Using easy-rsa certificates for authentication within IPsec in standalone Windows systems

Author(s): David Perez

Jose Pico

Date: January 2014

Version: 1.0

Contents

1 Abstract

Abstract: In this white paper we describe how to use easy-rsa, the free and open source certification authority software based on OpenSSL, to generate digital certificates that can be used to mutually authenticate IPsec connections between standalone Windows systems. First we describe the problem to be solved, and then we provide two step by step guides to generate, install and use those digital certificates, using two different methods: generating certificate signing requests (CSR) in the hosts and signing those requests with easy-rsa, and generating the full certificates directly with easy-rsa, including their private keys and CSRs.

2 Introduction

2.1 Insecure communication protocols

Some communication protocols, like HTTP or TELNET, are vulnerable to man-in-the-middle attacks, because they exchange data in the clear, and because they do not provide mutual authentication of both ends of the communication. A subset of those insecure protocols have secure counterparts that we can use to replace them, like HTTPS instead of HTTP, or SSH instead of TELNET, but some of them do not have a secure alternative.

A clear example of the latter is the SMB (Server Message Block) protocol, which is used to access shared files and folders in Windows (and other) environments. Its variants SMBv1 and SMBv2 provide authentication of the user attempting to access resources in the shared folders, but they do not provide an authentication mechanism for the server to prove its identity to the client, and they do not offer the possibility of encrypting the traffic while in transit over the network. Version 3 of the protocol, SMBv3, introduces traffic encryption as a new feature[1] of the protocol, which is an improvement, although it still seems to fail to address the problem of the authentication of the server. However, since SMBv3 is only available in systems running Windows 8, Windows Server 2012, or later, the vast majority of SMB traffic in our networks today is still SMBv1 or SMB2 and therefore exchanged in the clear. That means that any attacker that can position himself between the client and the server can obtain a copy of all files transferred, as demonstrated by the SMB export plug-in that we implemented for Wireshark some time ago[2].

Nevertheless, SMB is just an example. In many, if not most, of the networks we have analyzed, both in the long distant past and in very recent times, we have found custom applications that use completely insecure communication protocols, with a total lack of mutual authentication and/or encryption.

When trying to protect the communications of these insecure protocols, IPsec may come in handy.

2.2 IPsec in Windows

IPsec (IP security) is a set of protocols that allow two IP entities to authenticate each other, negotiate cryptographic keys, and use those keys to authenticate and encrypt each IP packet they exchange,

[1] http://support.microsoft.com/kb/2709568/en
[2] http://www.layakk.com/en/lab.html#SMBPlugIn

thus protecting any communication carried out between those two endpoints at any higher level in the TCP/IP stack.

When compared to other security protocols that operate at higher levels in the TCP/IP stack, like TLS or SSH, IPsec offers the advantage of being transparent to any applications communicating through it: applications do not need to be aware of the existence of IPsec, whereas for an application to be able to use higher level security protocols, like TLS for example, the application must be designed or modified to support those protocols.

In the case of insecure protocols that cannot be replaced with a secure alternative, and which cannot be modified (be it because the source code is not available or because the cost would be prohibitive), IPsec may be a good option, if not the only option, to protect their communications.

IPsec has been available in Windows, included in the operating system (no need to install extra software), since Windows 2000. While the early versions could be considered difficult to configure and manage, that is hardly the case in recent Windows systems like Windows 7 or Windows Server 2008, where the configuration of IPsec rules is integrated in the Windows Firewall with Advanced Security console.

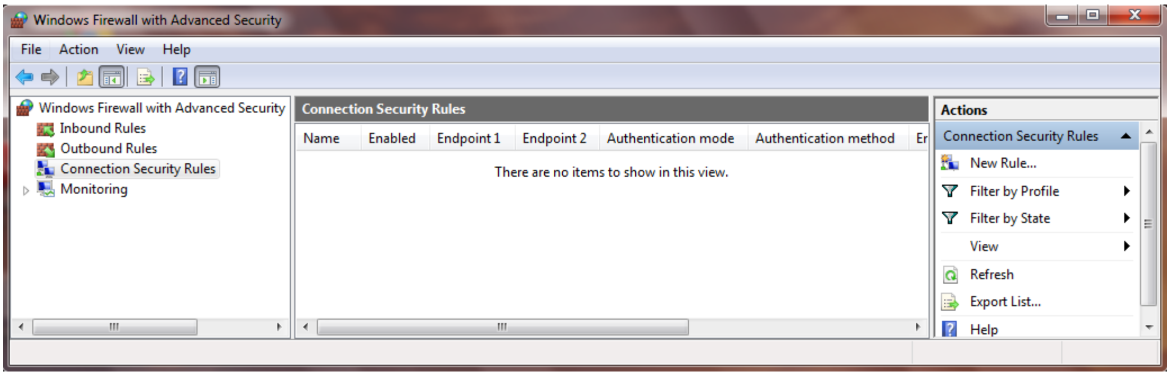


Figure 1 - Windows Firewall with Advanced Security

Windows' implementation of IPsec offers several authentication methods, including using digital certificates issued by trusted certification authorities (CA). Other methods of authentication include Kerberos (in a Windows domain environment) or NTLMv2, but these articles will concentrate on the use of digital certificates because it is the strongest authentication method available both to domain and standalone computers.

In a Windows domain environment, managing the generation and installation of the appropriate certificates in the appropriate systems is really easy. An Enterprise (online) Certificate Authority can be set up in any domain member server (the CA software is included in Windows Server), and all the appropriate computers can be configured to automatically request from it, and install, the appropriate certificates. Such configuration may (should) be carried out centrally, using Group Policy, and applied automatically to the appropriate organizational units. Furthermore, IPsec may (should) also be configured using Group Policy, making the management of the whole solution very convenient, even for hundreds of systems.

However, configuring two standalone Windows systems to communicate using IPsec is a little more tedious, because the certificates will have to be generated and installed manually, and IPsec will need to be configured manually on each system.

2.3 Easy-rsa: A free and open source CA based on OpenSSL.

Easy-rsa is a lightweight, free and open source software application that allows the user to set up and manage a certification authority (CA). It constitutes a subproject of OpenVPN (http://community.openvpn.net/openvpn/wiki/GettingEasyRsa)

Programmed in POSIX shell and based on OpenSSL, which it requires, easy-rsa is available for Linux and Windows among other platforms.

The installation of easy-rsa in the Linux platform is especially simple, because all the external utilities that it requires are usually already there or can easily be installed using the appropriate package manager.

Easy-rsa constitutes a great alternative to Microsoft Active Directory Certificate Services (MS ADCS, Microsoft's CA) if you don't need an online CA and you don't want to dedicate a Windows Server to this function. On top of that, using easy-rsa you will also get more flexibility, because its behavior can be tailored in much more detail.

2.4 Digital certificates

A digital certificate is a file that contains the public key and other relevant data (but not the private key) of some entity, all of which is signed by a CA. Anyone in possession of the public key of the CA will be able to validate its signature and therefore be assured that the information contained in the certificate was validated by the CA.

In order to generate a digital certificate, first a certificate signing request (CSR) must be generated. An entity needs to generate a private-public key pair and then include the public key in a so-called certificate signing request (CSR), along with whatever information it wants to have included in the certificate (e.g. common name, intended uses of its keys, etc.).

Then, the CSR must be brought to a certification authority (CA), which, after validating the information contained in the CSR, will use its own private key to sign the contents of the CSR, producing the corresponding digital certificate.

The certificate needs then to be given back to the requesting entity, so that such entity can use it, in tandem with the corresponding private key, which never left the requesting entity.

Finally, the requesting entity may or may not want to export the private key together with the certificate so that it can be backed up or transported to a different system.

2.5 Generation and distribution of certificates: different approaches

Given the general process of generating certificates described in the previous section, a system administrator who wants to use digital certificates for the authentication of IPsec communication among some systems that fall under his control, could use at least two different strategies to generate and install the certificates. One way to do it would be to generate each key pair and CSR on each system, have them signed by the CA and the resulting certificates installed back in their corresponding systems. An alternative method would be to generate a bunch of key pairs and their corresponding CSRs on a central system, have them signed by the CA, get the certificates back, and then export together each private key with its corresponding certificate, and install each whole set on each appropriate system.

The first method has the advantage of the private key never leaving the single system where it will be used, but has the disadvantage of the system administrator needing to visit each system twice, to generate the key pair an CSR first, and then to install the certificate signed by the CA. However, the fact that the private key never leaves the system may not be too relevant if all systems are managed by the same administrator.

The second method has the advantage of allowing the system administrator to generate all keys and certificates at once, having to visit each system only once, to install both the certificate and the private key at the same time.

Additionally, the administrator also has the option to deal with X.509 extensions in different ways. Extensions in a certificate are fields that, among other things, may declare the uses for which such certificate should be trusted. Examples are "Server authentication", "IP security end system" or "Code Signing". CSRs may contain a list of desired extensions, which the CA may then decide to include or not in the final certificates, and the CA may or may not include additional extensions (not specified in the CSRs) to the final certificates. Thus, for a given set of required extensions, the administrator may include those extensions in the CSRs and configure the CA to include those extensions in the certificates, or generate the CSRs without extensions, and configure the CA to add those extensions when generating the certificates, or any combination of the above.

The following sections present two step by step guides that illustrate two different ways to set up an IPsec connection between two standalone Windows systems, using digital certificates for authentication, issued using easy-rsa. In the first guide, each key pair and certificate signing request (CSR) will be generated on its corresponding Windows system, including the desired extensions in the CSR, and easy-rsa will be configured to sign those CSRs including the requested extensions to generate the certificates. In the second guide, however, all CSRs will be generated in the Linux host and they will not contain information about any desired extensions; then, the CSRs will be signed using easy-rsa, configured to add the appropriate extensions to the corresponding certificates, and then the certificates, including their corresponding private keys, will be distributed to the Windows systems.

By no means are the two methods presented in these two step by step guides the only possible methods that could be followed. Indeed, a combination of both techniques, and possibly many other variations, could also be successfully applied. These step by step guides should be regarded just as illustrative examples.

3 Step-by-step guide #1: Generate each CSR on its corresponding host, and include desired extensions in them

This section constitutes a step by step guide to generate, install and use digital certificates to communicate via IPsec two standalone Windows 7 systems, using easy-rsa to create the certificates. The method described here includes generating a certificate request on each host, including the desired extensions in the CSRs, and then using easy-rsa to sign those requests, including the extensions specified in them, thus generating the corresponding certificates.

3.1 Step 0: Starting point and goal

In this step-by-step guide we will assume that we start with the following elements:

- Two standalone Windows 7 Enterprise systems, each with at least one network interface, with IPv4 properly configured, and network connectivity between them. We will also need administrator level access to them. In the example, the hostnames and IP addresses of these systems will be assumed to be those listed in the following table:

Hostname	IP address
PCALICE	192.168.108.10
PCBOB	192.168.108.20

- A Linux system, which may or may not have a network interface. During the installation of easy-rsa it would be convenient to have the system connected to the Internet, so that it can download the software, but the software installation could also be done via alternative methods. After that, all that is required is a means to transfer files to and from the system, which could be through the network, through the use of a USB memory stick, or via any other means. In the example, an Ubuntu Desktop 12.10 64 bit will be used, although any other distribution should work similarly, the hostname of the system will be assumed to be "linuxhost", and the system will have Internet connectivity.
- A network sniffer capturing the traffic exchanged by the two Windows systems. This is not strictly necessary, but it will be useful in order to observe the difference between the cleartext and the encrypted network traffic. In the example, an instance of Wireshark running in a third Windows system and capturing the traffic exchanged by PCALICE and PCBOB will be used.

The goal pursued in the example will be to configure the two Windows systems so that any IP traffic that they exchange gets protected by IPsec, using digital certificates for mutual authentication of the nodes, generated using an easy-rsa CA configured in the Linux system.

3.2 Step 1: Configure the Windows firewall of PCALICE to allow incoming traffic from PCBOB

On PCALICE, launch the Windows Firewall with Advanced Security console snap-in, by clicking the following options:

Start -> Control Panel -> System and Security -> Windows Firewall -> Advanced Settings

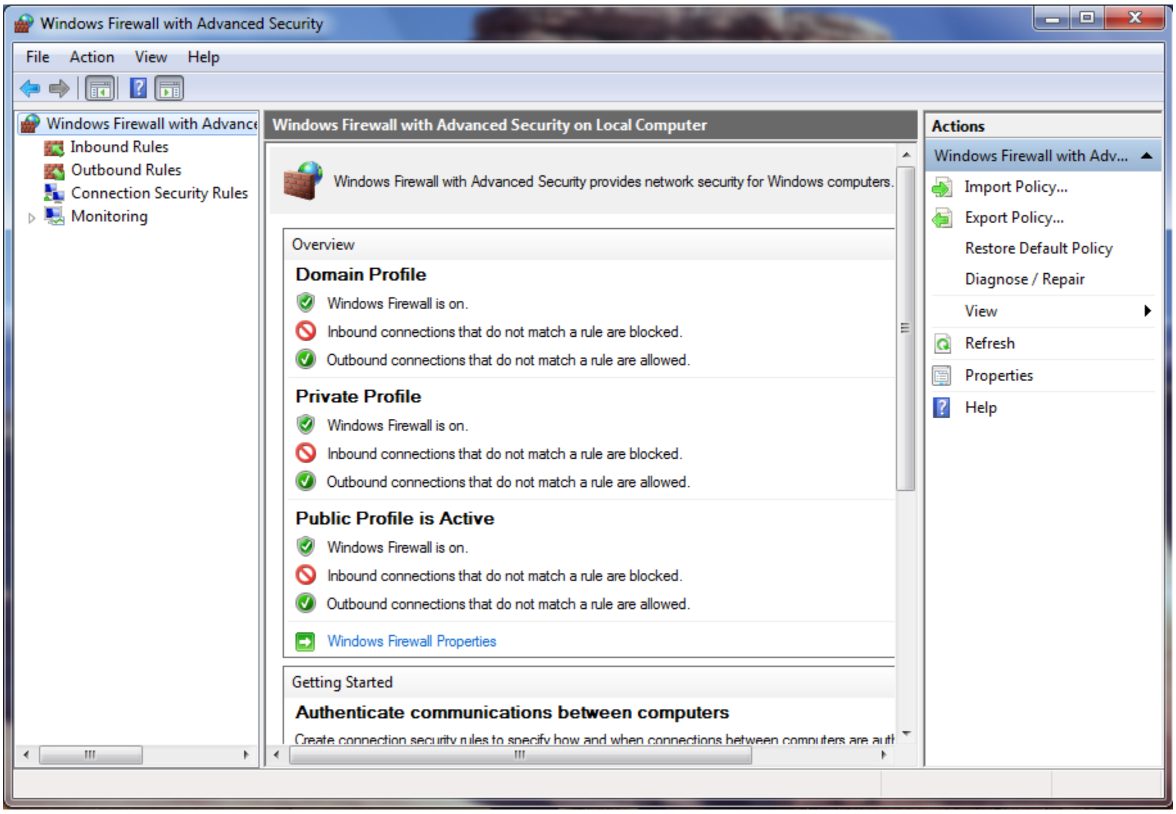


Figure 2 - Windows Firewall with Advanced Security on PCALICE

Select the container "Inbound Rules", right click on it, and select "New Rule..." to launch the "New Inbound Rule Wizard":

