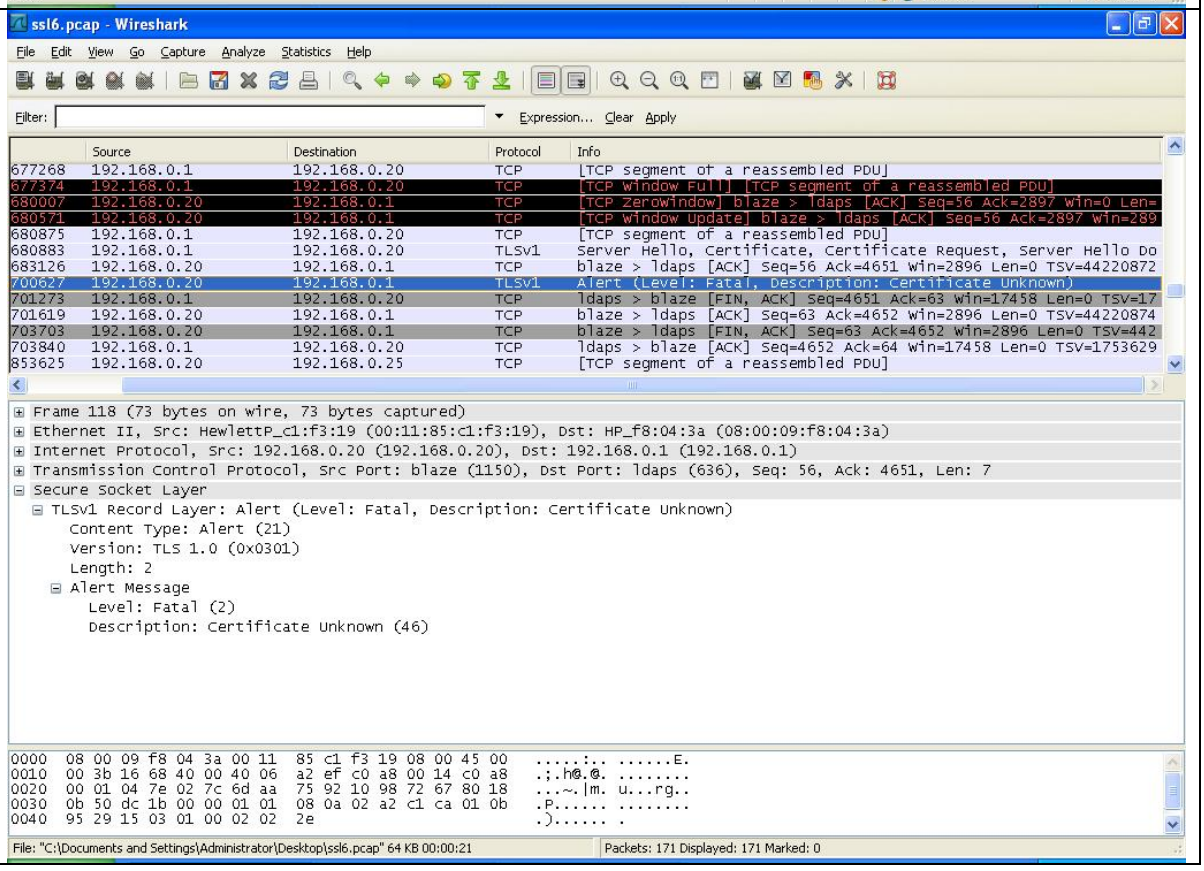
Install it.



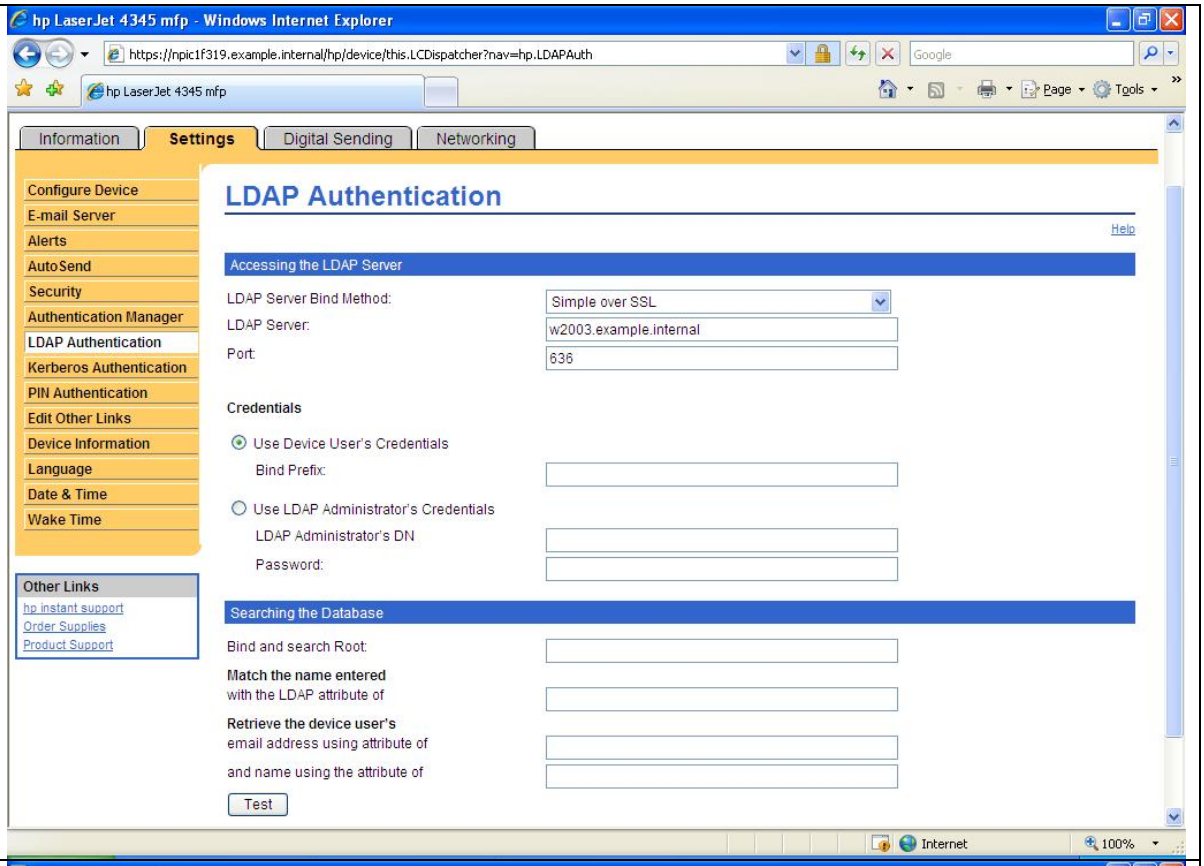
Try again. Another failure! Let's check the trace.



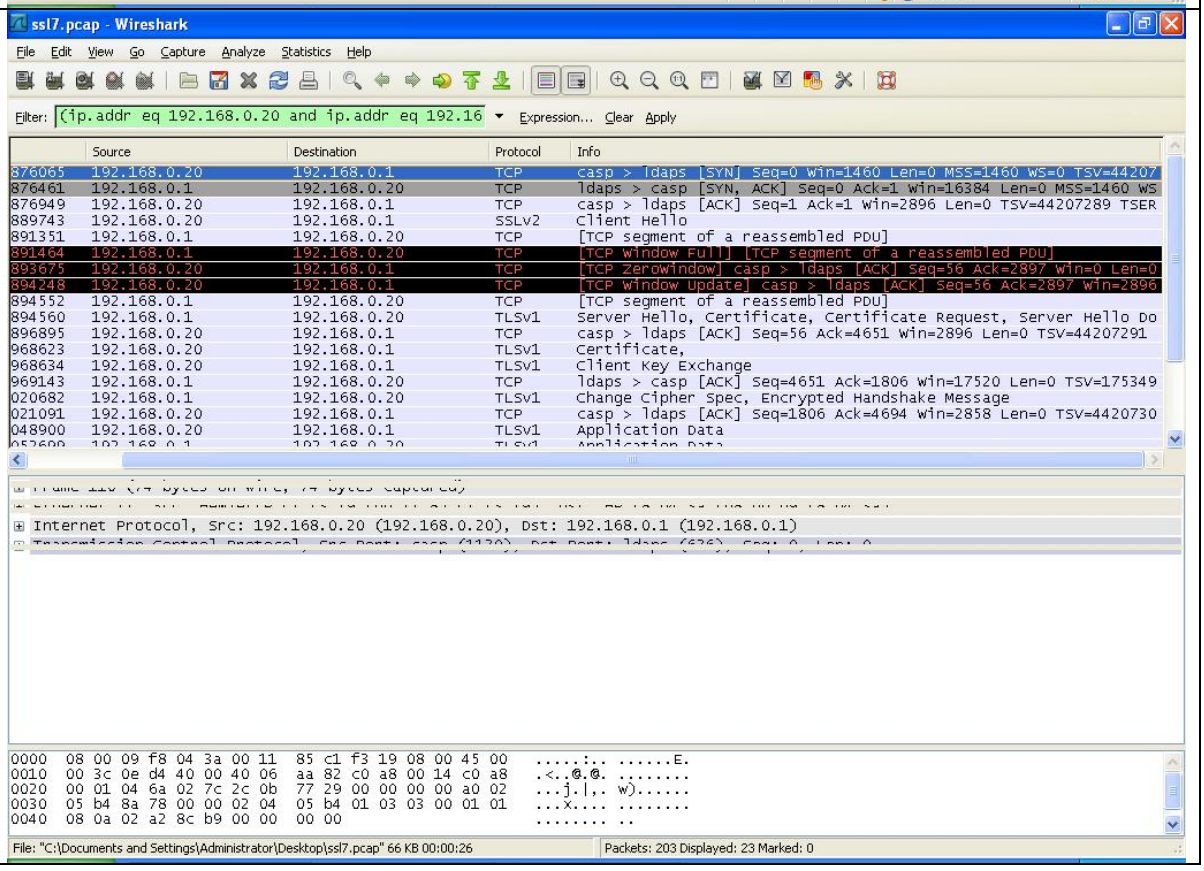
Here we get a "Certificate Unknown" message. Well, it could be we are using the IP address rather than the name. Let's check that.