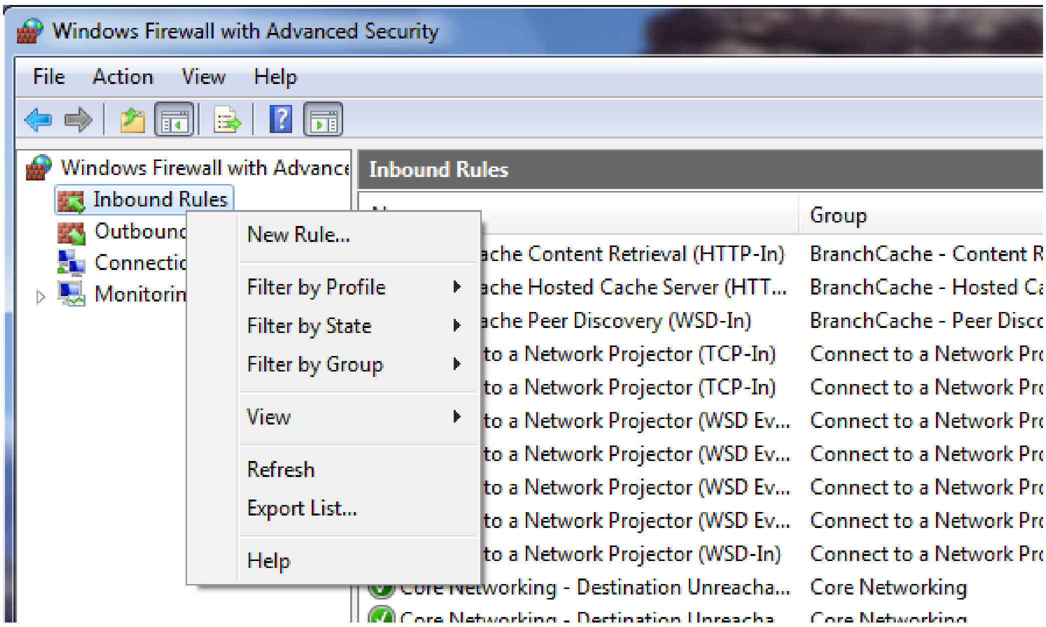


Figure 3 - Launching the New Inbound Rule Wizard

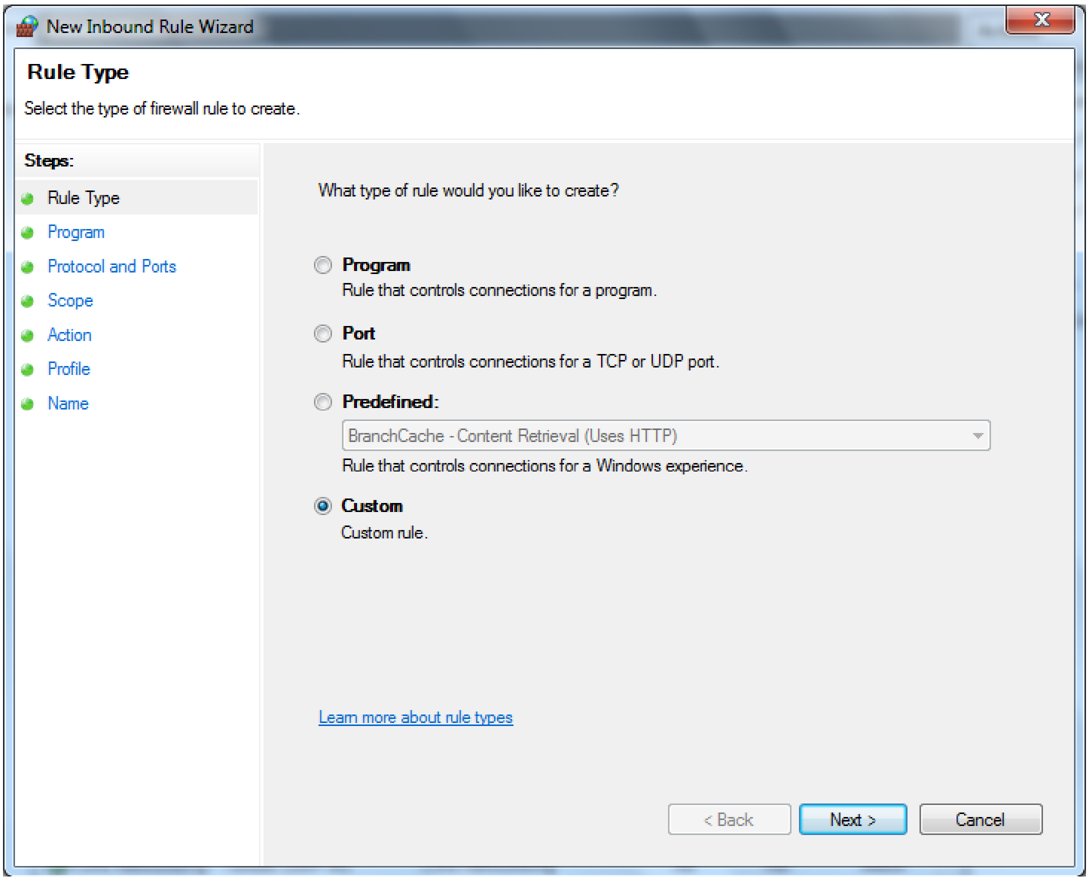


Figure 4 - New Inbound Rule Wizard

In the "Rule Type" tab, select "Custom" and click Next.

In the Program tab, select "All programs" and click Next.

In the "Protocol and Ports" tab, select "Protocol type: Any" and click Next.

In the Scope tab, in the section "Which local IP addresses does this rule apply to?" select "Any IP address", and in the section "Which remote IP addresses does this rule apply to?" select "These IP addresses", click "Add..." and type in the IP address of PCBOB (192.168.108.20) in the section "This IP address or subnet" of the dialog box that will pop up, and click OK to close that dialog box. The Scope tab of the "New Inbound Rule Wizard" will show the IP of PCBOB in the remote IP addresses section, as shown in the following figure:

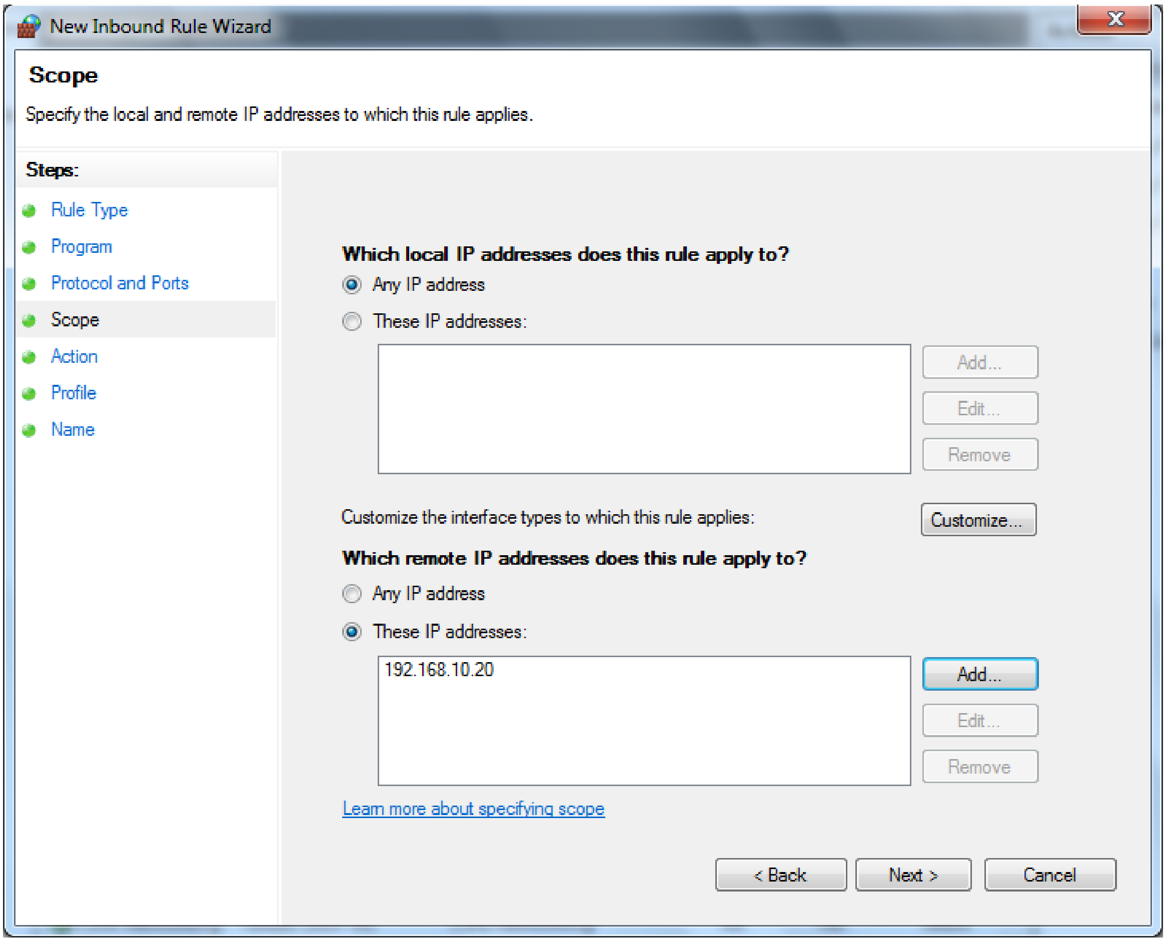


Figure 5 - Scope tab on PCALICE, showing the IP of PCBOB in the remote IP addresses section

In the Action tab, for now, select the option "Allow the connection" and click Next.

Note: Later on, you may want to change the action to "Allow the connection if it is secure", but for now we will allow the connection unsecured, so that we can verify the network connectivity independently of IPsec. Nevertheless, note that such change will not be absolutely necessary: it would only make a difference while there is no IPsec rule enabled that applies to this traffic, because if there is an IPsec rule that is enabled and that applies to this traffic, the traffic will be protected using IPsec.

In the Profile tab, select all three profiles (Domain, Private, Public) and click Next.

In the Name tab, give the rule a name, for example "Communication with PCBOB", and click Finish.

The new rule will appear in the Inbound Rules container of the Windows Firewall with Advanced Security console. Verify that in the Enabled column the rule displays "Yes":

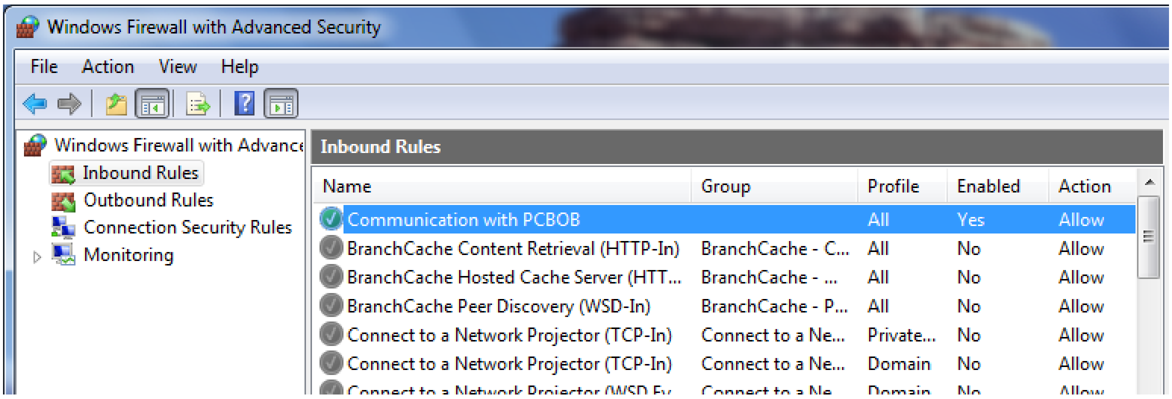


Figure 6 - The new rule, "Communication with PCBOB", configured and enabled on PCALICE

At this point PCALICE is ready to accept any incoming IP traffic coming from PCBOB.

Note: In this example, for simplicity, all traffic between PCBOB and PCALICE will be allowed and protected by IPsec. Obviously, these firewall rules could be tailored to allow and/or protect only specific traffic.

3.3 Step 2: Configure the Windows firewall of PCBOB to allow incoming traffic from PCALICE

Mirror, on PCBOB, the configuration tasks detailed in the previous step. Obviously, in this case the remote IP address to configure will be PCALICE's (192.168.108.10) and the name of the rule should be "Communication with PCALICE".

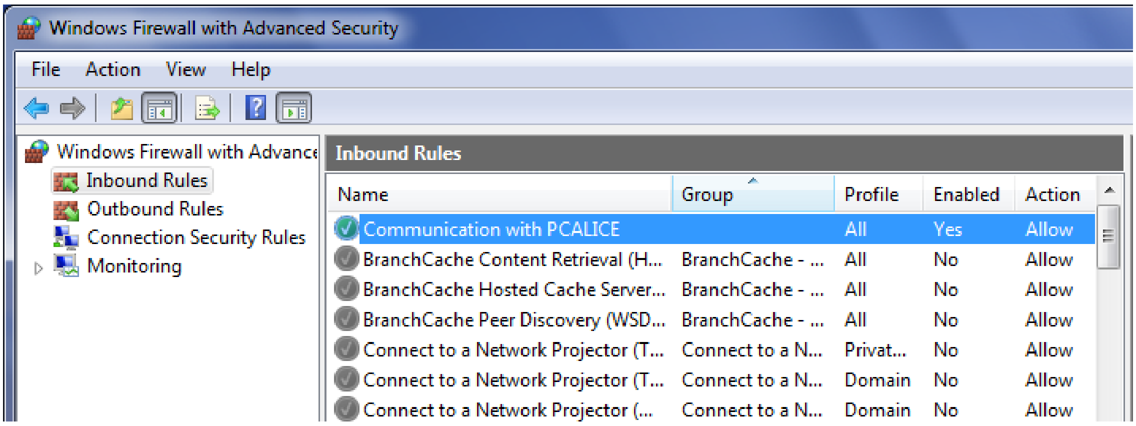


Figure 7 - The new rule, "Communication with PCBOB", configured and enabled on PCALICE

At this point PCALICE is also ready to accept any incoming IP traffic coming from PCBOB.

3.4 Step 3: Test the network connectivity between PCALICE and PCBOB

If the network connectivity between PCALICE and PCBOB is correct, and the new firewall rules are in place allowing the traffic exchanged by both systems, a PING should work correctly in both directions. Verify this is the case, confirming that responses are received when executing "ping 192.168.108.20" on PCALICE, and "ping 192.168.108.10" en PCBOB, in cmd consoles (Start -> type cmd.exe -> ENTER)

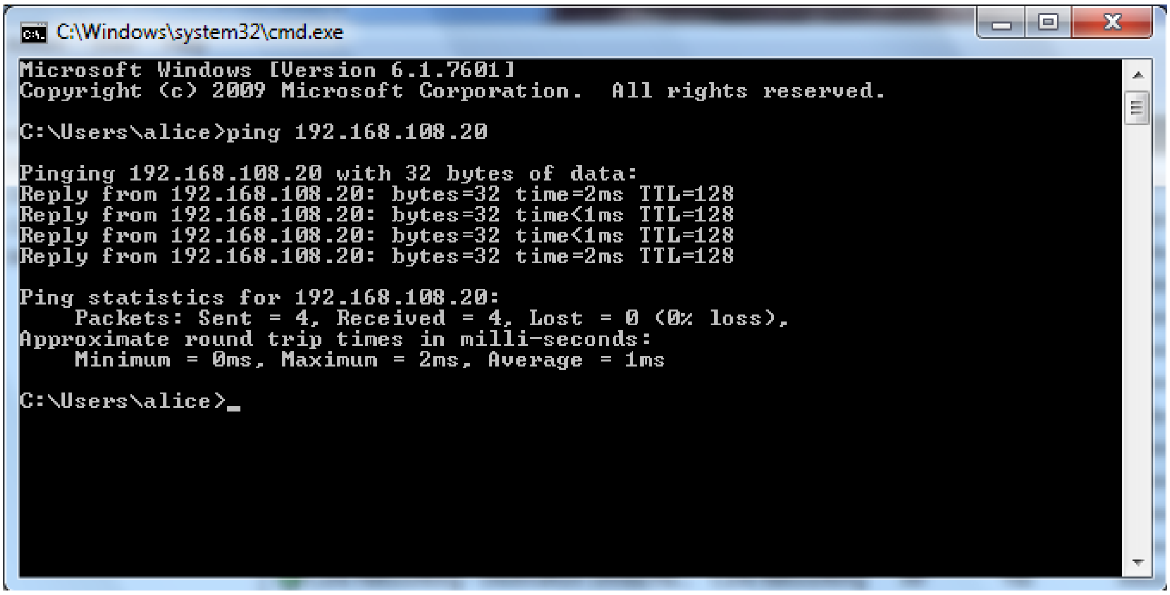


Figure 8 - Successful ping from PCALICE to PCBOB

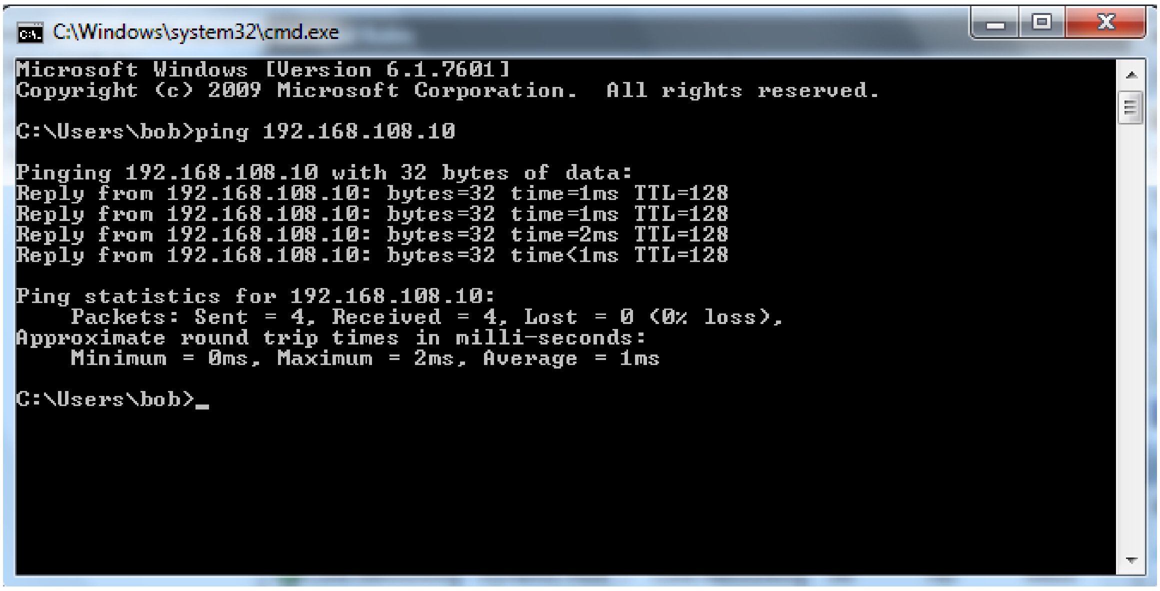


Figure 9 - Successful ping from PCBOB to PCALICE

If you capture that traffic with Wireshark you will see the ICMP packets flying in cleartext:

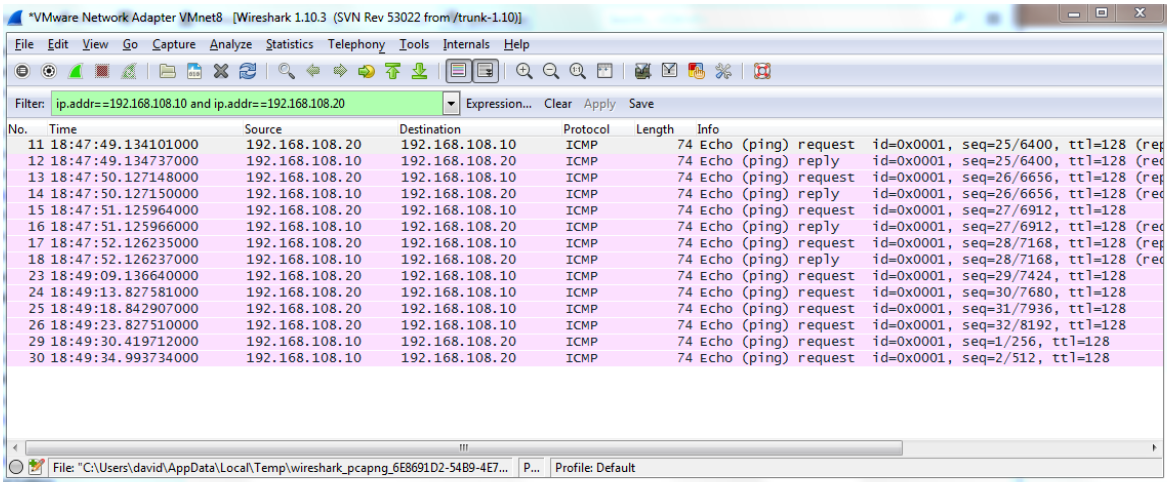


Figure 10 - Capture of PINGs between PCALICE and PCBOB, still in cleartext.

Note: If you don't want to allow cleartext traffic while you perform the rest of the configuration steps, you may disable the firewall rule in both systems now. Just make sure in that case that you do not forget to re-enable it later on when we modify it to require IPsec protection.

3.5 Step 4: Generate a key pair and a certificate signing request on PCALICE

Next, you will be generating a private-public key pair and a certificate signing request on PCALICE. Since the keys will need to be managed by the computer itself, you will need to open a Certificates console under the identity of the computer. In order to do so, perform the following actions.

First, launch an instance of the Microsoft Management Console (mmc.exe) with administrative privileges:

Start -> Type "mmc" -> Right click on the mmc program -> Run as administrator

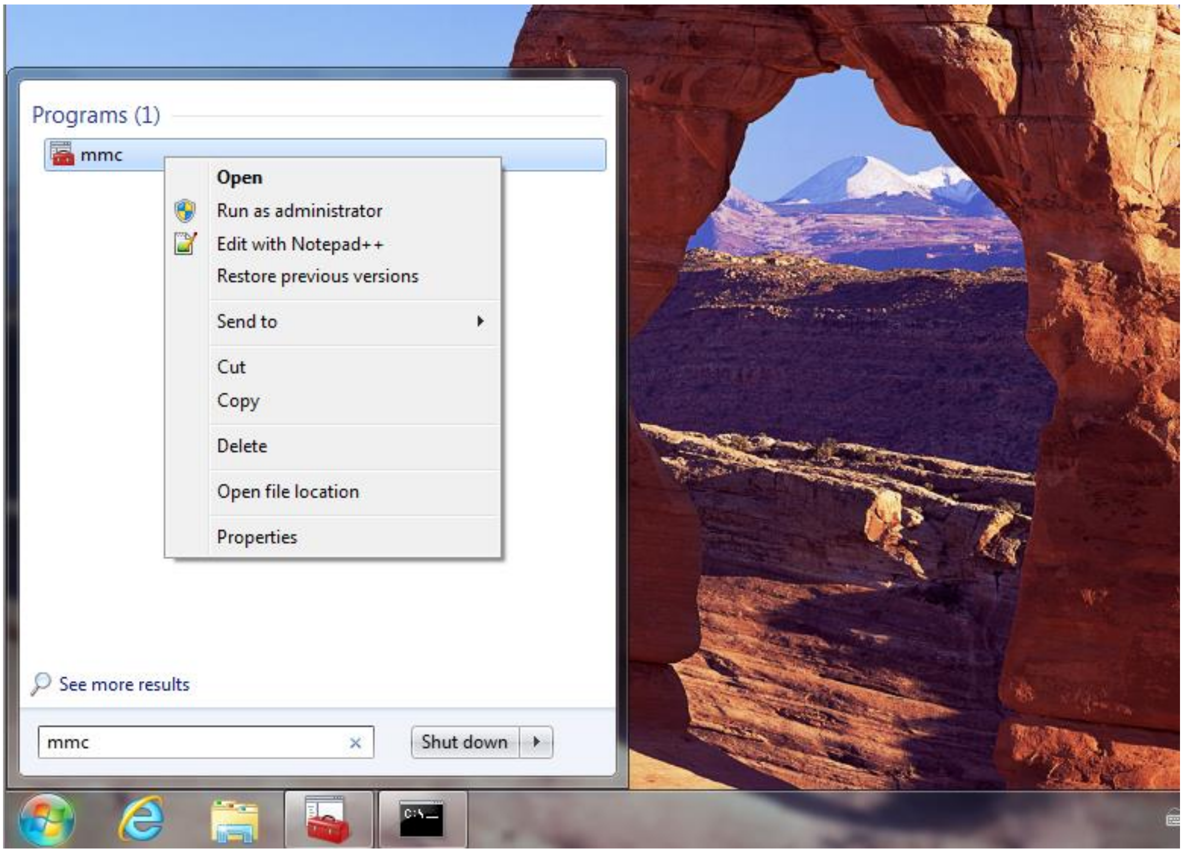


Figure 11 - Launching Microsoft Management Console (mmc) with administrative privileges

The following window should appear:

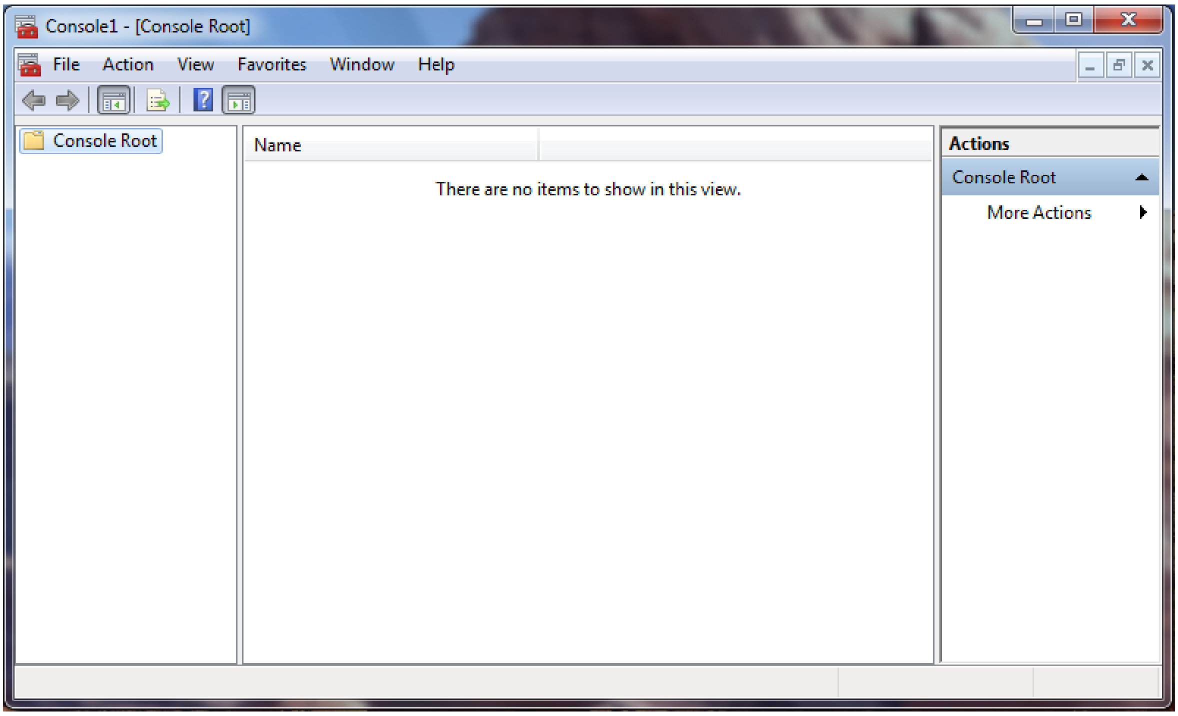


Figure 12 - Microsoft Management Console (mmc)

Note: At this point the window will not show any sign of being running with administrative privileges, but the following steps will not work as expected if you fail to start mmc.exe with administrative privileges.

Next, add the Certificates snap-in under the identity of the local computer, as follows:

File -> Add/Remove Snap-in... -> Select "Certificates" and click "Add >" -> Select "Computer account" in the pop-up window that will appear and then click "Next" in that window -> Select "Local computer: (the computer this console is running on)" and click "Finish" (still in the pop-up window, to return to the "Add or Remove Snap-ins" window).

At this point your Add or Remove Snap-ins window should show "Certificates (Local Computer)" in the "Selected snap-ins:" section:

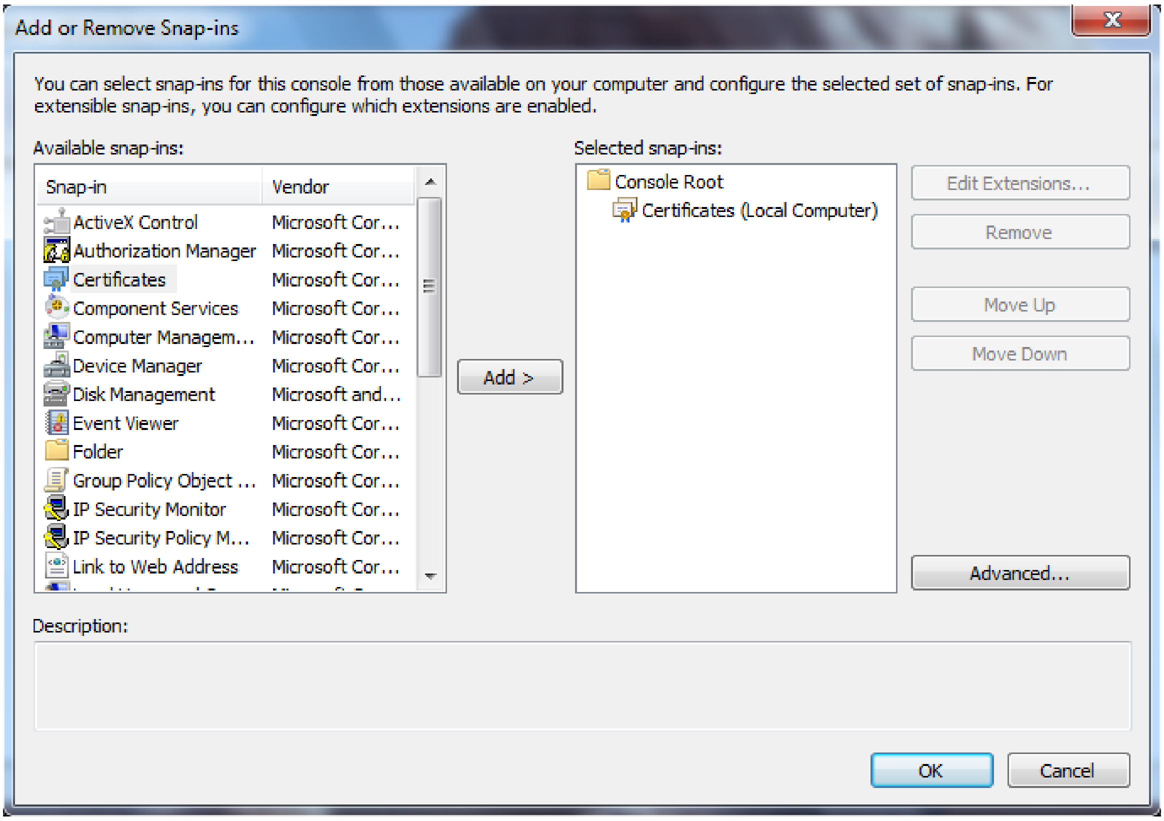


Figure 13 - Certificates (Local Computer) snap-in selected

Click OK to close the Add or Remove Snap-ins window and return to the MMC console, which should show the selected snap-in inside:

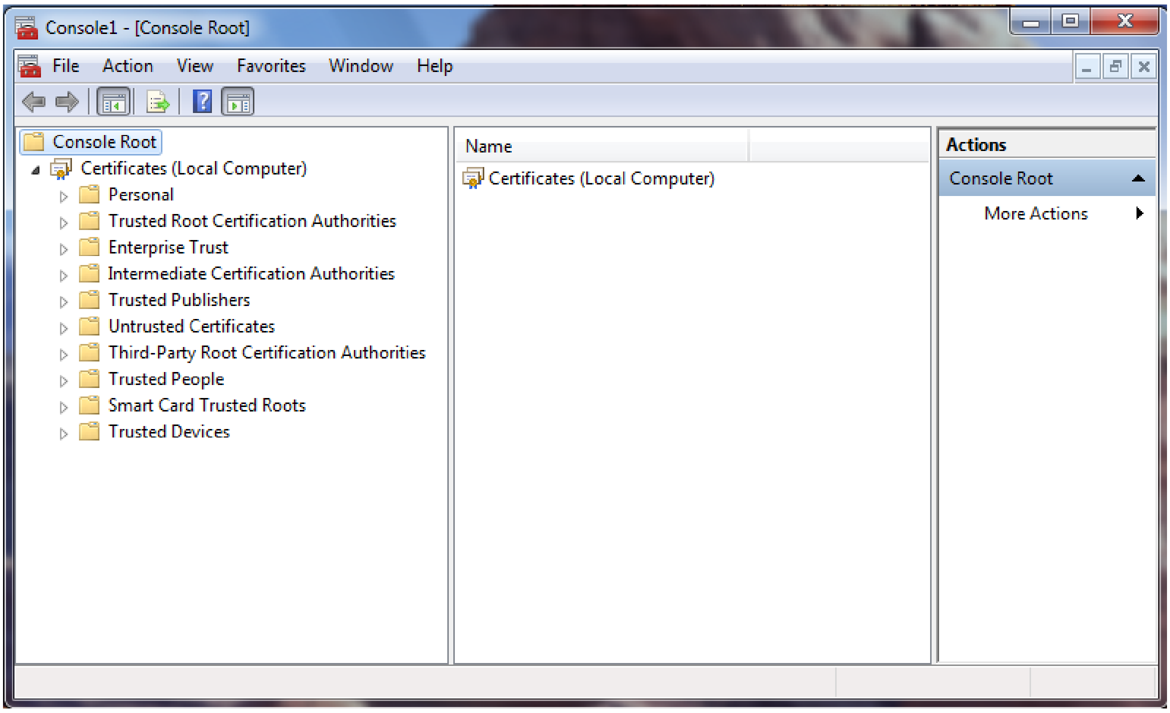


Figure 14 - Certificates (Local Computer) snap-in open in MMC

Note: The open snap-in must display "Certificates (Local Computer)" at this point. If it shows "Certificates - Current User" or anything else, you will need to stop, close it, and go back to repeat the above steps, but making sure this time that you perform them exactly as indicated.