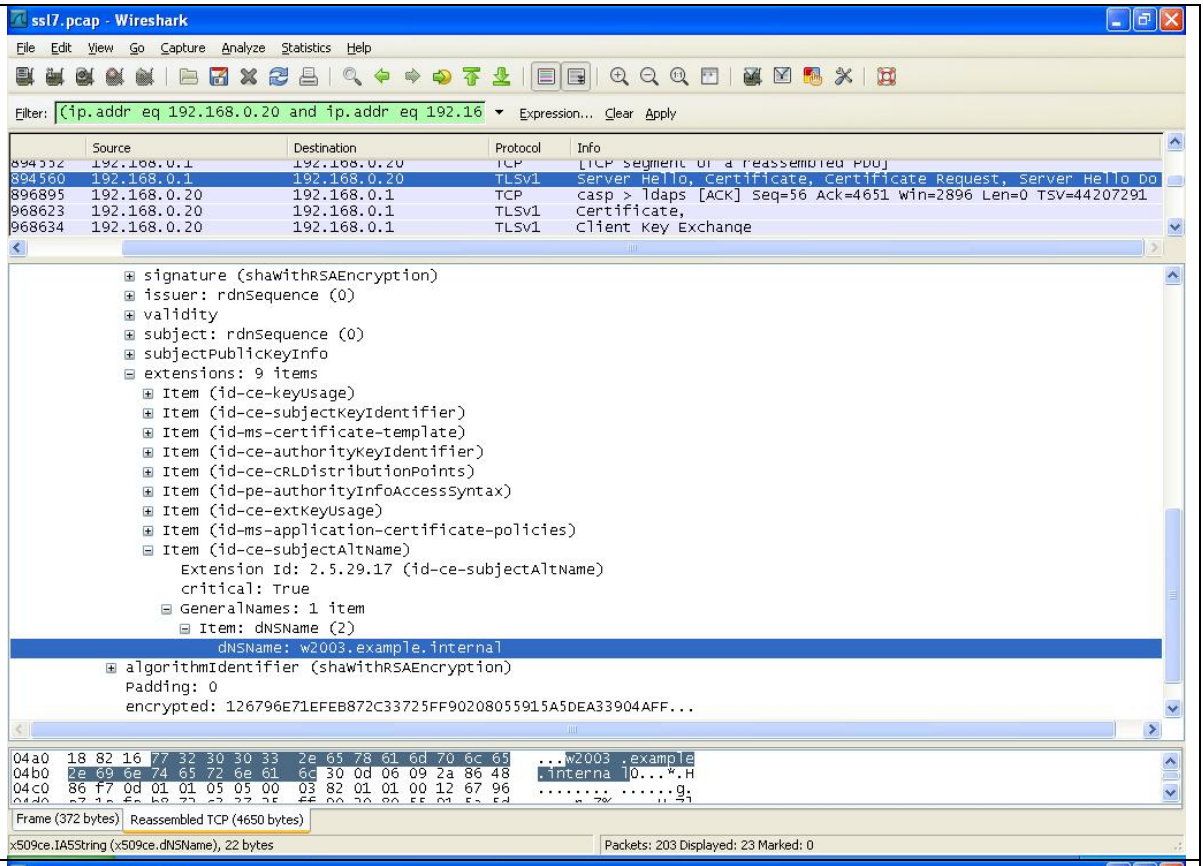
We use the DNS name and try again.



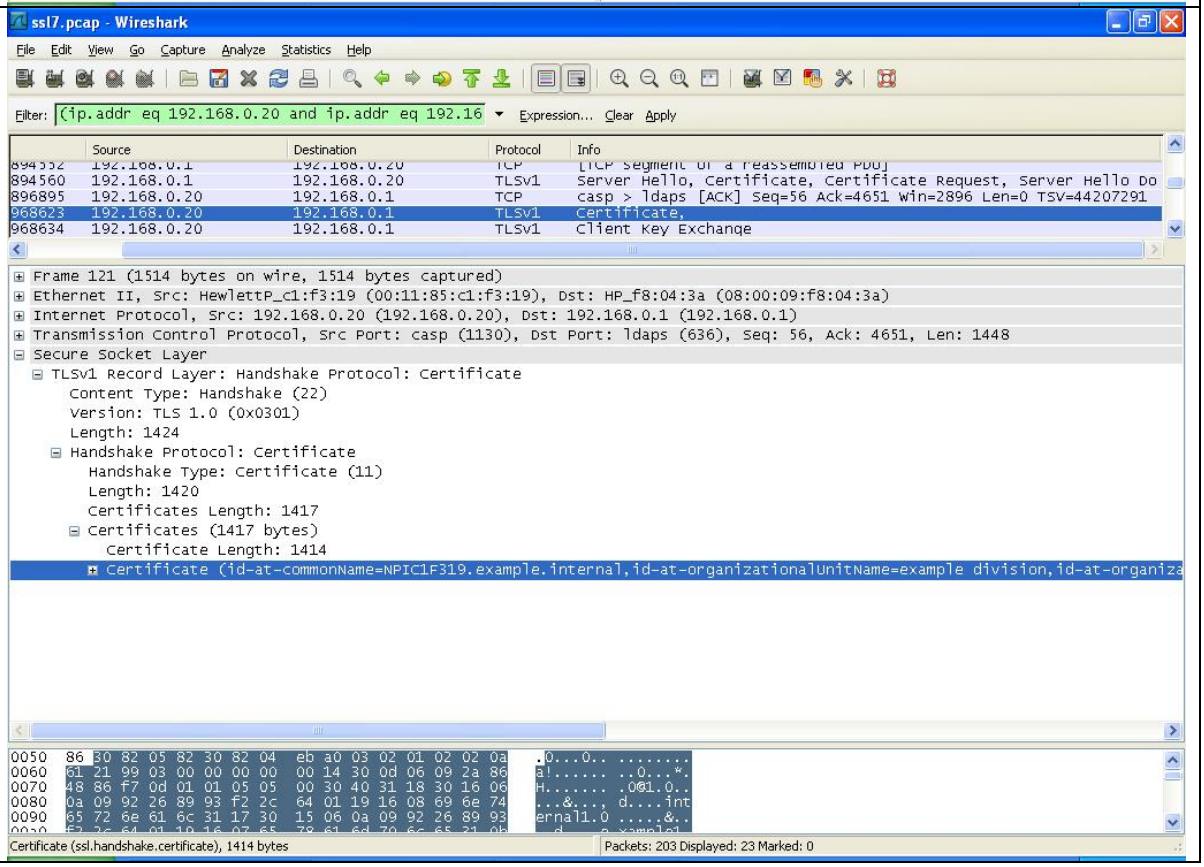
Success!!



Now that we are successful, we see the server's certificate has a SubjectAltName with a dnsName identifier.



Remember that the server wanted Jetdirect's certificate too. It sent us a "Certificate Request" and we sent back our Certificate just like we would do if we were a server. Now the server has to perform the appropriate certificate validity checks.



SSL/TLS Client: Understanding Certificate Chains

In the previous section, we described a situation where the wrong CA certificate was configured on Jetdirect. Let's explain this more thoroughly because it is a common issue reported on Jetdirect. Remember, Jetdirect is an embedded system and has limited flash space. Therefore, it cannot store a multitude of certificates on its flash file system. What Jetdirect needs to do is "Walk the Certificate Chain". Let's explain by reviewing our CA Hierarchy.

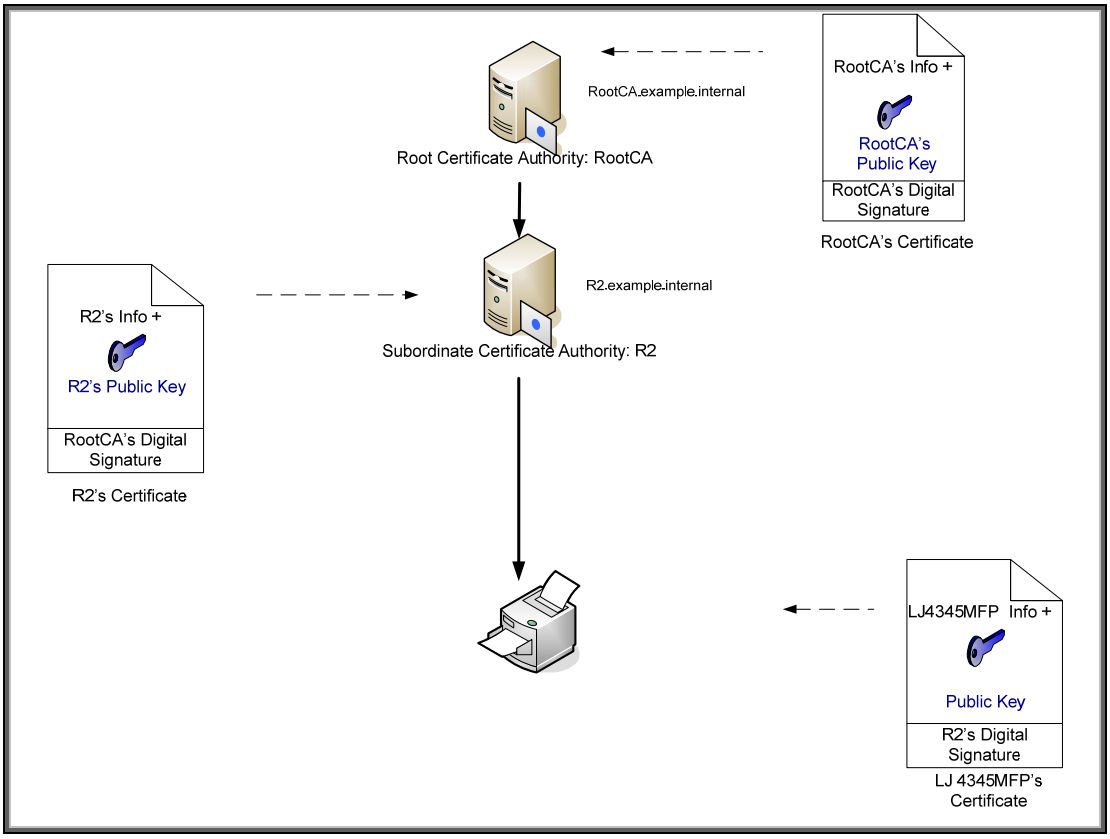


Figure 31 - CA Hierarchy

In this example, RootCA is the top level CA, which is also called the Root. What usually happens at customer sites is that the Root CA is created and it issues one or more certificates to Subordinate CAs, also known as Intermediate CAs, and they do the dirty work of issuing certificates to various entities in the customer's network. The Root CA is then shutdown and locked up in a secure room with this information backed up in several places. The Root CA establishes the trust of the whole environment and is very well protected.