Then, select the Personal container, and launch the appropriate Certificate Enrollment wizard by invoking:

Personal -> Right click -> All Tasks -> Advanced Operations -> Create Custom Request...

Note: Make sure you select the indicated wizard, "Create Custom Request...", and NOT "Request New Certificate"

The following window should appear:

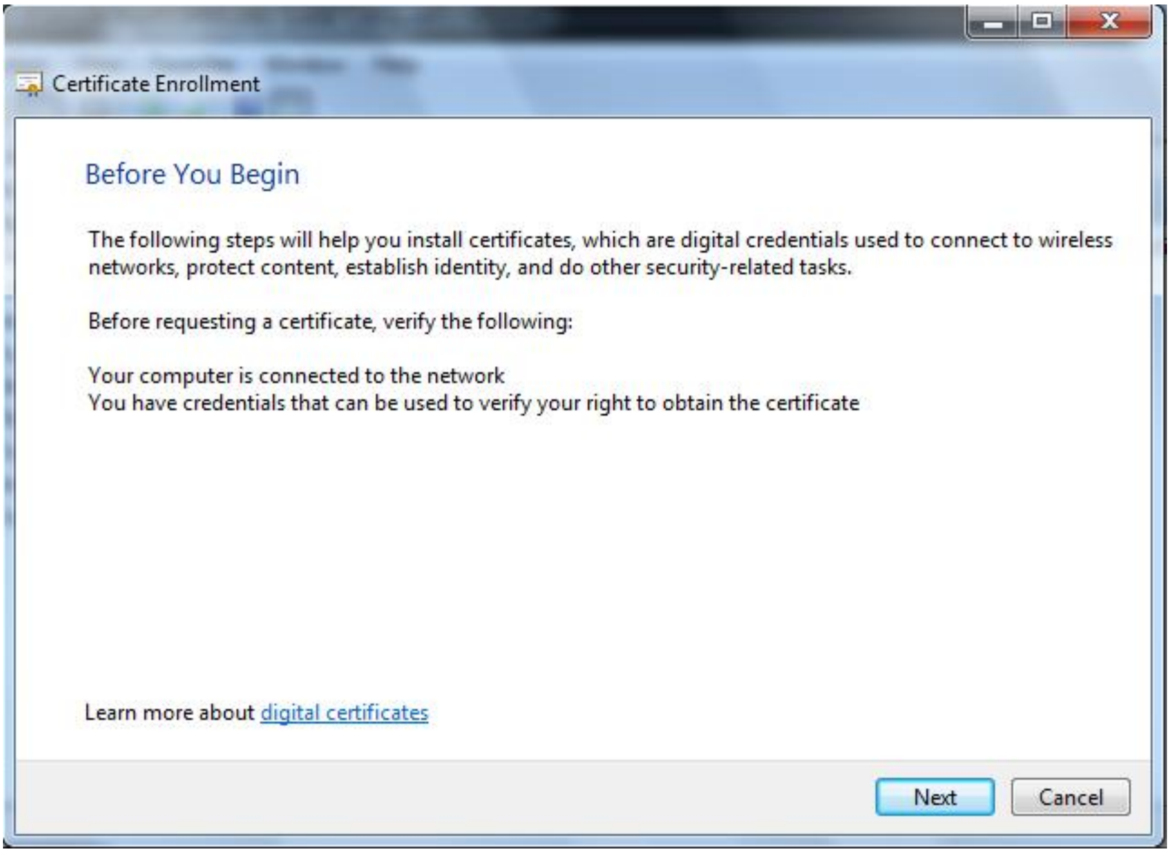


Figure 15 - Certificate Enrollment wizard

Click Next and the window will change to:

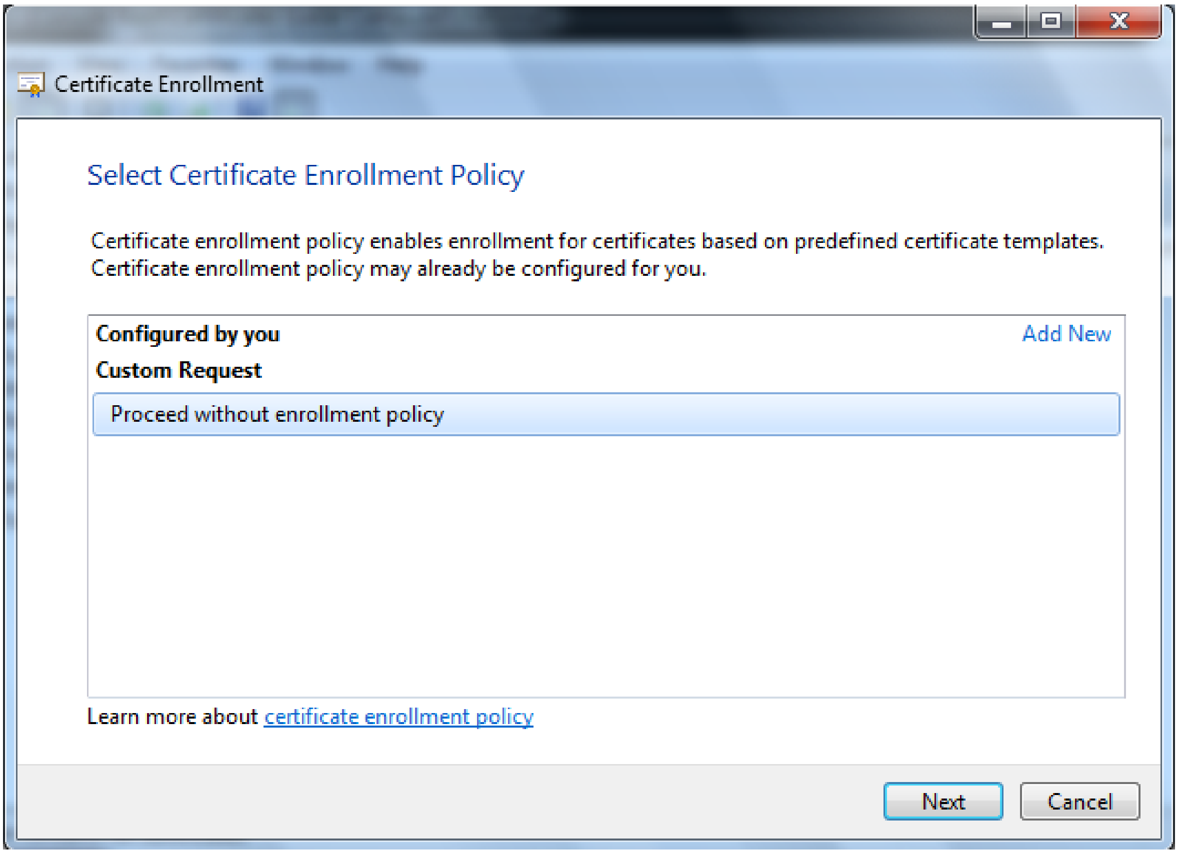


Figure 16 - Proceed without enrollment policy option

Note: If the option "Proceed without enrollment policy", shown in the previous figure, does not appear, then you probably launched the wrong wizard. Go back and make sure you follow the instructions precisely.

Click Next. The following window will appear:

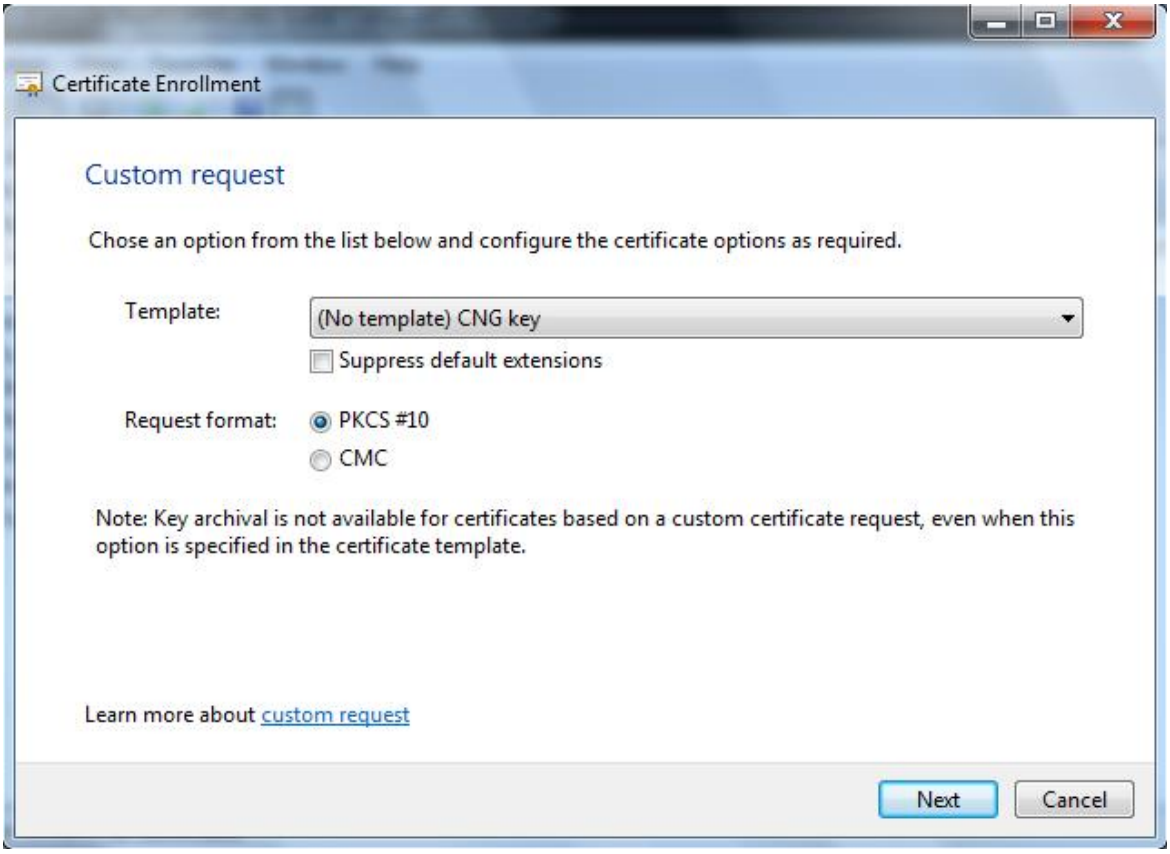


Figure 17 - Custom request default options

Leave the default options selected, as shown in the previous figure, and click Next. The following window will appear:

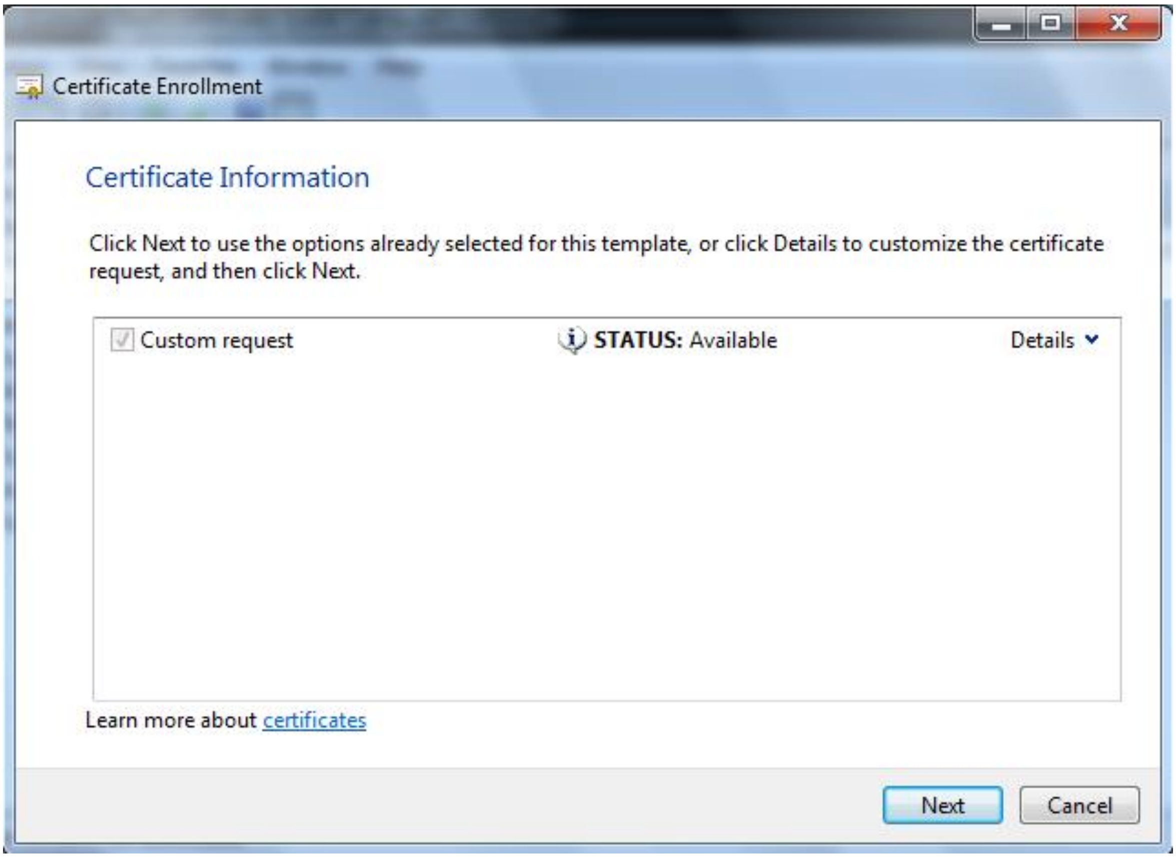


Figure 18 - Custom request, before customization

DO NOT click Next yet. Instead, click the little down arrow next to the text "Details", in the top right corner of the inner rectangle. The displayed information about the custom request will be expanded to:

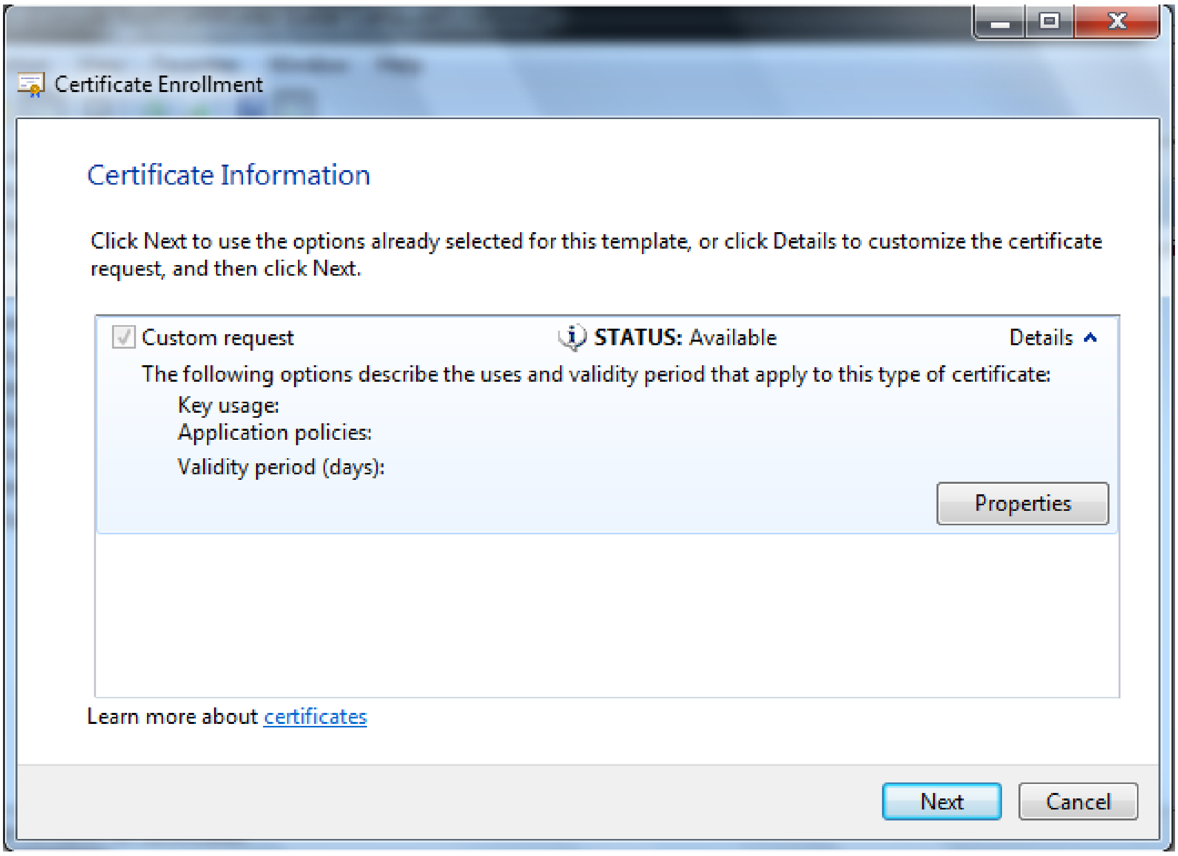


Figure 19 - Custom request details

At this point, as shown in the above figure, important details of the request, including Key usage, Application policies or Validity period are still empty. In order to configure these and other parameters, click the Properties button. The following window, with the title "Certificate Properties" will appear:

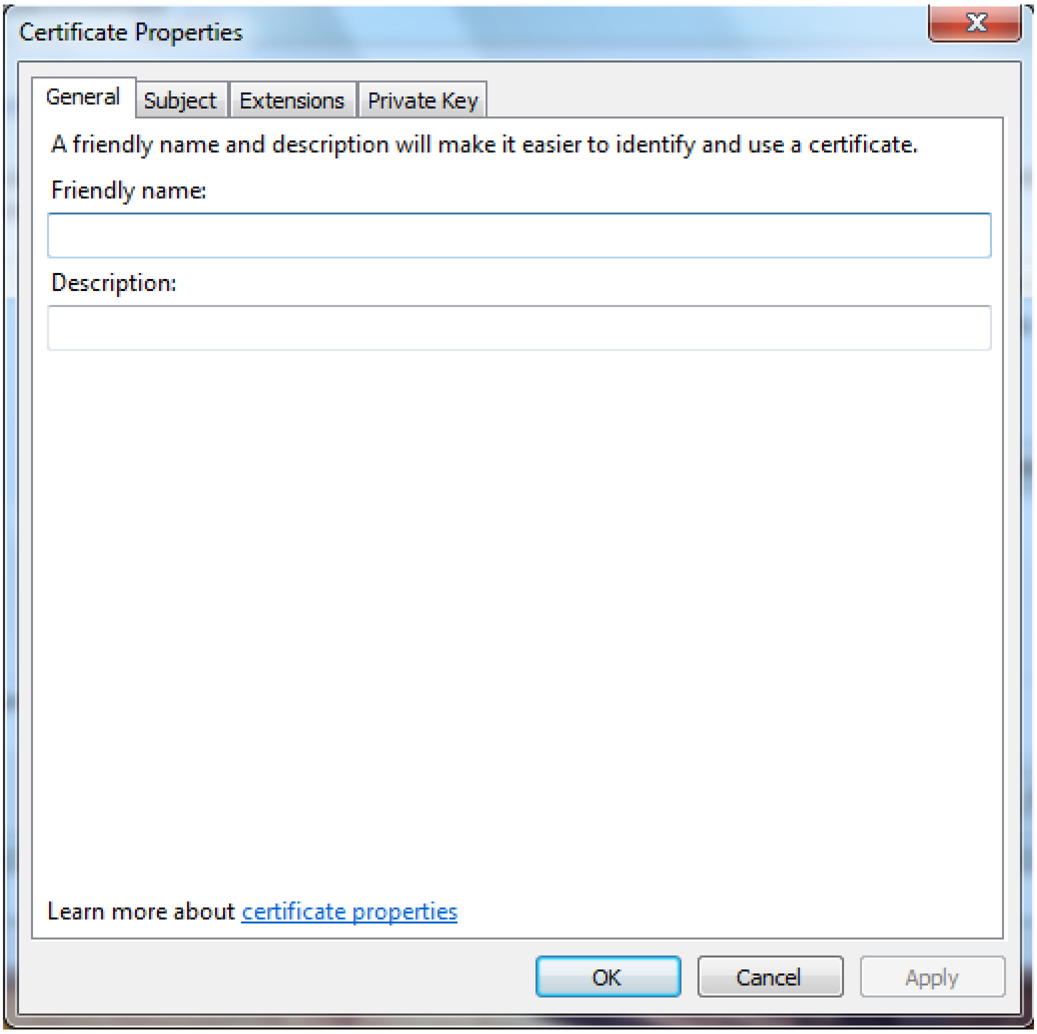


Figure 20 - Certificate Properties window

You may configure many properties of the request, but in this example we will configure just a minimal set that will be needed for the correct operation of IPsec.

Open the Subject tab by clicking on its name, and on the "Subject name" box select type "Common name", fill in the Value field with the text "PCALICE", and click the corresponding "Add > " button so that it gets displayed on the right column top box, as shown in the following figure:

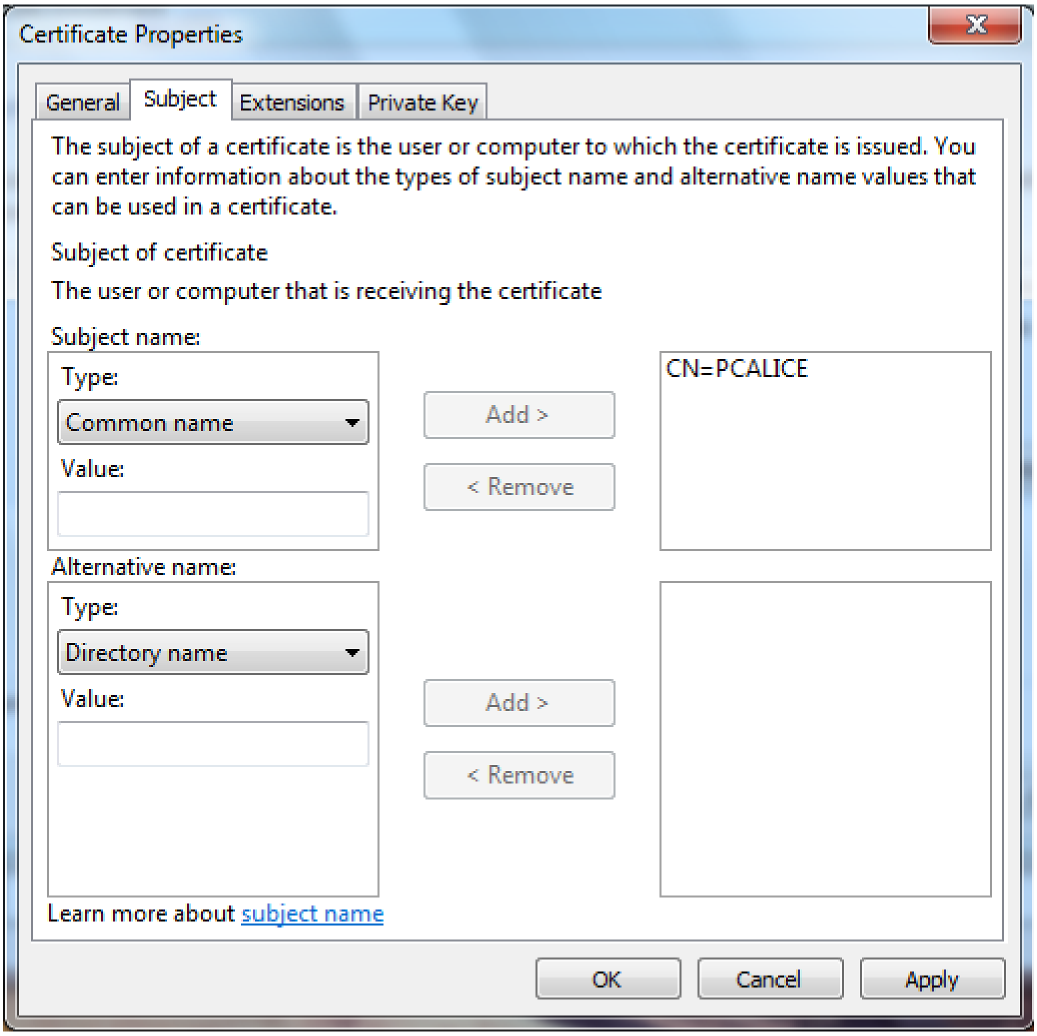


Figure 21 - Common name property

Note: You may choose any common name you wish, as long as you do specify one and it is unique for each host. We recommend you use the host name for your own sanity, but you may use any text you want: IPsec will not care about the particular names you choose.

Click "Apply" and then open the Extensions tab by clicking on its name. The Extensions tab will initially look like this:

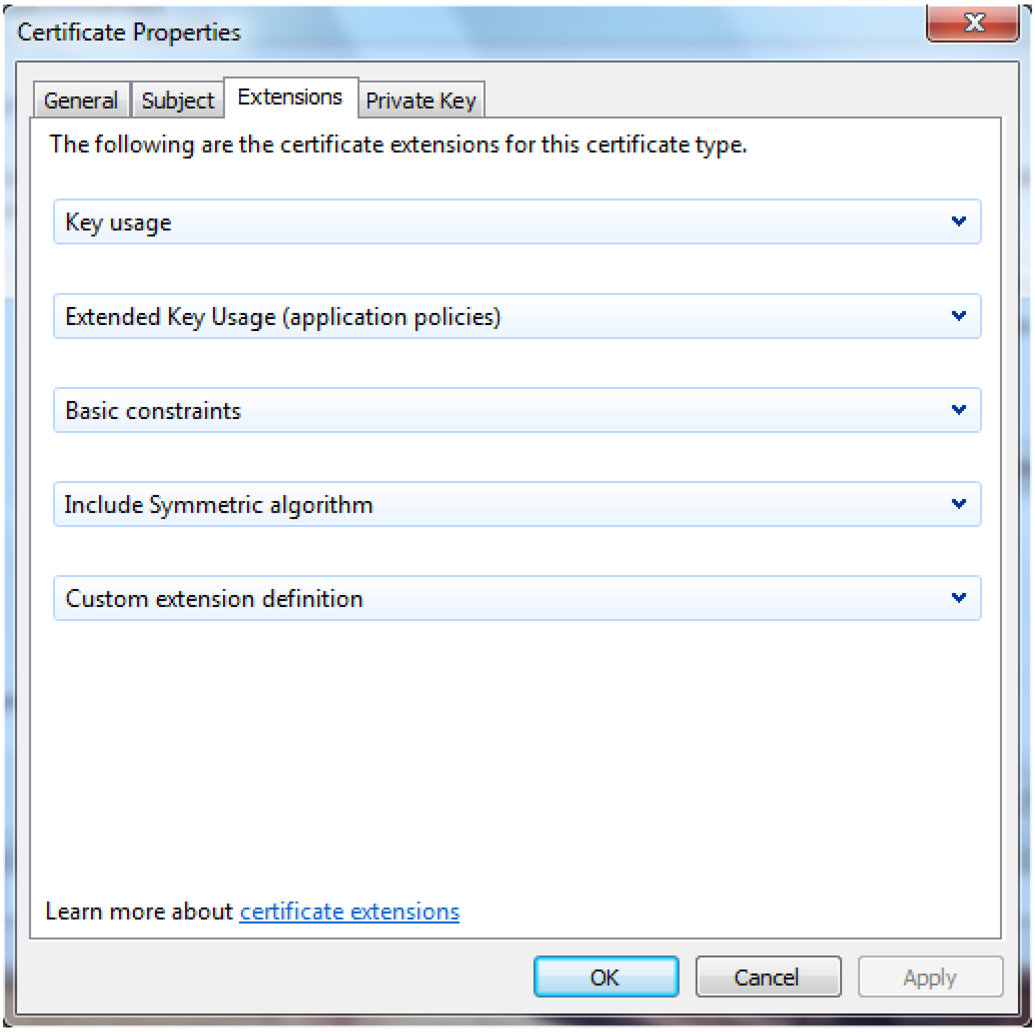


Figure 22 - Extensions tab

Expand the "Extended Key Usage (application policies)" section by clicking in the little down arrow located at the right end of its section name. Initially, this section will look like this:

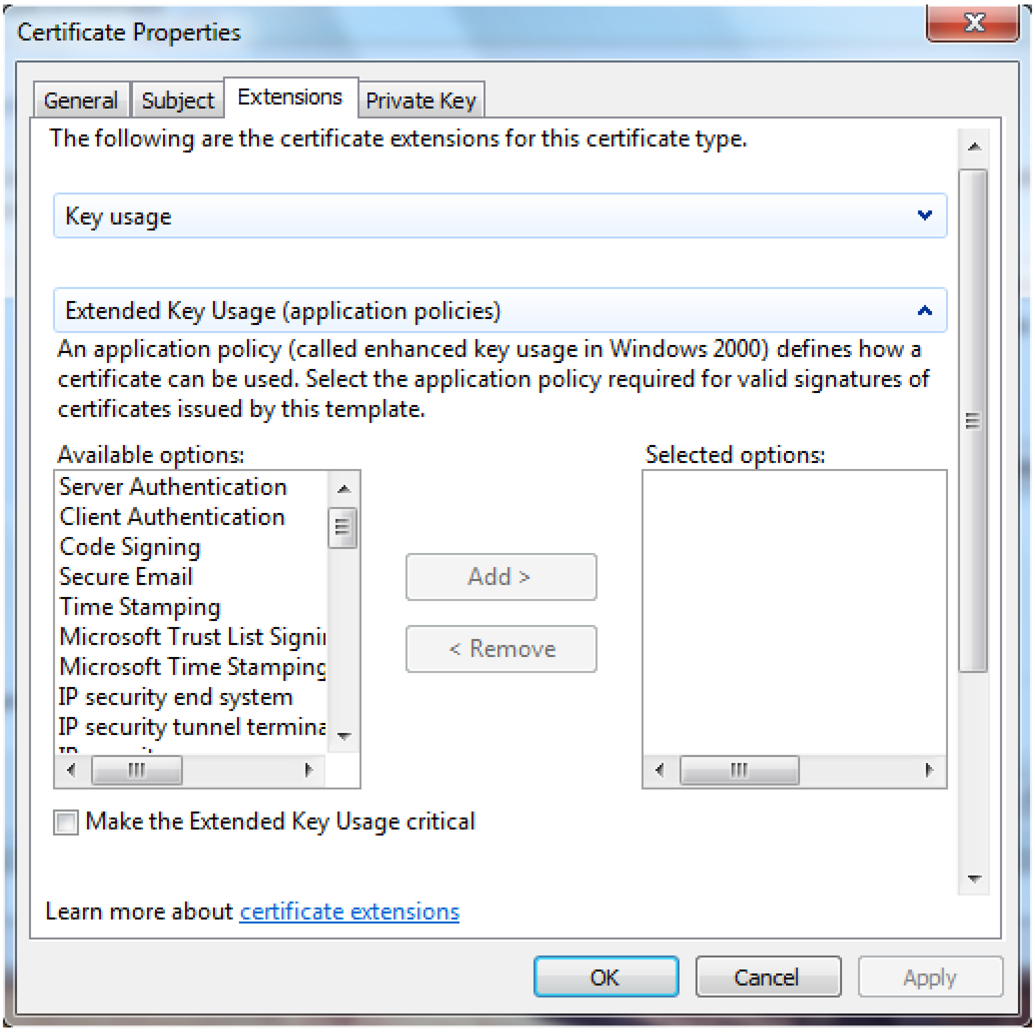


Figure 23 - Extended Key Usage (application policies) section, before configuration

In the "Available options:" box, select "Server Authentication" and click "Add > " so that it gets moved to the "Selected options box" on the right, and do the same with "Client Authentication" and "IP security end system". The end result must look like this:

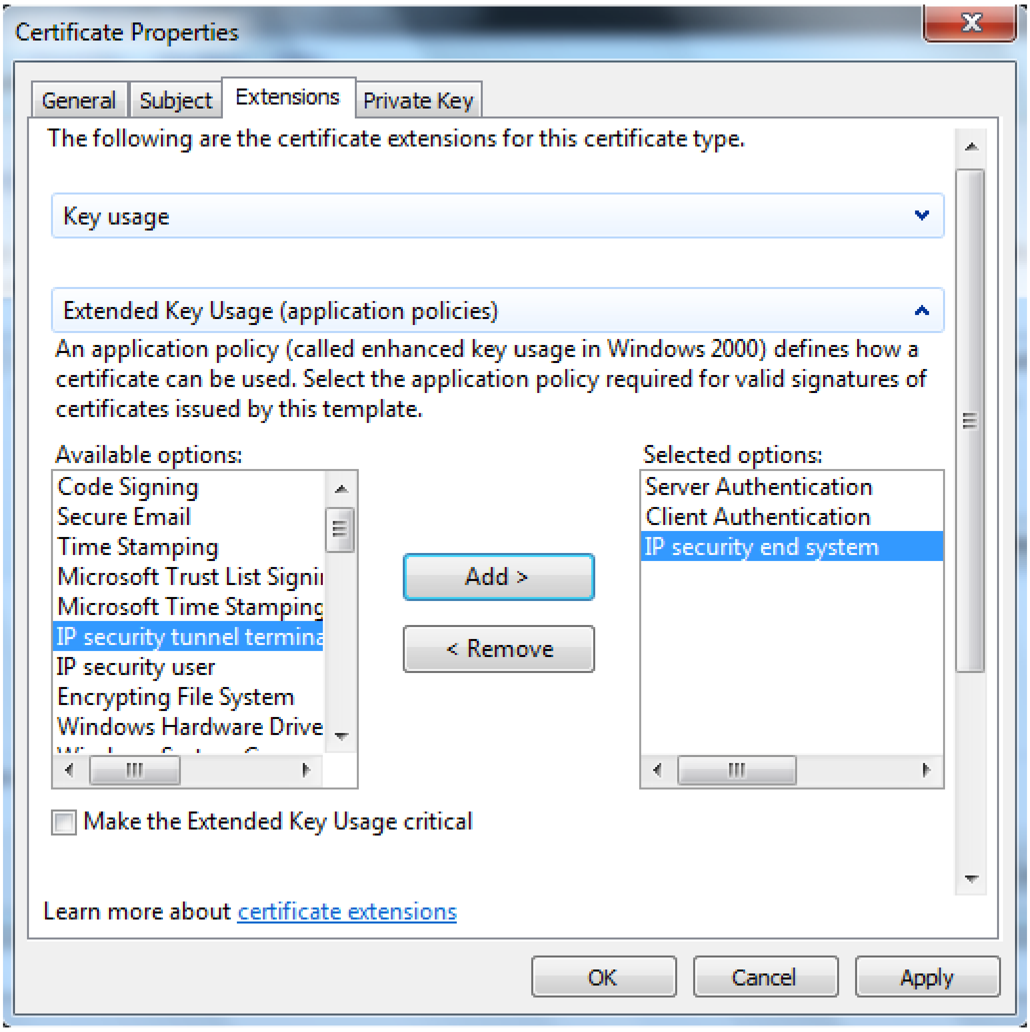


Figure 24 - Extended Key Usage (application policies) section, after configuration

Click "Apply" and then open the "Private Key" tab by clicking on its name. The "Private Key" tab will initially look like this:

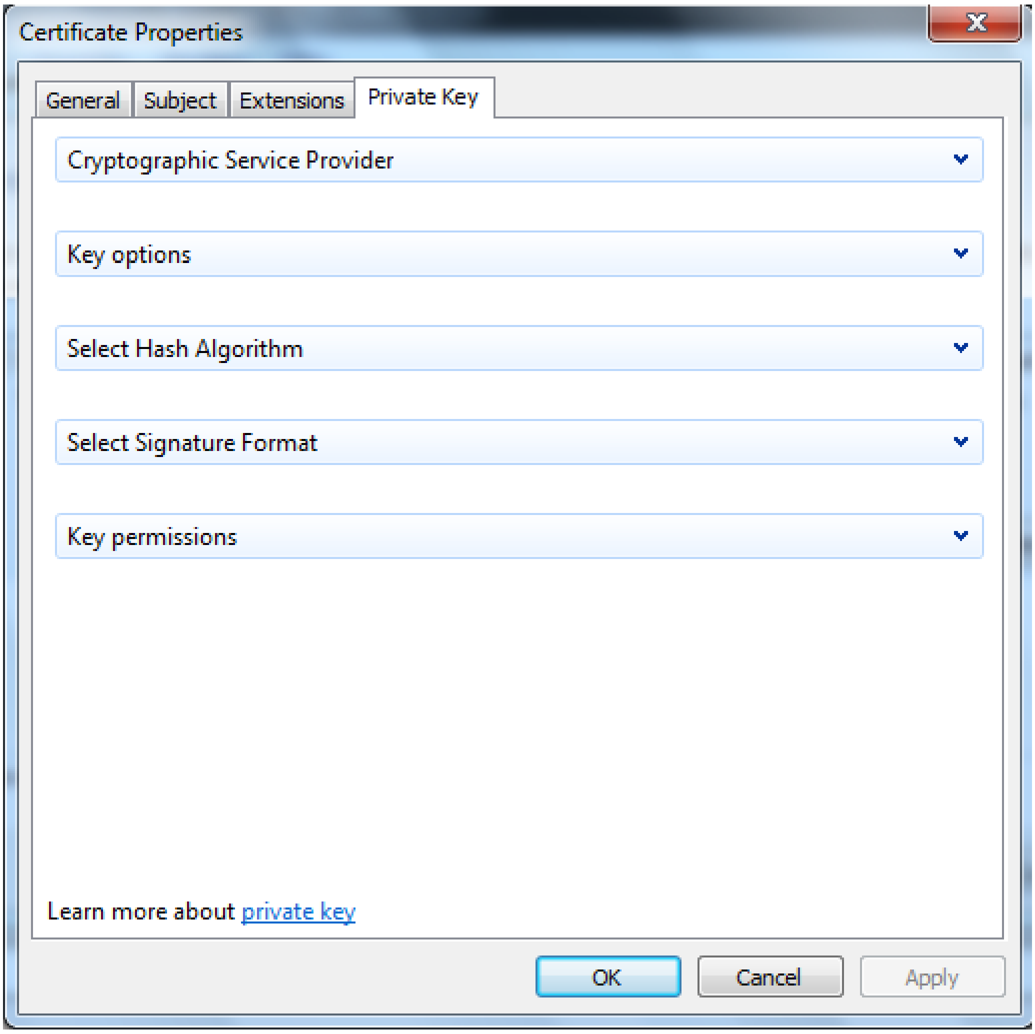


Figure 25 - Private Key tab

Here you may choose any properties you wish for the private key. In this example we will configure the use of a 2048 bit RSA key with a hashing algorithm SHA256. To do so, expand the "Cryptographic Service Provider" section and verify that the option "RSA, Microsoft Software Key Storage Provider", and only that option, is selected:

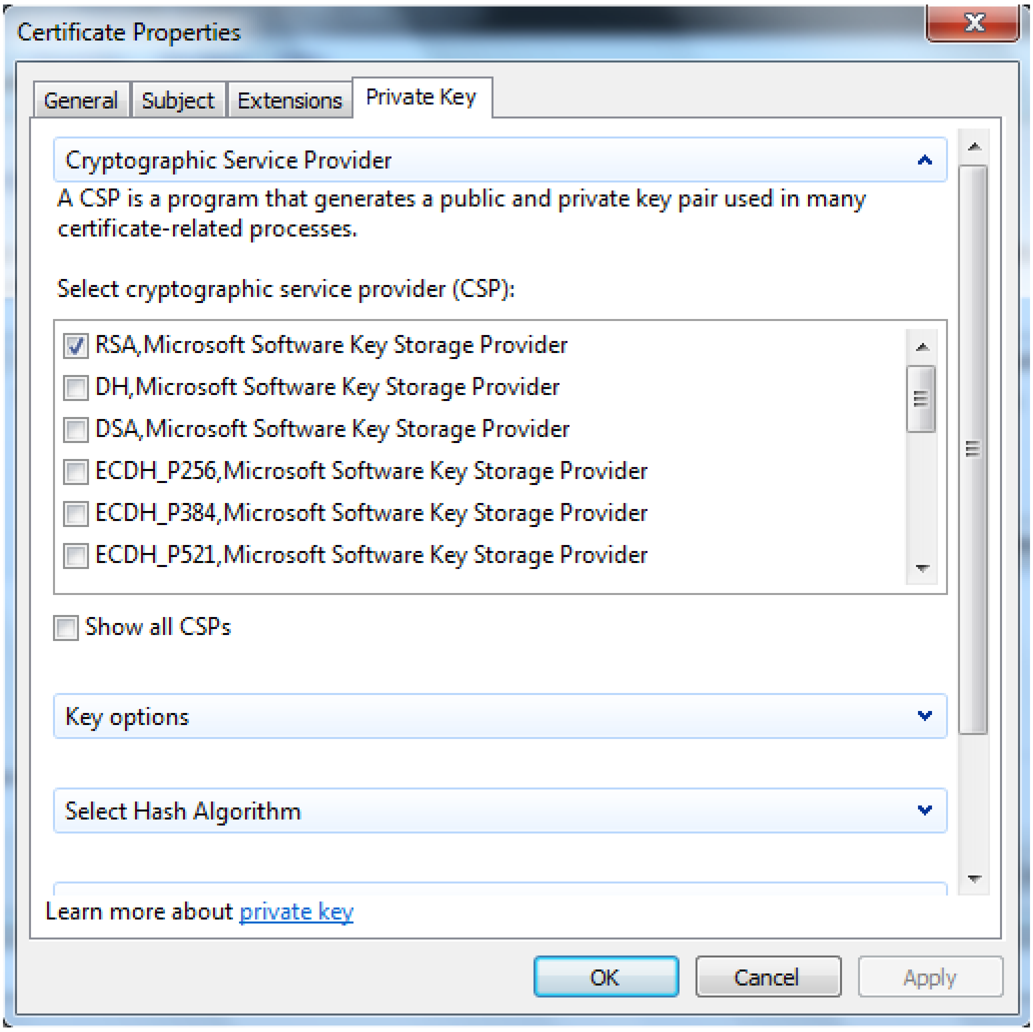


Figure 26 - Cryptographic service provider (CSP): RSA

Then, expand the "Key options" section and select the value 2048 in the "Key size:" field:

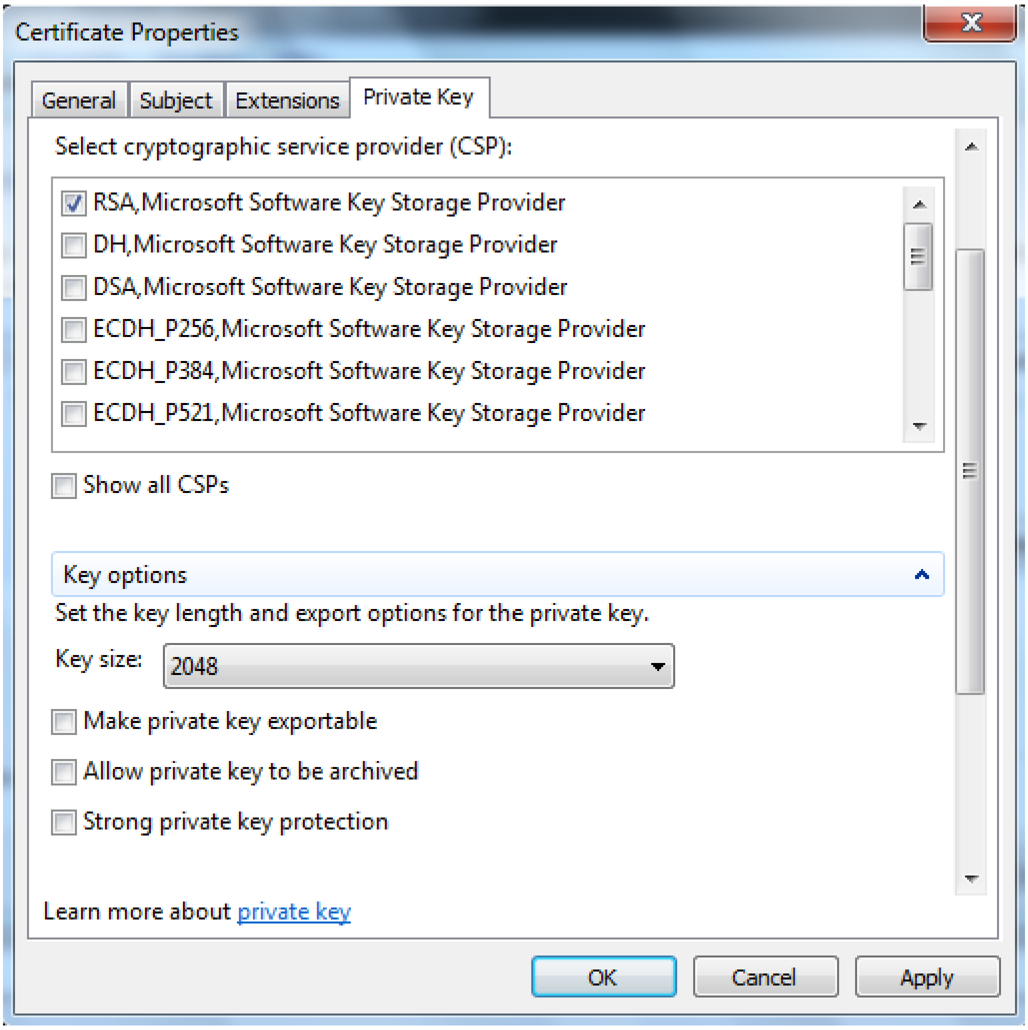


Figure 27 - Key size: 2048

Note: If you want to be able later on to export the private key, for example to save a backup of the certificate including its private key, you may select the option "Make private key exportable".

Scroll down and expand the "Select Hash Algorithm" section, and then select "sha256" in the "Hash Algorithm" field:

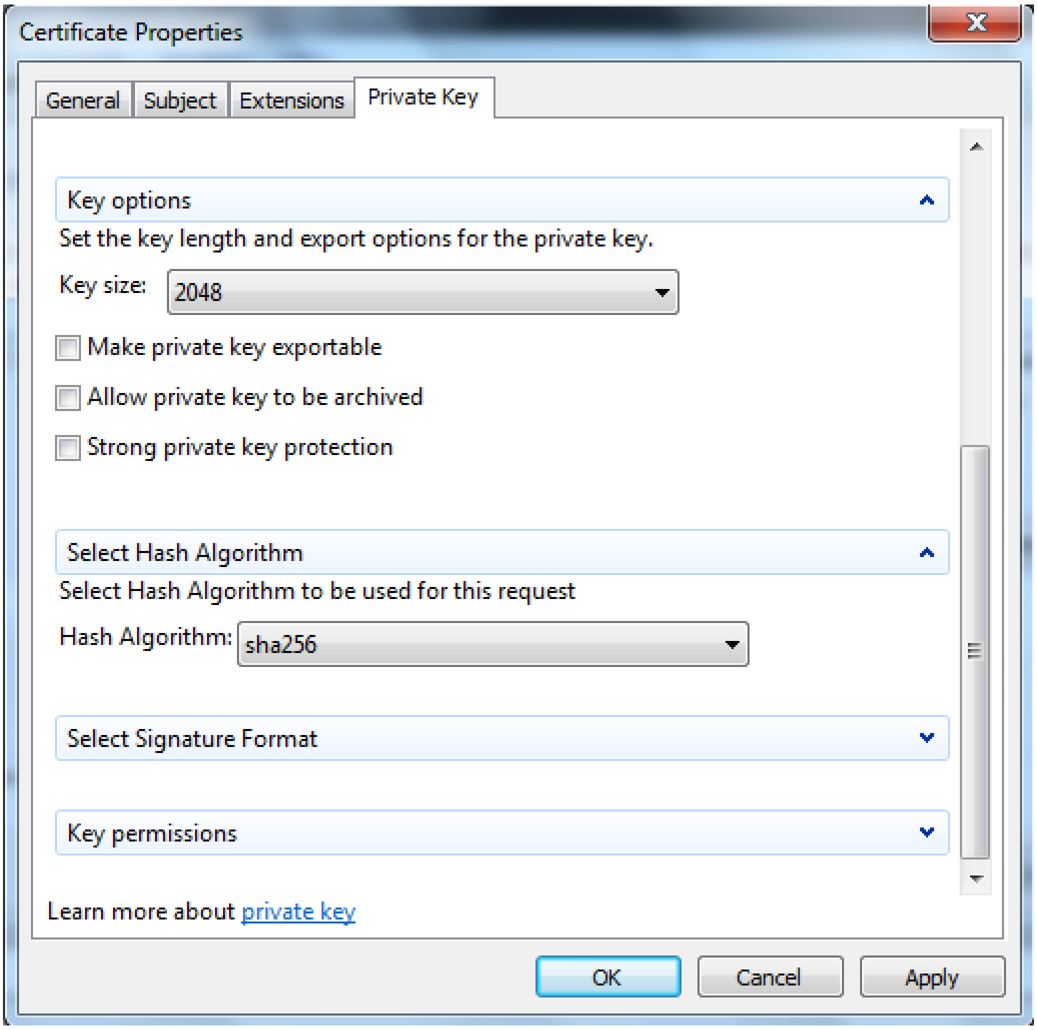


Figure 28 - Hash Algorithm: sha256

Click "Apply" and finally "OK" to close the "Certificate Properties" window. You will be brought back to the Certificate Enrollment window, which by now will be showing the application policies that were selected. Make sure it says "Server Authentication", "Client Authentication" and "IP security end system":