We can see that RootCA issues a certificate to R2, which grants R2 the capability to issue certificates to other entities. R2's certificate is signed by the Root CA. R2 then can issue certificates to other devices.

Figure 32 -

Notice that R2's certificate is issued by RootCA. What does RootCA's certificate look like? Let's look at Figure 33.

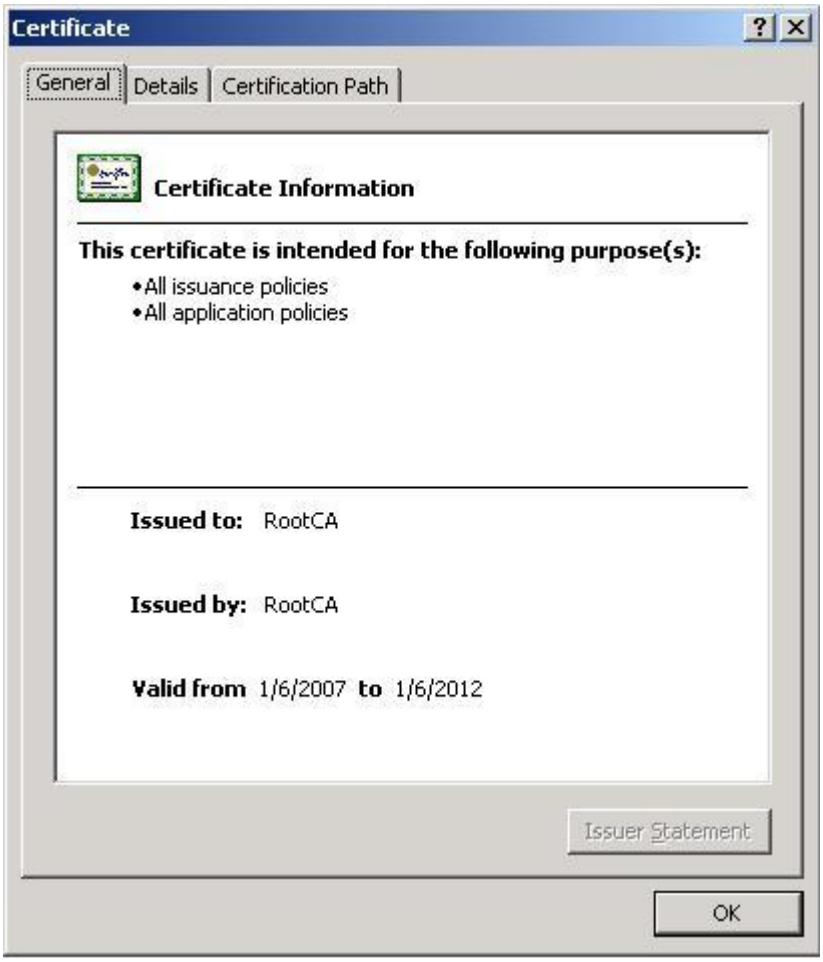


Figure 33 - RootCA

Notice the RootCA is "self-signed". All Root CAs will be self-signed – these CAs represent the single point of trust. A logical question would be: "Which CA do I configure on Jetdirect?" Let's look at some diagrams**. First, we have an incorrect configuration, as shown in Figure 34 – Incorrect HP Jetdirect CA Configuration**.

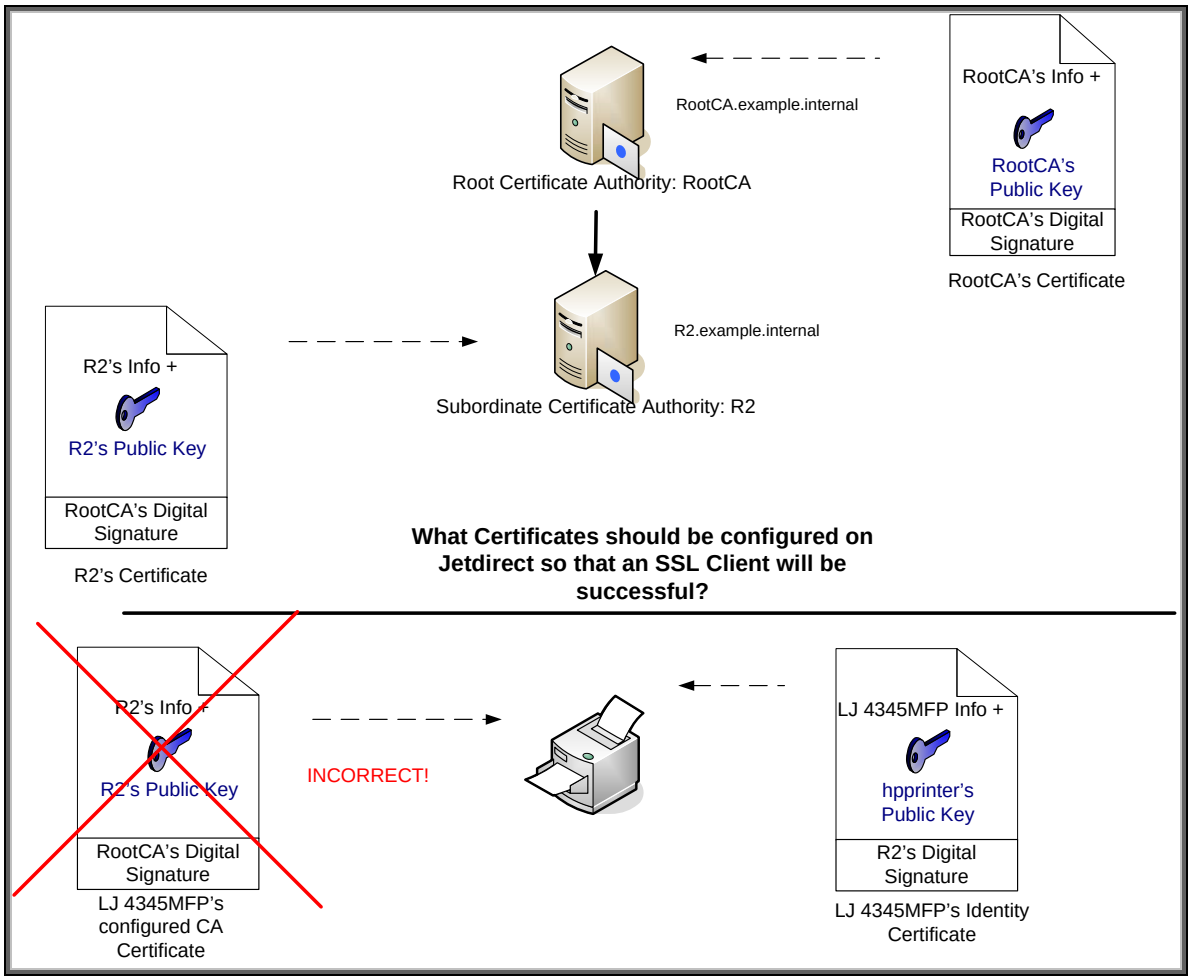


Figure 34 - Incorrect HP Jetdirect CA Configuration.

The Subordinate CA cannot be used as the CA certificate on Jetdirect!

Now we can look at a correct configuration in Figure 35 – Correct HP Jetdirect CA Configuration.

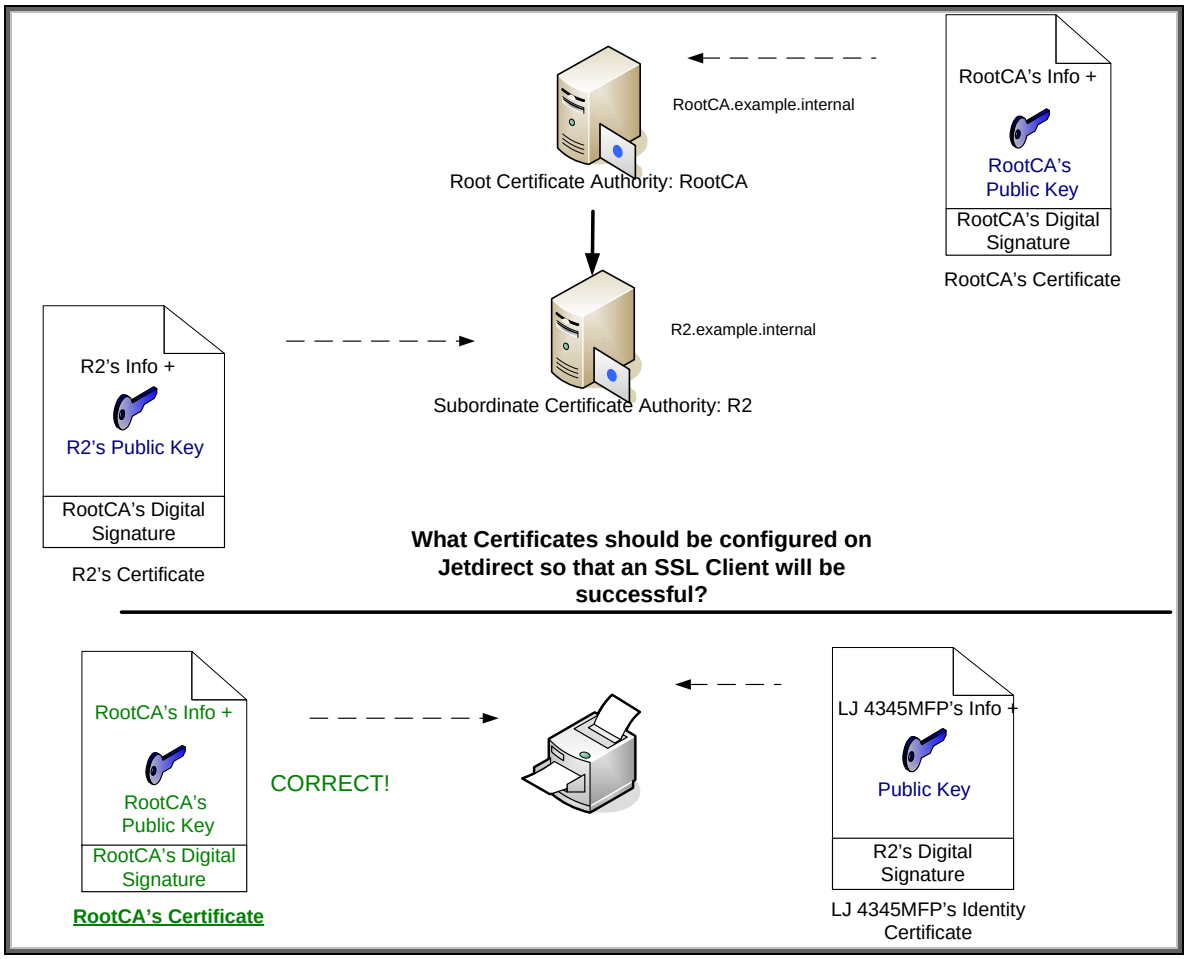


Figure 35 - Correct HP Jetdirect CA Configuration

Be sure the Root CA of your CA Hierarchy has its public key certificate configured on Jetdirect!