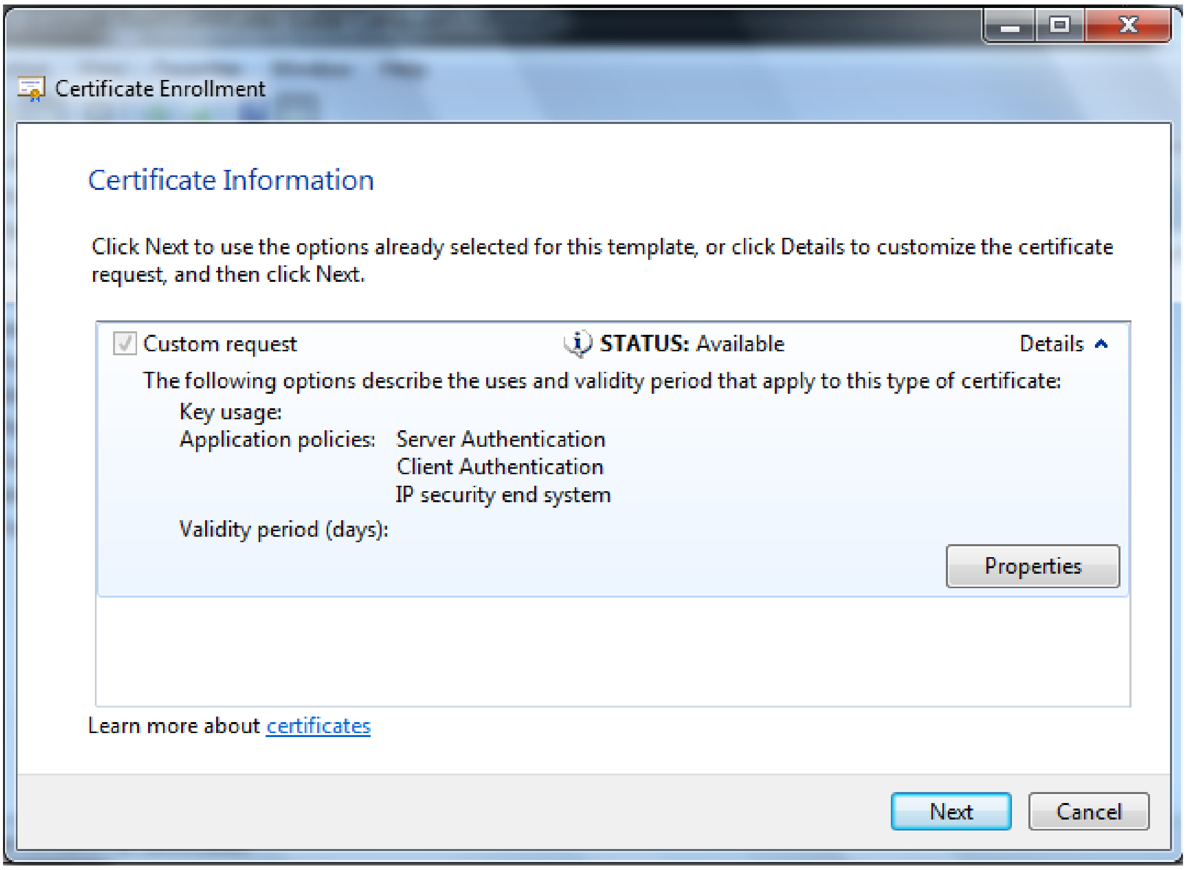


Figure 29 - Certificate Enrollment window, after configuration of the custom request

Click Next. Then, you will be asked to provide a file name for the file in which the certificate signing request will be stored:

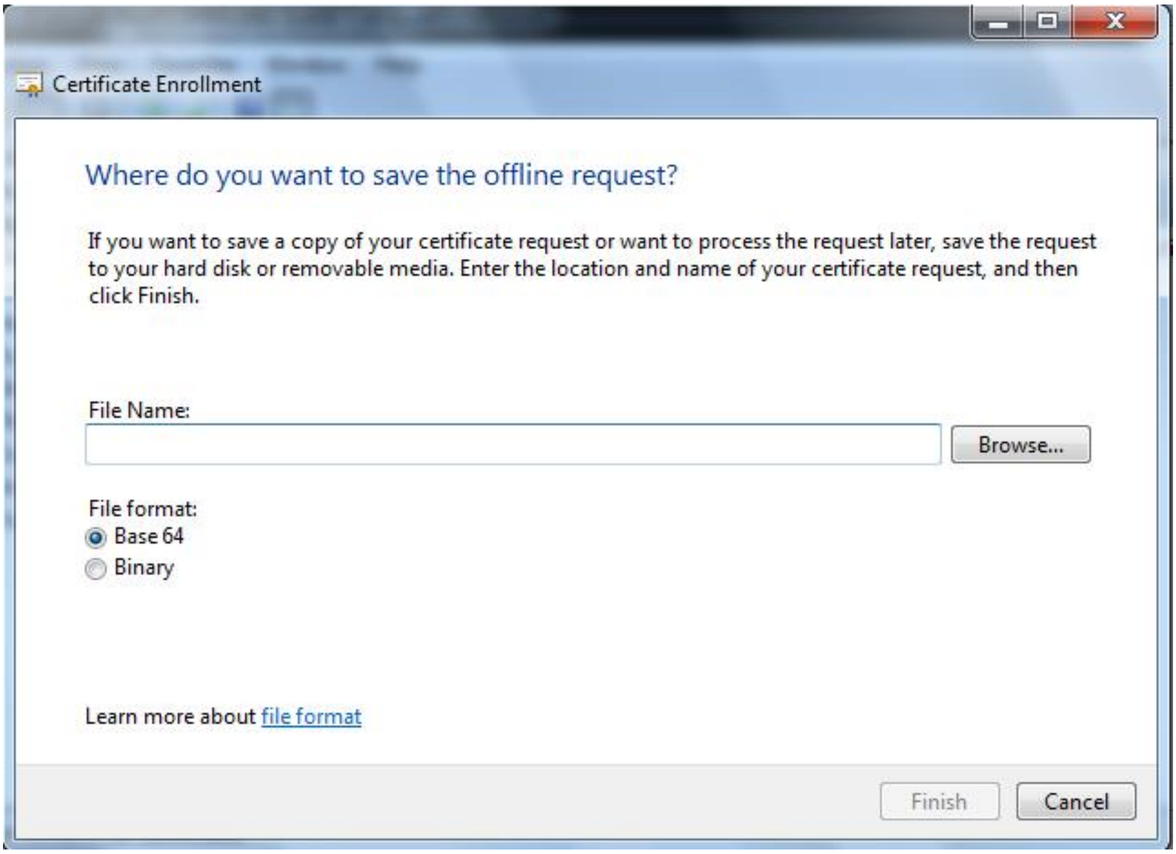


Figure 30 - Saving the certificate signing request to a file

Select the "Base 64" file format, specify a file name, for example "c:\tmp\PCALICE.csr" (assuming the folder c:\tmp already exists), and click Finish. The wizard will then disappear and the MMC console will not show any evidence of the certificate signing request (CSR) having been generated, but that is OK.

Note: If you don't specify an extension in the file name, it will be created without any extension. That is OK, but for clarity, we recommend that you give the file name the extension ".csr" (as in certificate signing request).

If everything went fine, you should now have a file (under the name you gave it) containing the CSR:

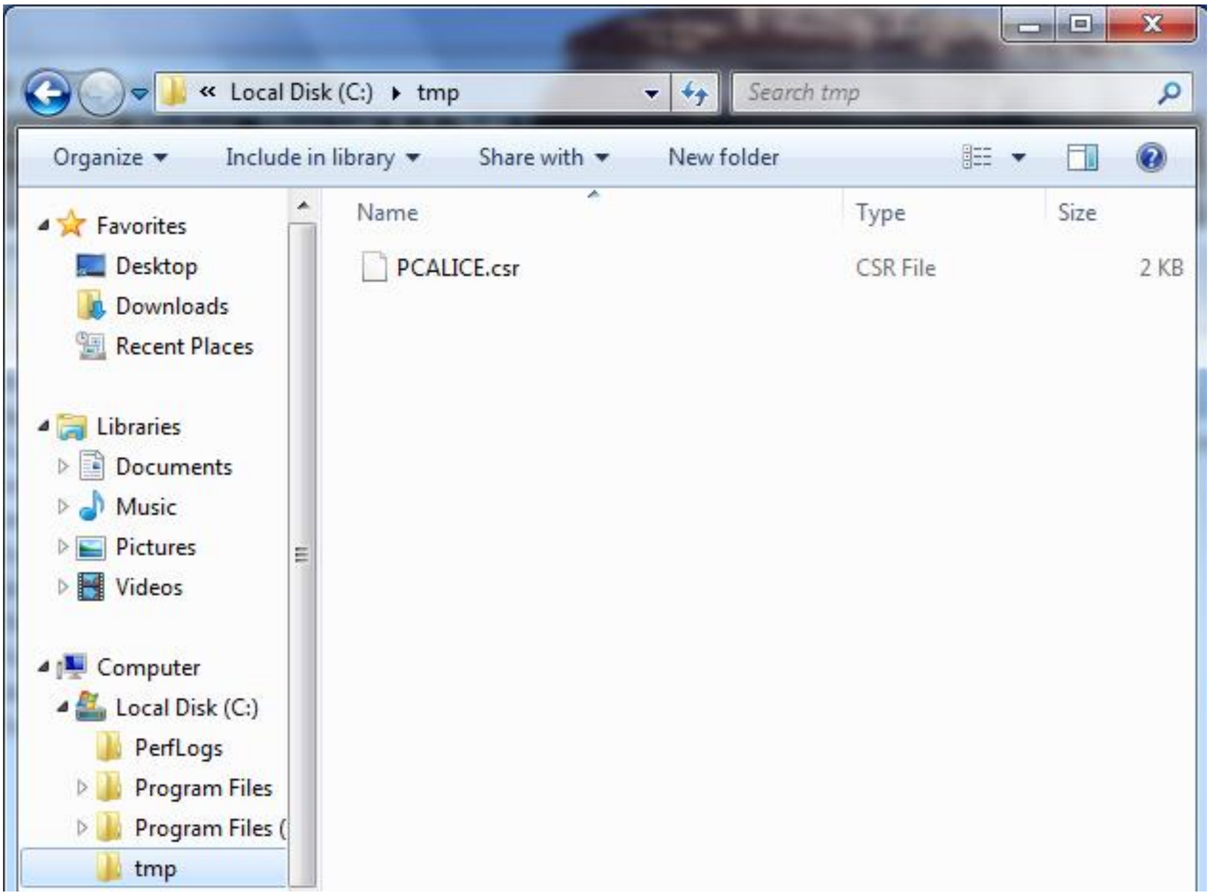


Figure 31 - File containing the CSR of PCALICE

Later on, we will need to copy this file to the CA, but first, we will generate PCBOB's CSR, as described in the next step.

3.6 Step 5: Generate a key pair and certificate signing request on PCBOB

Repeat on PCBOB the tasks detailed in the previous step, to generate its own key pair and certification signing request. Obviously, in this case the Common Name to use will need to be different, for example "PCBOB", and the CSR file should probably be named "PCBOB.csr".

3.7 Step 6: Install easy-rsa prerequisites on "linuxhost"

Easy-rsa depends on OpenSSL. If OpenSSL is not already available in your Linux system, you must install it.

If you are using an Ubuntu distribution, like we are in this example, you may install it by executing:

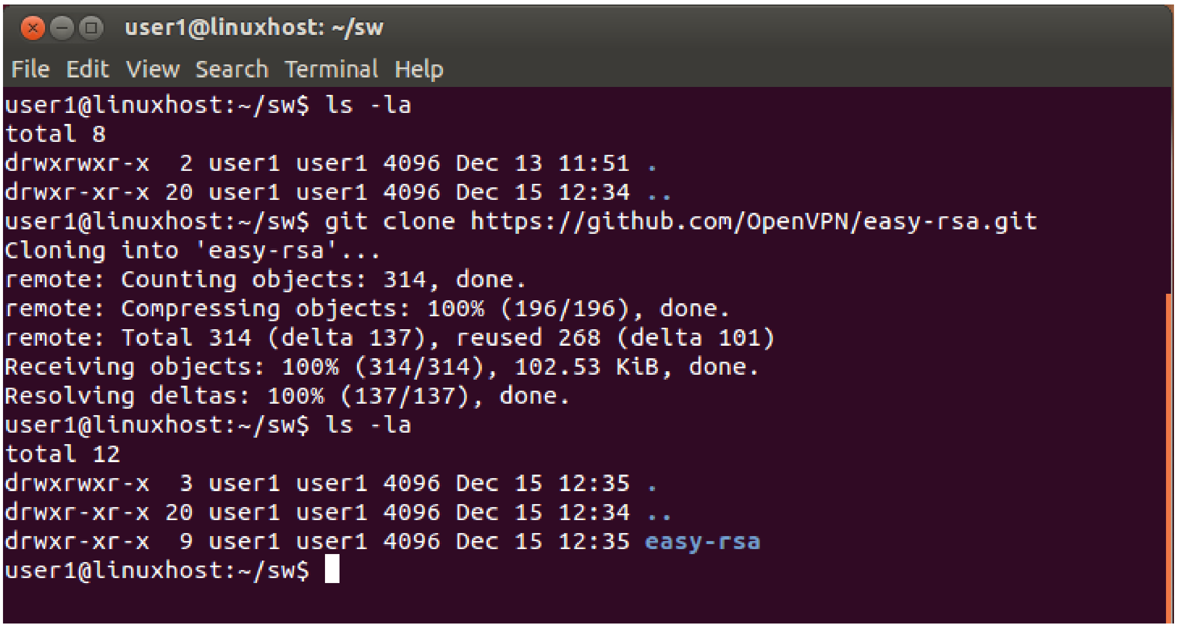
```
sudo apt-get install openssl
```

3.8 Step 7: Install easy-rsa on "linuxhost"

The latest version of easy-rsa is available at GitHub. You will need to clone the latest version of the code into your system by using the "git clone" command, as shown below. Be aware that the execution of the git clone command will create a new subdirectory called "easy-rsa" in your current directory, and all the code from easy-rsa will be located under that subdirectory. Therefore, before invoking the git command, make sure to set your current working directory to the directory where you wish easy-rsa to be copied to, as shown in the example below.

In this example, we will be installing easy-rsa in $HOME/sw of the user "user1" on our "linuxhost" system:

```
(log on as user1)
cd $HOME
mkdir sw
cd sw
git clone https://github.com/OpenVPN/easy-rsa.git
```



```
user1@linuxhost: ~/sw
File Edit View Search Terminal Help
user1@linuxhost:~/sw$ ls -la
total 8
drwxrwxr-x  2 user1 user1 4096 Dec 13 11:51 .
drwxr-xr-x 20 user1 user1 4096 Dec 15 12:34 ..
user1@linuxhost:~/sw$ git clone https://github.com/OpenVPN/easy-rsa.git
Cloning into 'easy-rsa'...
remote: Counting objects: 314, done.
remote: Compressing objects: 100% (196/196), done.
remote: Total 314 (delta 137), reused 268 (delta 101)
Receiving objects: 100% (314/314), 102.53 KiB, done.
Resolving deltas: 100% (137/137), done.
user1@linuxhost:~/sw$ ls -la
total 12
drwxrwxr-x  3 user1 user1 4096 Dec 15 12:35 .
drwxr-xr-x 20 user1 user1 4096 Dec 15 12:34 ..
drwxr-xr-x  9 user1 user1 4096 Dec 15 12:35 easy-rsa
user1@linuxhost:~/sw$
```

Figure 32 - Installation of easy-rsa from github

3.9 Step 8: Configure easy-rsa on "linuxhost" to accept extensions specified in certificate signing requests

By default, easy-rsa discards any extensions specified in certificate signing requests, and generates certificates using only extensions that are defined in its own configuration files. Since the CSRs generated in the Windows systems for IPsec contain extensions that are unknown to easy-rsa but required by Windows to use the for IPsec, if we used easy-rsa with its default configuration, the digital certificates generated would not contain those extensions and consequently they would not be valid for their intended use.

One way to solve this problem is configuring easy-rsa to accept (copy into the final certificate) any extensions specified in certificate signing requests. Of course, this introduces a potential security risk, because a maliciously crafted CSR could contain extra extensions, declaring the certificate good for uses beyond what should be authorized for it. For example, someone sending in a CSR with a declared purpose of IPsec authentication, could include in the CSR an extension that would allow them to use the certificate for signing software. Thus, care should be taken to review each CSR before processing it and to make sure that it doesn't include unwanted extensions.