Here is a question for you: When Jetdirect is acting as a client and receives the server's certificate signed by R2, how can it know that R2's certificate was signed by RootCA? The answer: It cannot! Another special thing must happen: The server must send R2's CA certificate along with its own certificate. This allows Jetdirect to "walk the chain" and verify the certificate chain is valid. Refer to Figure 36 – Walking the Chain 1

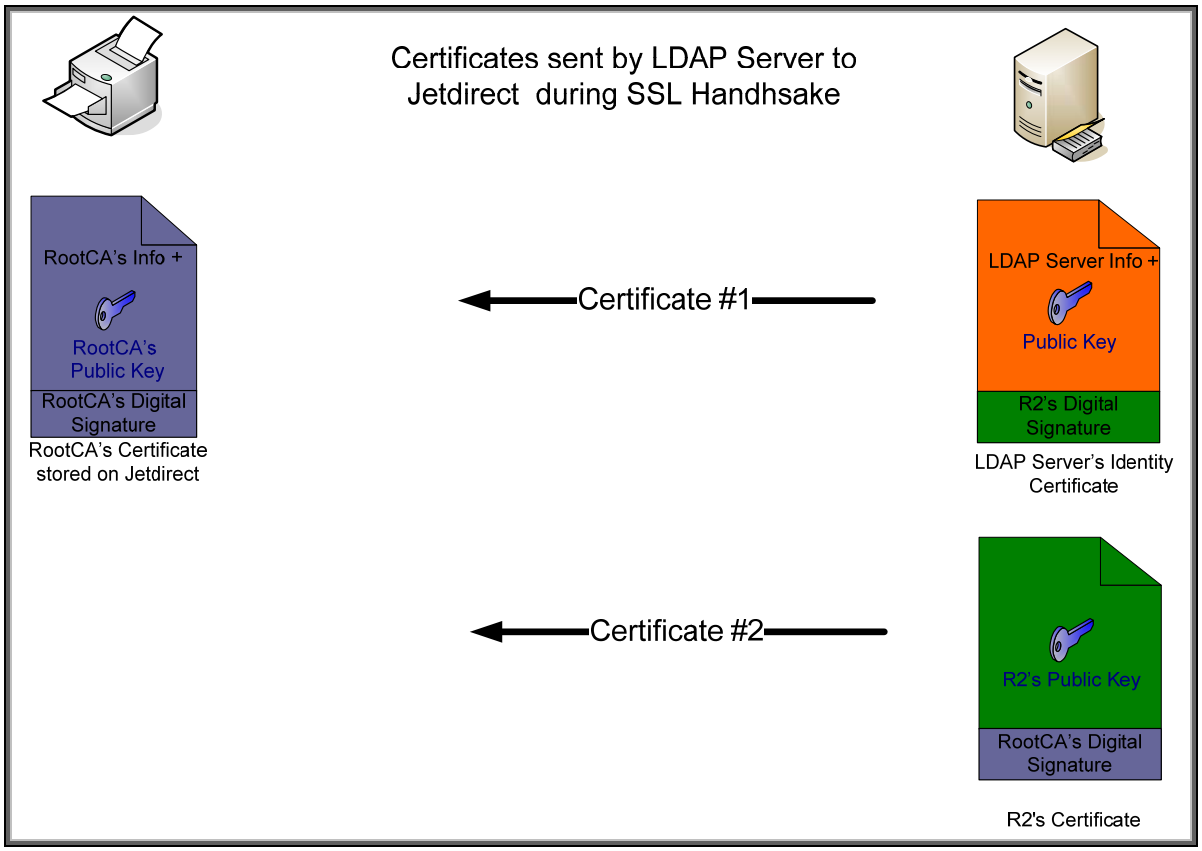


Figure 36 - Walking the Chain 1

Jetdirect has one certificate stored on it – the RootCA public key certificate. During the SSL/TLS handshake with the LDAP server, two certificates are sent back to Jetdirect. One is the LDAP Server's certificate and the other is the public key certificate for R2. Jetdirect stores them in its volatile memory and can begin to "walk the chain". Refer to Figure 37 – Walking the Chain 2:

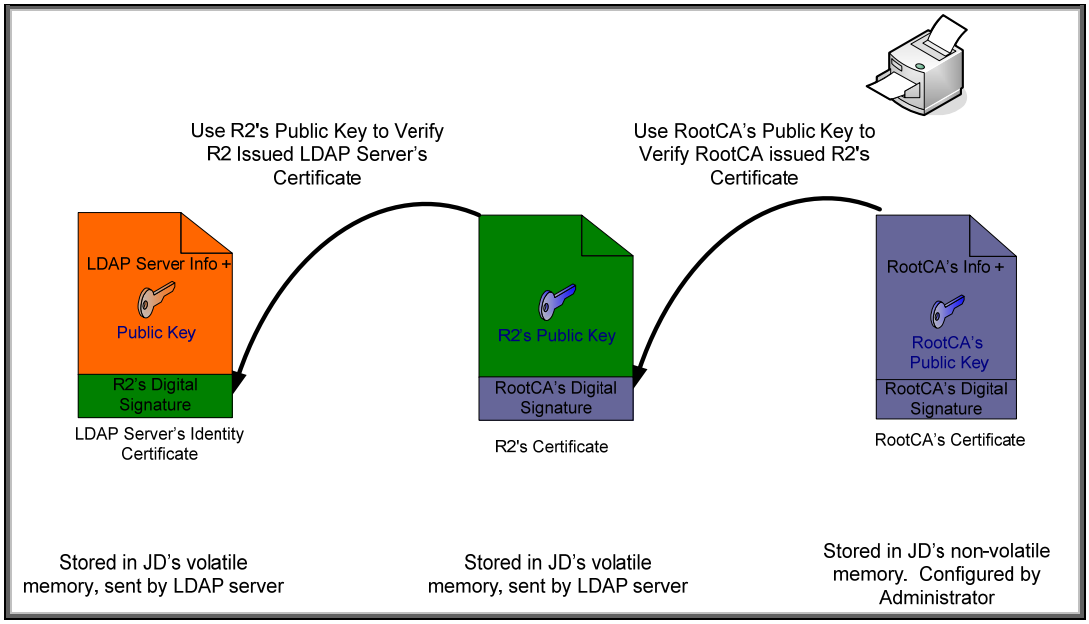


Figure 37 - Walking the Chain 2

Jetdirect verifies that R2 has signed the server's certificate. It also verifies R2's certificate (e.g., it has not expired and so on) and makes sure that R2's certificate was signed by RootCA. This "walking the chain" functionality is very important for devices with limited storage space for certificates – like HP Jetdirect.

SSL/TLS Client: Certificates and Name Verification

You may remember that having "https://192.168.0.20" in the URL of the browser resulted in Internet Explorer 7 reporting a certificate problem but that "https://NPIC1F319.example.internal" ended up with everything okay. How the SSL/TLS client authenticates the SSL/TLS server is very important and is unfortunately mired in practical deployment problems. We'll try to sort through it all!

The certificate itself has two very important fields that need to be discussed

- Subject
- SubjectAltName

The subject field is where the Common Name is stored. What is the Common Name? Well, that is a good question since it was never really properly defined. The most likely thing for HTTPS is that it is the Fully Qualified Domain Name. Let's look at a trace where a browser has established an HTTPS connection with the Jetdirect device. Refer to Figure 38 – Subject.

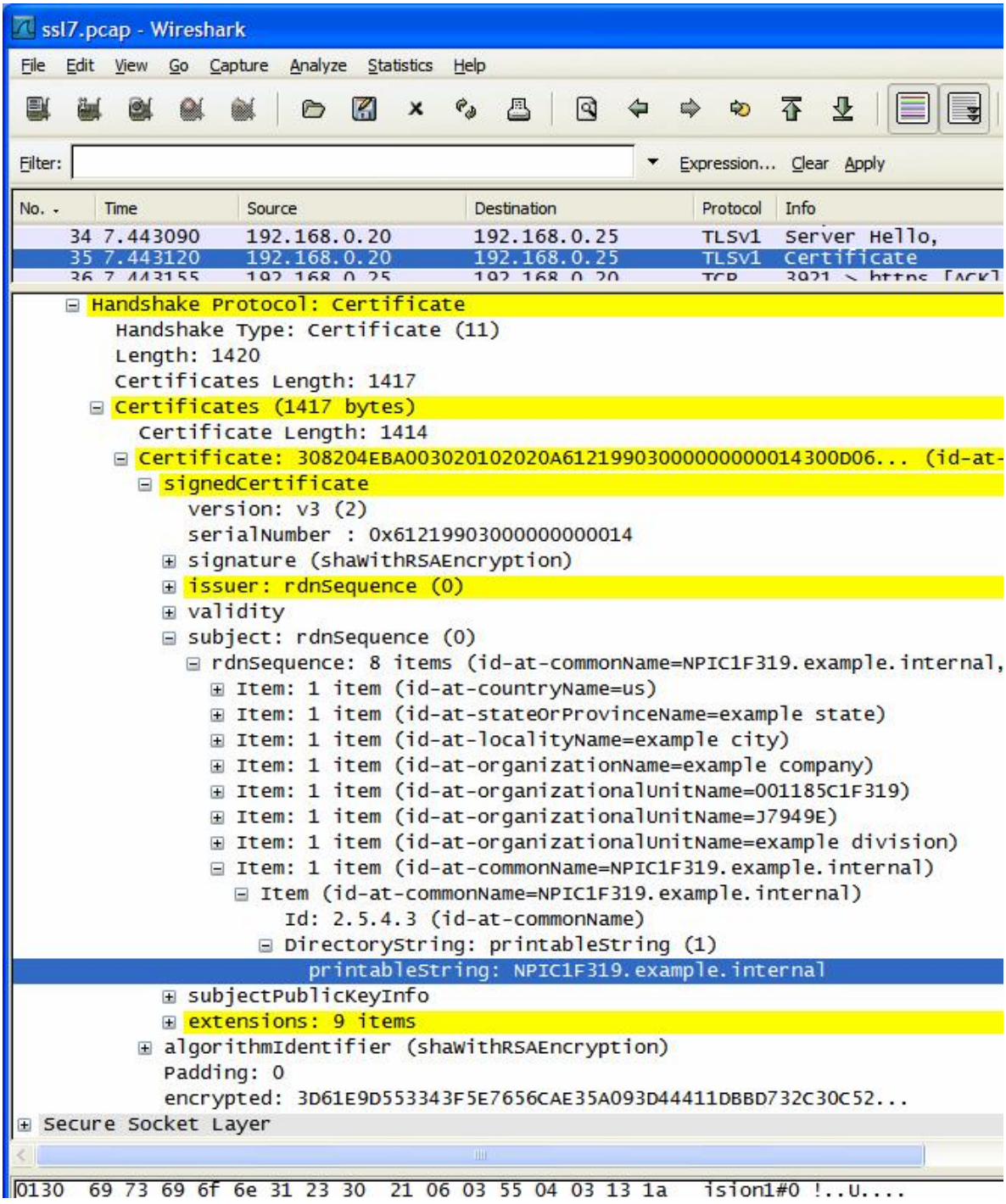


Figure 38 - Subject

We can se there are several things in the Subject – but the most critical is the Common Name. Here we can see why the browser URL of "https://192.168.0.20" would fail to pass the certificate check but "https://NPIC1F319.example.internal" would not fail. This interesting fact comes as a surprise to most people – the IP address is not usually part of the certificate (Note: IP addresses can be included in certificates). Another way of verifying the name is to use the SubjectAlternateName. To see this, we need to look at the trace of the LDAPS connection shown in Figure 39 – SubjectAltName.

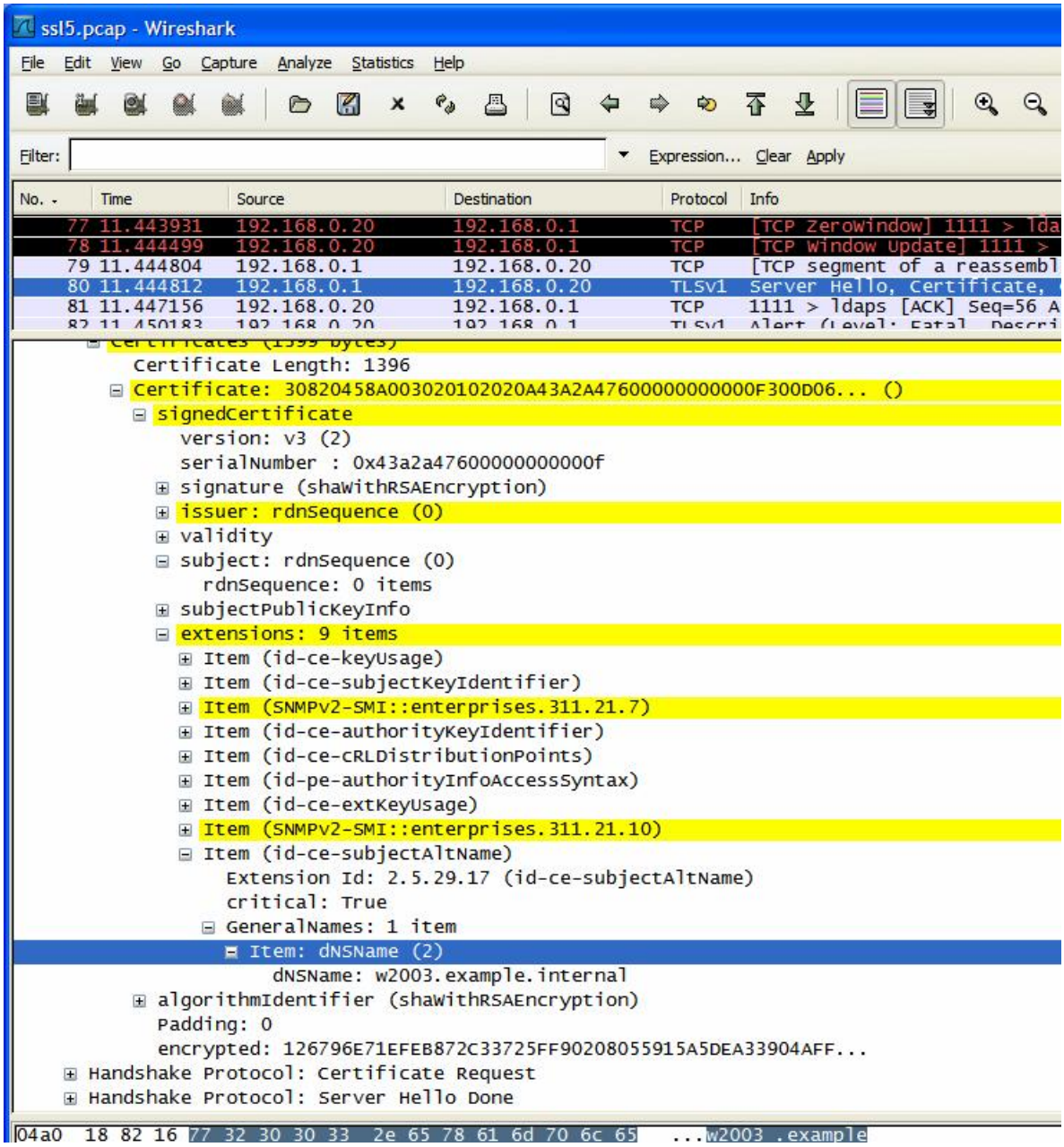


Figure 39 - SubjectAltName

Notice how there isn't even a Common Name in the LDAP server's certificate. If you remember, we tried connecting to the LDAP server using the IP address of 192.168.0.1 and it failed. When we switched to w2003.example.internal, it passed. We can now see why. A name check was done between the FQDN specified for the LDAP server and the SubjectAlternativeName of a type of dNSName whose syntax is very well known. The SubjectAlternativeName can contain multiple dNSNames and it can contain IP Addresses as well. As time goes on, the SubjectAlternativeName will be used more and more since its syntax better understood than the Common Name, at least for HTTPS. From RFC2818 which describes HTTP over TLS:*"If a subjectAltName extension of type dNSName is present, that MUST be used as the identity. Otherwise, the (most specific) Common Name field in the Subject field of the certificate MUST be used. Although the use of the Common Name is existing practice, it is deprecated and Certification Authorities are encouraged to use the dNSName instead."*

Effectively, the algorithm is going to be something like this:

```
If( dNSName is present)
{
        Match dNS Name
}
Else
{
        Match Common Name
}
```

For HTTPS, we saw that a mismatch caused a warning dialog box with Internet Explorer 6 and an explicit Certificate Error with Internet Explorer 7. Combining everything we have learned so far, we can form a very easy rule with SSL/TLS:

> One name to one IP Address to one port number identifies one certificate.

For instance, looking at the previous trace:

w2003.example.internal => 192.168.0.1 + TCP 636 => LDAPS certificate

That was easy, right? Well, things get more complicated due to a few factors:

- Server Farms – having multiple servers respond to one name
- Virtual Hosting – having one web site for many customers
- Limited IP addresses – public servers require public IP addresses
- SSL Certificates for the Internet cost money

Server farms are where I have several machines handling SSL requests in order to load balance. For example, if you do an nslookup on a major site, you may see more than one IP address. Going back to our LDAPS example, it would be something like this:

w2003.example.internal = 192.168.0.1, 192.168.0.2, 192.168.0.3

Each time the name w2003.example.internal is resolved to an IP address, a different IP address is used. This behavior allows for load balancing. If each IP address is a separate machine, a single certificate needs to be stored on multiple machines because of how the naming checks are done. When distributing the same certificate to multiple machines, there is a danger around private key protection. Alternatively, separate certificates can be used with the same name but a differing organizational unit so that they can be distinguished (if the CA supports issuing certificates in this form). For example, you can see how Jetdirect populates the organizational unit:

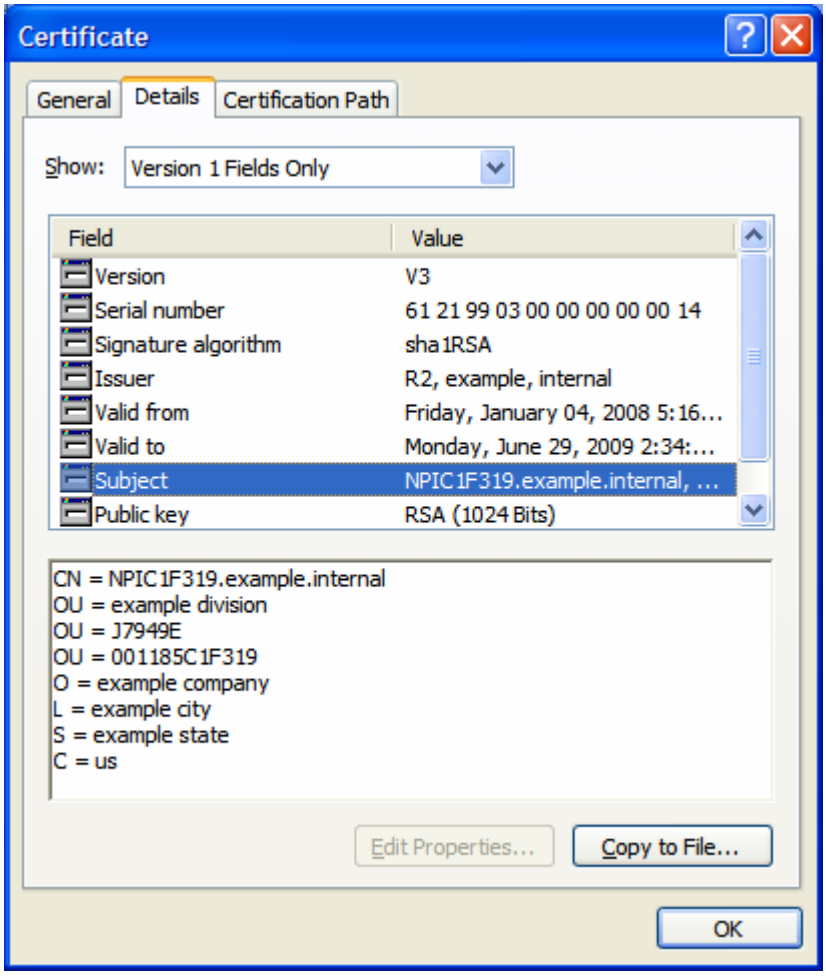


Figure 40 - OU

Here the Common Name is the FQDN of Jetdirect but there is additional information provided in the Organizational Units (OU). This same approach could be used for server farms where there would be several certificates with the same FQDN but differing in their OU values so that they will have separate public/private key pairs and provide better security over a single private key distributed to many servers. However, if the customer is cost sensitive to new SSL certificates, they may wish to take the risk on the private key being stored on multiple machines.

With Virtual Hosting, you have the opposite problem: Many names but only one IP address. This causes a lot of grief, especially for those customers that have problems with getting a valid IP address as well as those who are cost sensitive and require SSL certificates that can be used on the Internet. Here is an example: Let's assume that you are running a garage sale site on the Internet that allows clients to sign up and sell the stuff they don't need that is taking up space in their garage. Each user gets their own domain name. You want to use SSL to provide security. For instance, if the site is "example.com" at 192.168.0.250, each user would have something like this:

- hsimpson.example.com maps to 192.168.0.250
- msimpson.example.com maps to 192.168.0.250
- bsimpson.example.com maps to 192.168.0.250

Each person gets their own SSL certificate that has the SubjectAlternativeName set to their corresponding FQDN. Unfortunately, when "msimpson" and "bsimpson" try to use HTTPS,

("https://msimpson.example.com" or "https://bsimpson.example.com"), they get a Certificate Error indicating a name mismatch. Why?

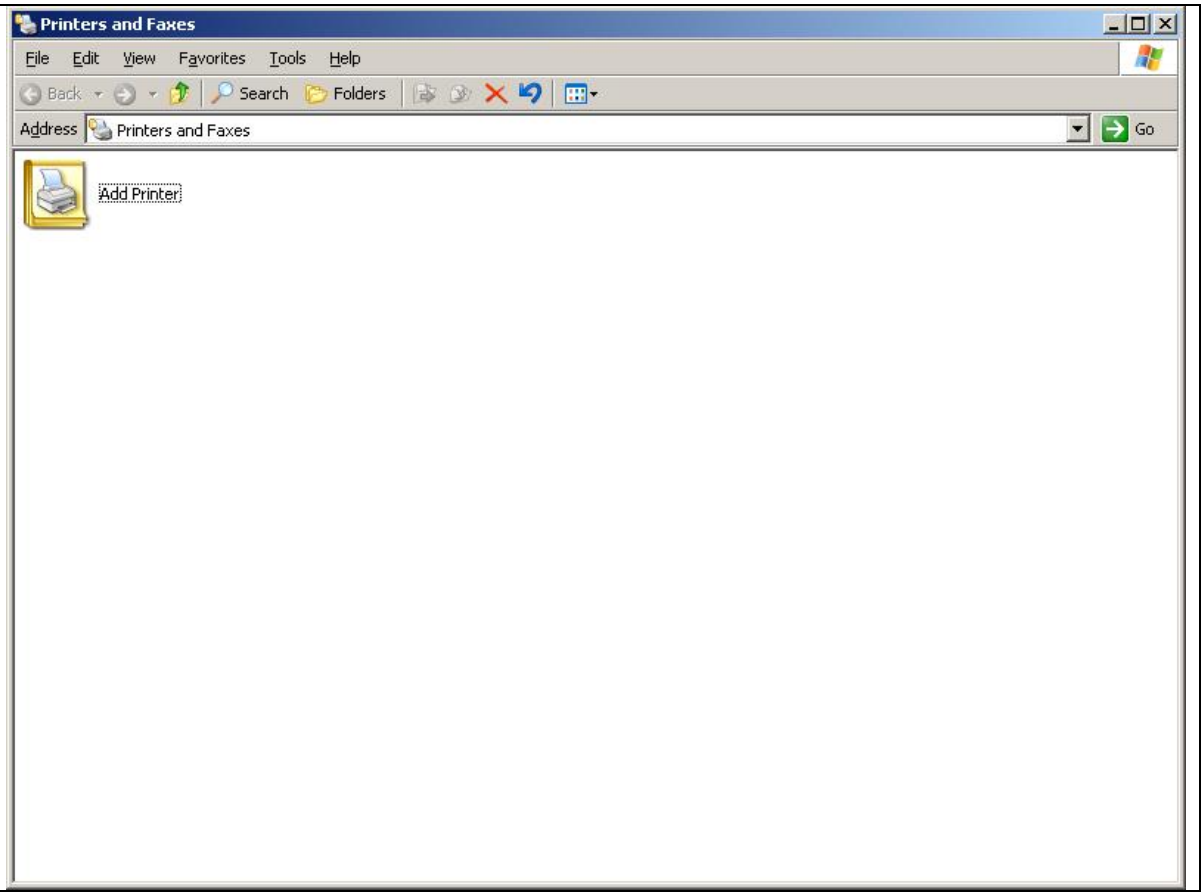
If we refer back to our SSL/TLS protocol diagrams, we can see that the server must send back a certificate before it knows what website is being referenced. This is part of the SSL/TLS negotiation. It happens before HTTP can be used to indicate the website (via the 'Host:' header). There are a few ways to try and fix this problem.

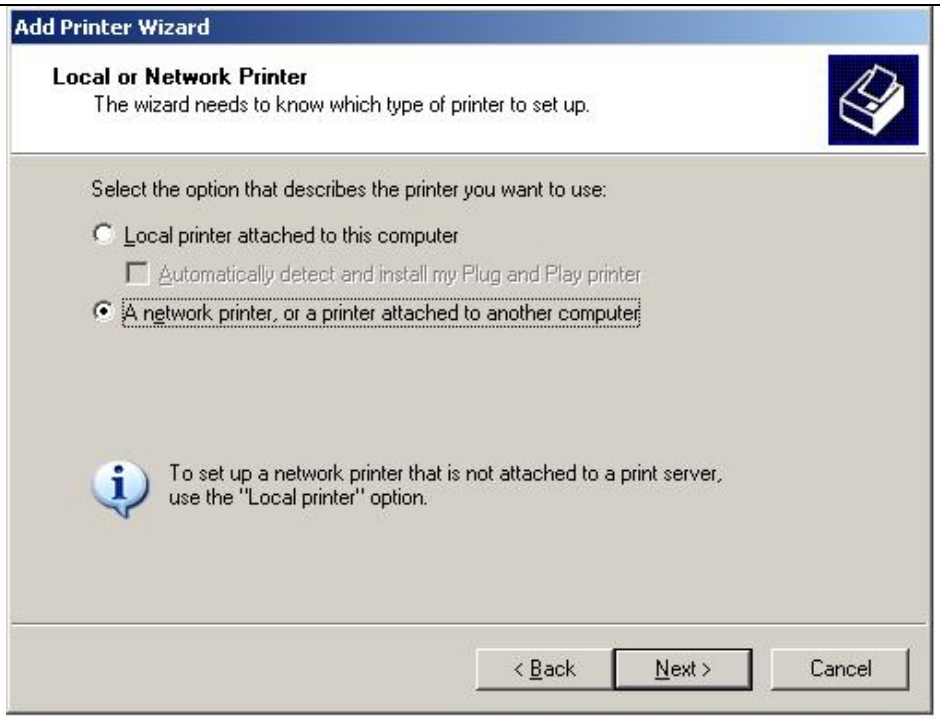
- A single certificate with multiple DNS names as part of the SubjectAlternativeName field. The security of the private key isn't a concern because we are still only on one machine. The problem with this approach is that when "lsimpson.example.com" wants to advertise on the website, a new certificate must be issued.
- RFC4366 allows for the ClientHello in the SSL/TLS handshake to indicate the server name field. Support of this feature has been limited, but is becoming more supported as time goes by. However, it offers the tremendous benefit of being able to use separate certificates for each user, which is an increase in security (and in increase in the cost associated with each certificate). The security vs. cost tradeoff would have to be weighed.
- RFC2817 allows HTTP to be upgraded to use TLS which would also solve this problem because it allows for HTTP to specify the server name of interest first and then upgrade to run HTTP over TLS. An interesting idea but support doesn't seem to be there for this feature in actual customer deployments.
- Wildcard Certificates. This is where the Common Name or the DNS SubjectAlternativeName contain an asterisk (*) to specify a wildcard match. For instance, in our example, the SubjectAlternativeName could be wildcarded as "*.example.com". Many browsers support the wildcard, unfortunately in a variety of ways, and then a single certificate can be used to support a variety of names. Unfortunately, wildcard support also depends on the protocol being used (e.g., LDAPS versus HTTPS).
    - RFC4513 which specifies LDAP authentication methods offers this guidance: *"The '*' (ASCII 42) wildcard character is allowed in subjectAltName values of type dNSName, and then only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject *.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com."*
    - RFC2818 which describes HTTP over TLS specifies: *"Matching is performed using the matching rules specified by [RFC2459]. If more than one identity of a given type is present in the certificate (e.g., more than one dNSName name, a match in any one of the set is considered acceptable.) Names may contain the wildcard character * which is considered to match any single domain name component or component fragment. E.g., *.a.com matches foo.a.com but not bar.foo.a.com. f*.com matches foo.com but not bar.com."*
    - When presented with such confusion, we should refer to the RFC that controls the format of the certificate, RFC3820 defines that format. RFC3820 does not mention anything about wildcard characters in the Subject Field. It does mention it for the SubjectAlternativeName field: *"Finally, the semantics of subject alternative names that include wildcard characters (e.g., as a placeholder for a set of names) are not addressed by this specification. Applications with specific requirements MAY use such names, but they must define the semantics."* Hardly a ringing endorsement!

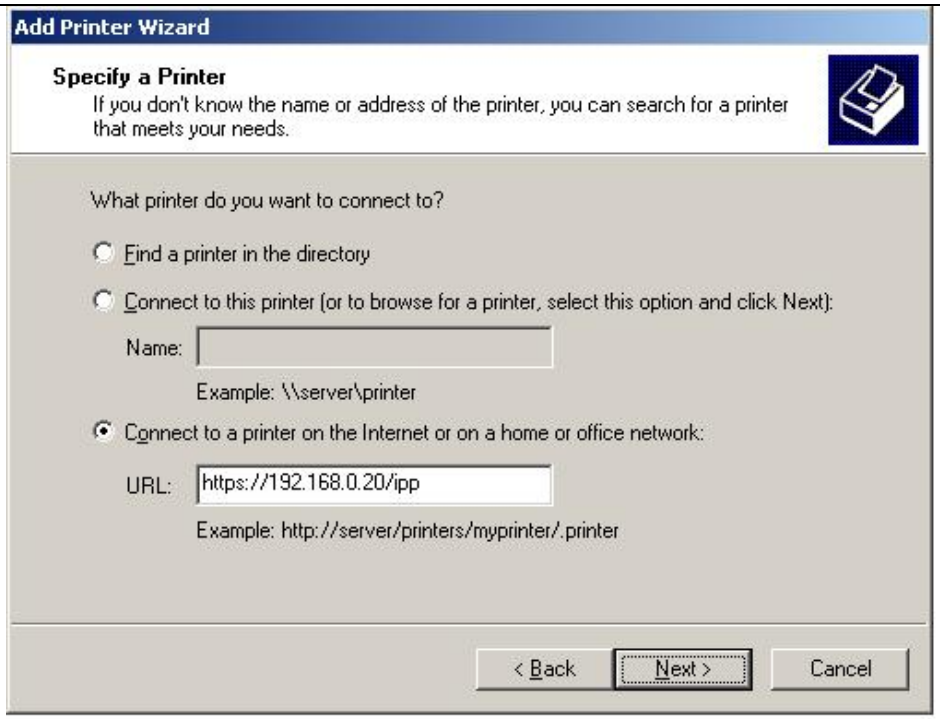
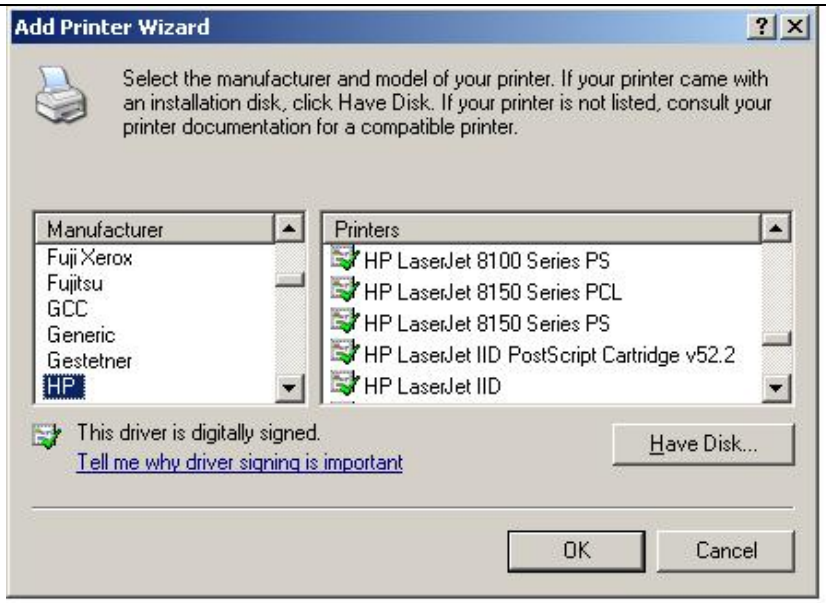
Why are we talking so much about name validation? Well, it is a common complaint from customers and it is extremely important that everyone understand the types of checks that go on.

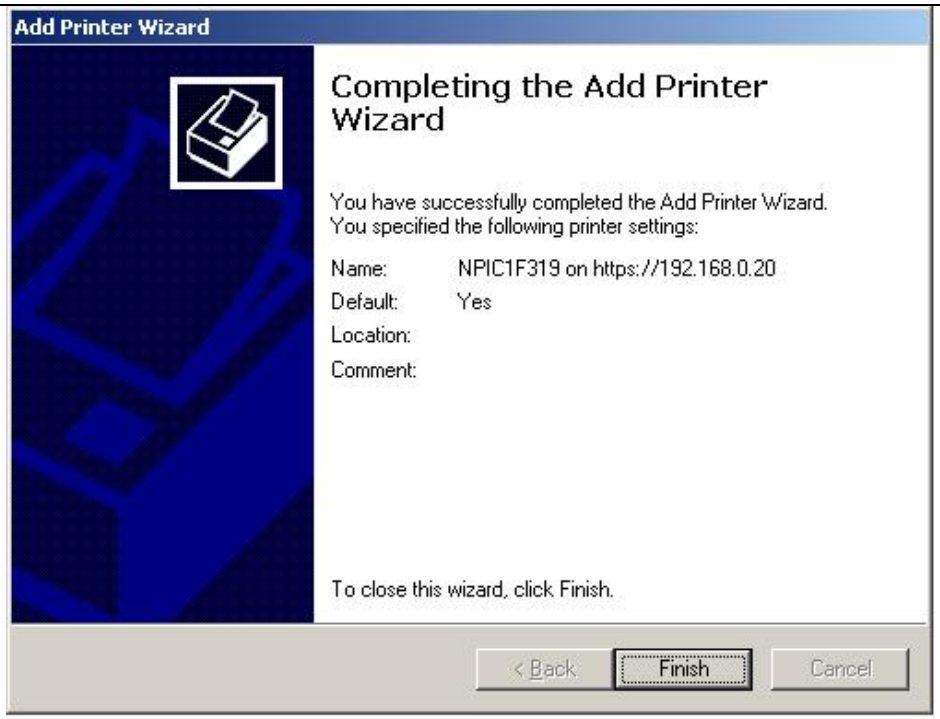
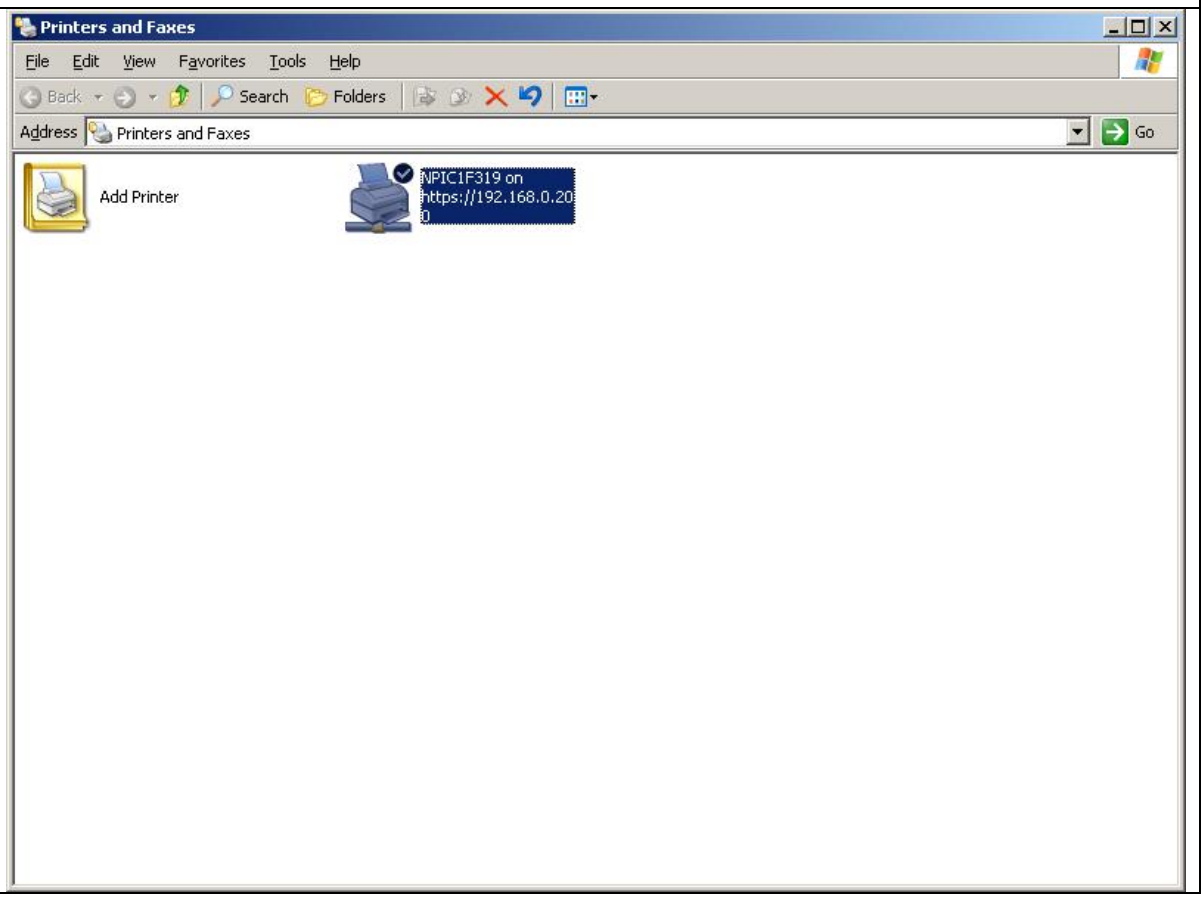
IPP over SSL/TLS

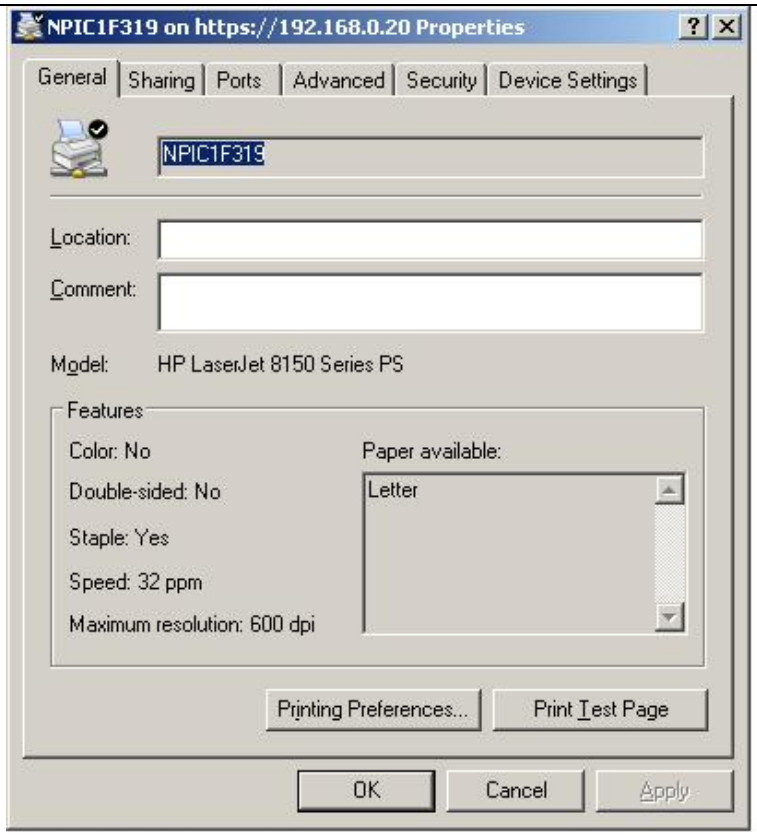
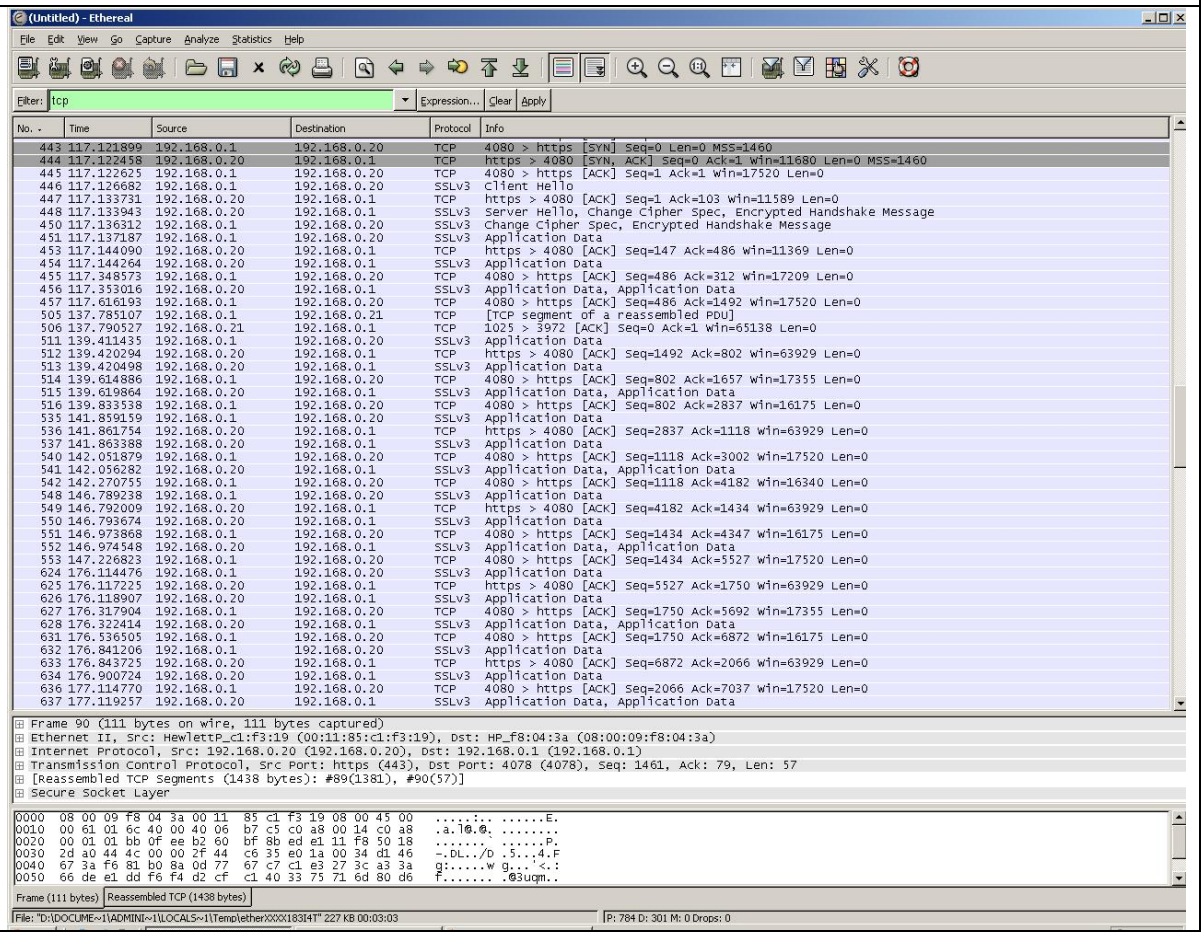
SSL/TLS can also be used to protect printing. HP Jetdirect supports IPP over TLS (henceforth, IPPS), but does not support any client authentication to control printing. Therefore, only server side authentication using the digital certificate can be done. Using the same Jetdirect we've been using so for, here is how to set it up on Windows XP:

Start with the Printers and Faxes folder and click "Add Printer"	

Click “Next”	
Select “A network printer…”	

Specify a URL of HTTPS and be sure to end it with a "/ipp" so Jetdirect knows what it is for.	
Select the appropriate driver.	

Click “Finish”	
Now we have a printer. Right Click and select properties.	

Print a test page.	
Yep – we have our print data protected by SSL/TLS.	

That wasn't too bad to get security for your print data. However, there is a problem. Notice that we used the IP address in the URL. After the big section on name checking, we should know that there is not a way to verify our Jetdirect's certificate with an IP address in the URL. IPPS should have flagged a certificate error but it did not.

This behavior brings up an important point. Just because SSL/TLS is being used doesn't mean the proper checks are being done. When you have a new application protocol using SSL/TLS, there are only three words for you: Test! Test! Test!

HP Jetdirect Certificate Guidelines

We've covered several client and server scenarios regarding HP Jetdirect and SSL/TLS. Here are some guidelines for issuing digital certificates to an HP Jetdirect:

- There is only one Identity certificate on HP Jetdirect, so supporting multiple certificates and things of that nature when Jetdirect is an SSL/TLS server are not an issue at the present time.
- Because there is only one certificate, be sure to issue a certificate that can do Client Authentication and Server Authentication unless you are absolutely sure that Jetdirect will only act as one or the other.
- Do not use the self-signed certificate that Jetdirect generates by default except for a temporary session in order to replace it with a better one.
- Each Jetdirect should have a unique certificate.
- Each Jetdirect should have host records in DNS and those DNS names should match the Jetdirect certificate.
- Use the Web Server (as described previously) to generate a Certificate Request so that the private key for Jetdirect always remains private. Be sure that the Common Name matches the FQDN of Jetdirect to avoid certificate errors.
- Do not store the same Identity Certificate on multiple Jetdirect cards (e.g., wildcard certificates for printing).
- When Jetdirect is acting as an SSL/TLS client and is being presented with a server certificate that has a wildcard character, follow the guidelines in RFC4513 for *all* protocols, not only LDAP. Otherwise, the wildcard certificate may not be accepted.
- When Jetdirect is acting as a client, be sure to have the CA certificate installed. The CA certificate has to be the top-level CA certificate.
- When Jetdirect is acting as a client, be sure to have the server send back the certificate chain. Jetdirect has minimal amount of storage available for certificates so it requires that functionality.

Following these guidelines will keep you looking younger.

Embedded Devices and Digital Certificates

One of the more common uses of digital certificates is for Virtual Private Network (VPN) access. In some environments, remote users are issued a USB key and the USB key is programmed with a digital certificate assigned to the user. The private key of the certificate is protected via a PIN number that provides encryption. When a user brings up the VPN, the PIN number must be used so that the private key can be accessed to establish the VPN connection. This behaves like two factor authentication: What do I know (the PIN #)? What do I have (the digital certificate on the USB key)?

With Embedded Devices, there is no password that can be entered. This means that the certificate private key is likely stored "in the clear". Even if a password was required, it is going to be used by the firmware of the Embedded Device rather than a user (an Embedded Device may not even have a

physical user interface) and is probably stored right next to the digital certificate. In short, an analysis of the non-volatile storage of your embedded device may reveal more information than you want. When deploying certificates to embedded devices, there are several questions that you should ask the vendor:

- Can the private keys be exported?
- Can exporting private keys be prevented?
- Is the private key password protected?
- How is the password protected in non-volatile storage?
- How is the digital certificate protected in non-volatile storage?
- Does the embedded device meet any security standards regarding the handling of security information like passwords or digital certificates?
- Can this information be securely erased when I no longer need the embedded device?

Anytime you are deploying digital certificates to embedded devices, you need to be sure that you know the answers to these questions. When an embedded device leaves your physical possession, due to a hardware failure, warranty failure, theft, or simply selling them as used, the non-volatile storage may be able to be accessed and some information about your PKI or network could become available. The worse case is that the digital certificate could be obtained from a device no longer used and that digital certificate may be used to attack your network.

Which HP Jetdirect Products Support SSL/TLS?

HP Jetdirect has supported SSL/TLS for a long time. It is actually easier to answer this question in the negative. Here is a list of devices that are popular but do **not** support SSL/TLS

- Any external parallel port print server does not support SSL/TLS. Common products are the 170X, 300X, 500X, and 510X. The firmware versions are X.08.XX or lower.
- Any MIO products. The firmware versions are X.08.XX or lower.
- The 600n series of EIO print servers – such as the J3113A 10/100. The firmware versions are X.08.XX or lower.
- Value based products like the 175X (external USB) or the 200m (LIO).
- The Embedded Jetdirect in value based products such as the CP4005n and HP Color LaserJet 3600n

Here are some popular HP Jetdirect products that do support SSL/TLS. This is not a comprehensive list and, as always, be sure to upgrade your Jetdirect to the latest firmware available for the best experience (http://www.hp.com/go/wja_firmware).

- EIO print servers 610n, 615n, 620n, 625n, 630n, 635n with the latest firmware.
- External USB print server en3700
- Embedded Jetdirect products with the Jetdirect product number J7949E, J7982E, J7991E, J7997E, J7974E, and J7992E. You can get the product number of Embedded Jetdirect devices through Web Jetadmin or via the HP Download Manager.

Summary

Well, you've made it to the end. Hopefully, you've learned a lot about SSL/TLS and how it works on HP Jetdirect. Most importantly, know that when someone says their communication is secure because it uses SSL/TLS, you'll know that there are many more questions to ask before any actual statement about security can be made. Hope you enjoyed it!