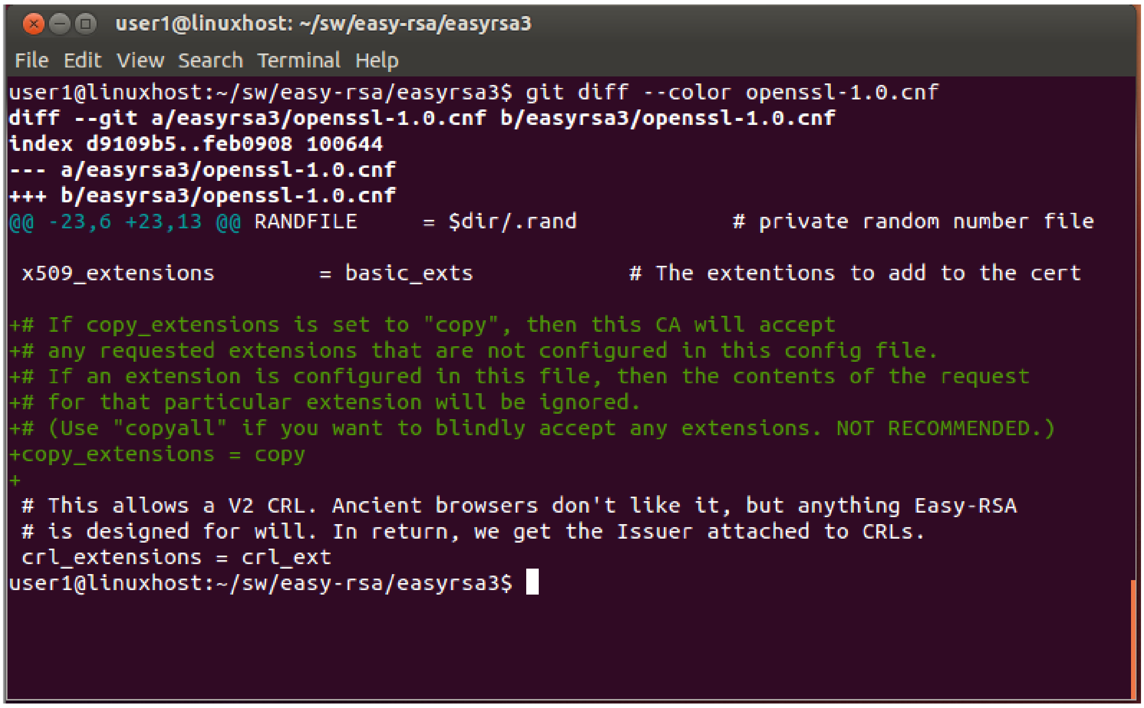
Configuring easy-rsa to accept extensions from CSRs involves two simple modifications to the default easy-rsa configuration.

The first modification is to insert the following line in the "[CA_default]" section of the file "easy-rsa/easyrsa3/openssl-1.0.cnf":

```
copy_extensions = copy
```

This configuration option tells easy-rsa to copy into the final certificate any extension that is present in the CSR and that is not defined in easy-rsa's configuration files. If an extension is configured in easy-rsa's configuration files and also in the CSR, the value in the configuration files will take precedence and be copied onto the final certificate, ignoring what is defined in the CSR. However, if an extension is present in the CSR and not defined in easy-rsa's configuration files, the extension will be copied onto the final certificate without modification.

Just for reference, the following figure shows this edition of the configuration file, in git diff format (a few lines of comments have been added too, for clarity):



```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File Edit View Search Terminal Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ git diff --color openssl-1.0.cnf
diff --git a/easyrsa3/openssl-1.0.cnf b/easyrsa3/openssl-1.0.cnf
index d9109b5..feb0908 100644
--- a/easyrsa3/openssl-1.0.cnf
+++ b/easyrsa3/openssl-1.0.cnf
@@ -23,6 +23,13 @@ RANDFILE      = $dir/.rand            # private random number file

 x509_extensions        = basic_exts            # The extentions to add to the cert

+# If copy_extensions is set to "copy", then this CA will accept
+# any requested extensions that are not configured in this config file.
+# If an extension is configured in this file, then the contents of the request
+# for that particular extension will be ignored.
+# (Use "copyall" if you want to blindly accept any extensions. NOT RECOMMENDED.)
+copy_extensions = copy
+
 # This allows a V2 CRL. Ancient browsers don't like it, but anything Easy-RSA
 # is designed for will. In return, we get the Issuer attached to CRLs.
 crl_extensions = crl_ext
user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 33 - Configuration option "copy_extensions = copy "added to openssl-1.0.cnf"

The second modification involves defining a new type of certificate that easy-rsa will know how to generate, adding to the built-in types "ca", "client" and "server".

This may sound more complicated than it is. All you have to do is create a new file in the subdirectory "x509-types" of easy-rsa (easy-rsa/easyrsa3/x509-types/), naming it with the same name you want easy-rsa to use to refer to the new type of certificate. For example, you may create a file with the name "freeformat" (as in free formatted requests). The contents of the file must be the extensions that you want easy-rsa to add to all certificates generated using this type definition, overriding the values for those contained in the CSR, if present. For example, you may create the file with the following contents (note that lines starting with the hash sign are just comments):

```
# X509 extensions for a free format request
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
```

These 3 extensions will make sure the certificates contain an extension stating that they cannot be used to sign other certificates, another extension with the hash of the certificate, and finally, another extension with information about the CA that signed the certificate. Any other extensions specified in the CSR will be copied unmodified onto the final certificate (provided the copy_extensions option is configured as explained before).

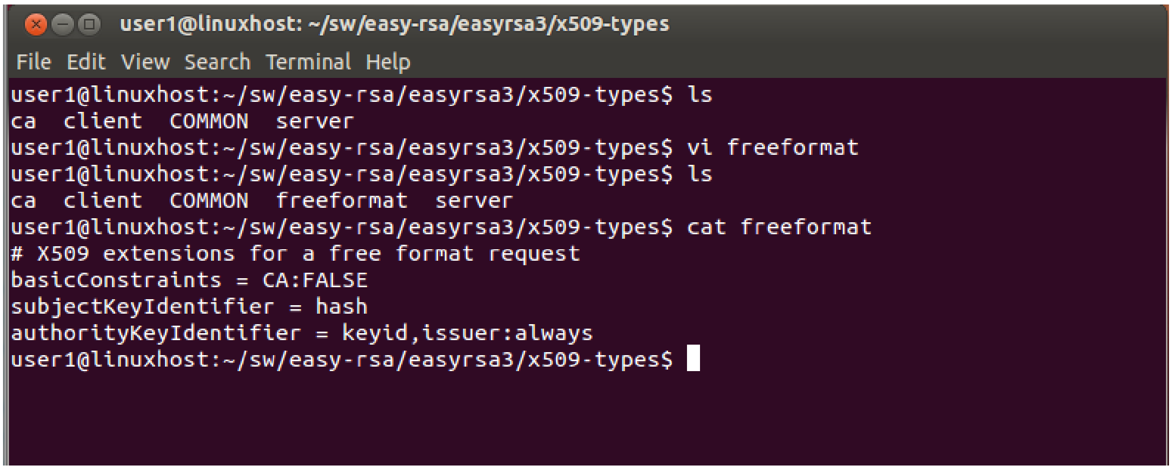


Figure 34 - Creation of file easy-rsa/easyrsa3/x509-types/freeformat

Note: If you want to save these edits using git, you may do so by executing:

```
cd $HOME/sw/easy-rsa/
git add easyrsa3/openssl-1.0.cnf
git add easyrsa3/x509-types/freeformat
git commit -m "Add freeformat certificate type"
```

3.10 Step 9: Create your own root CA using easy-rsa

Creating a brand new root CA using easy-rsa: you just need to go the directory where the executable easyrsa is located, and execute the following commands:

```
cd $HOME/sw/easy-rsa/easyrsa3
./easyrsa init-pki
./easyrsa build-ca
```

When you execute the second command, you will be asked for a pass phrase to protect the private key of the CA (ca.key), for a name (Common Name) for the Root CA. In this example we will name it "Example Root CA", as shown in the following figure:

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File  Edit  View  Search  Terminal  Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa init-pki

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/user1/sw/easy-rsa/easyrsa3/pki

user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa build-ca
Generating a 2048 bit RSA private key
..................................................................+++
..............+++
writing new private key to '/home/user1/sw/easy-rsa/easyrsa3/pki/private/ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:Example Root CA

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/user1/sw/easy-rsa/easyrsa3/pki/ca.crt

user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 35 - Generation of a new root CA (Example Root CA)

At this point the new root CA should have been created, and its certificate saved to a newly created file named "ca.crt", in the pki subdirectory. This certificate is the one that will need to be imported into the Trusted Root Certification Authorities of the systems, but that will be done in a later step.

3.11 Step 10: Transfer the CSR files to linuxhost

Copy the certificate signing request files, PCALICE.csr and PCBOB.csr, over to linuxhost. You may leave them anywhere in the file system, for example under /tmp:

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File  Edit  View  Search  Terminal  Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ls /tmp/*.csr
/tmp/PCALICE.csr  /tmp/PCBOB.csr
user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 36 - CSR files copied over to linuxhost

3.12 Step 11: Import the CSRs into easy-rsa

In order for easy-rsa to work with the CSRs, it first needs to have them imported into its own directory structure, and given a name that will be used in later commands. Therefore, you must import each CSR and give it short name, using the following commands:

```
./easyrsa import-req /tmp/PCALICE.csr PCALICE
./easyrsa import-req /tmp/PCBOB.csr PCBOB
```

The first argument is the name of the file containing the CSR (full or relative path), and the second argument is the name that will be used in easy-rsa as a basename for all files related to that CSR, including, eventually, the final certificate. You may choose any name, but for your own sanity, you may want to stick to using the common name, as shown in the example.

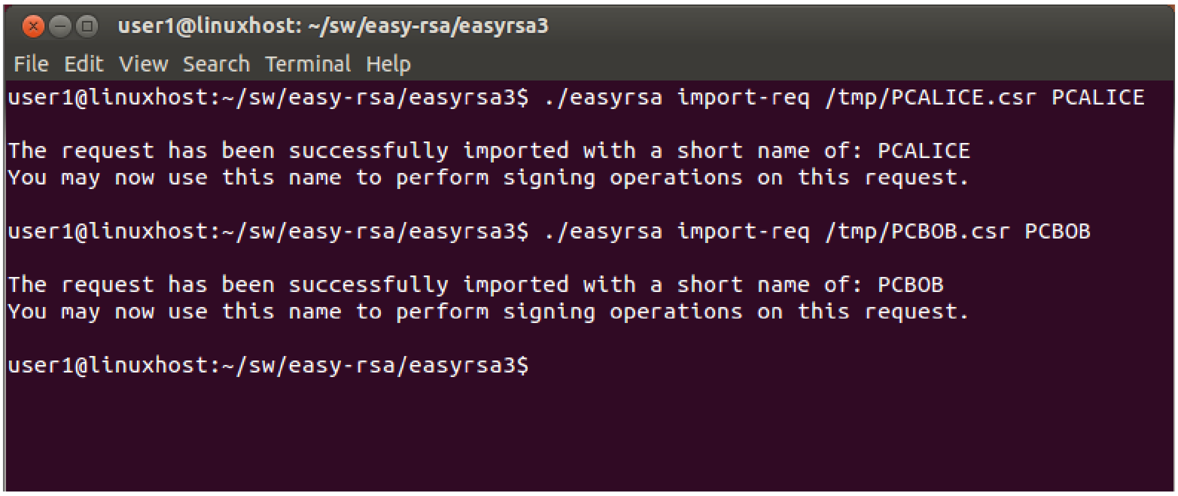


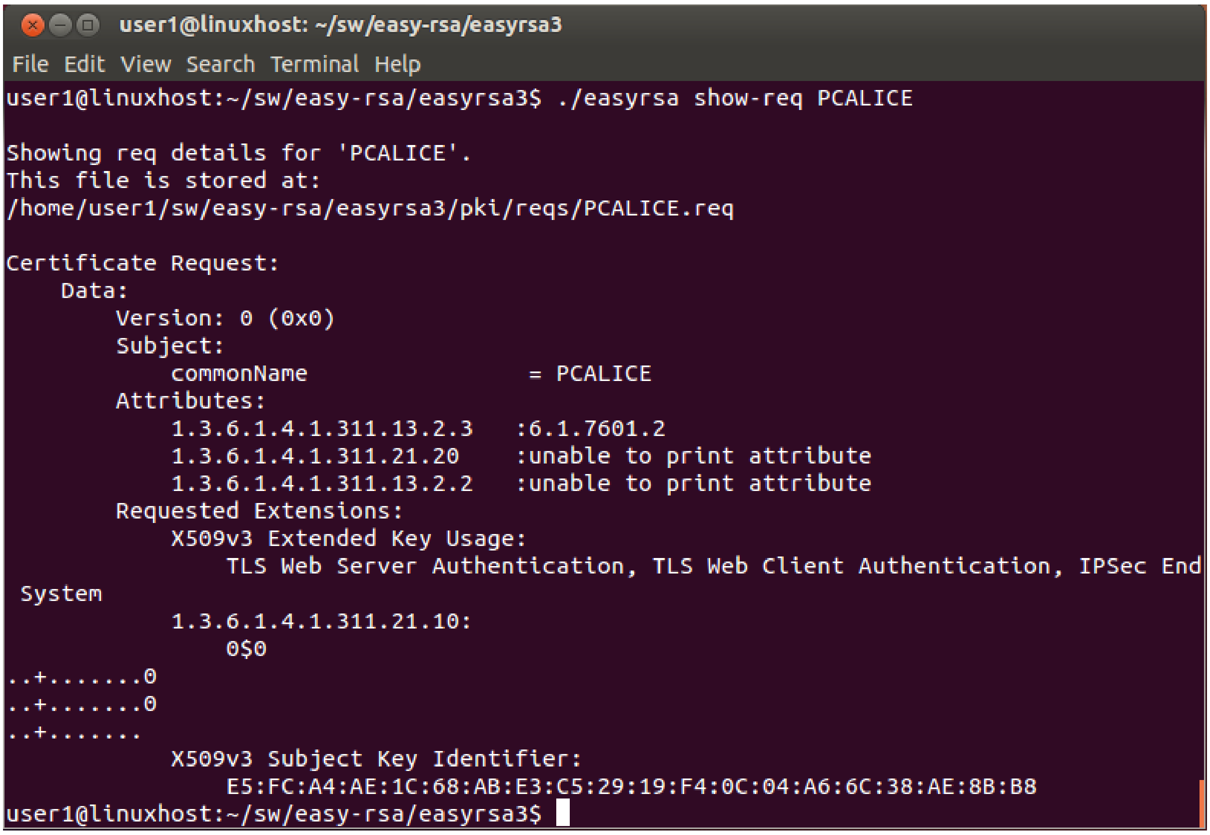
Figure 37 - Importing the CSRs

3.13 Step 12: Generate the certificates by signing the CSRs

Prior to signing the CSRs, you may want to review the contents of each CSR, to verify that they do not contained unwanted extensions, for example. You may do so using the easyrsa command show-req, specifying the short name you gave to the CSR when importing it:

```
./easyrsa show-req PCALICE
```

The following figure shows the kind of output that you will get:



```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File Edit View Search Terminal Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa show-req PCALICE

Showing req details for 'PCALICE'.
This file is stored at:
/home/user1/sw/easy-rsa/easyrsa3/pki/reqs/PCALICE.req

Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject:
            commonName                = PCALICE
        Attributes:
            1.3.6.1.4.1.311.13.2.3   :6.1.7601.2
            1.3.6.1.4.1.311.21.20    :unable to print attribute
            1.3.6.1.4.1.311.13.2.2   :unable to print attribute
        Requested Extensions:
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication, IPSec End
 System
            1.3.6.1.4.1.311.21.10:
                0$0
..+.......0
..+.......0
..+.......
            X509v3 Subject Key Identifier:
                E5:FC:A4:AE:1C:68:AB:E3:C5:29:19:F4:0C:04:A6:6C:38:AE:8B:B8
user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 38 - Displaying the contents of the CSR from PCALICE

Unfortunately, some of the object identifiers cannot be translated to English by easyrsa, and the contents of some extensions do not get completely displayed in a readable manner. This could most probably fixed, but for the purpose of this example, we will just leave it as it is.

When you are sure that the CSRs are valid to you, you may proceed to sign them, thus generating the corresponding certificates. This is achieved using the easyrsa sign-req command that takes two arguments: the type of certificate to be generated (the file name of any of the files under the x509-types subfolder), and the short name of the CSR (given at import time):

```
./easyrsa sign-req <type> <filename_base>
```

In the case of our example:

```
./easyrsa sign-req freeformat PCALICE
./easyrsa sign-req freeformat PCBOB
```

For each signature, some details of the request will be displayed and you will be asked to confirm that you want to proceed by typing "yes", and then you will be prompted to type in the pass phrase that protects the private key of the CA (that you chose when you created the CA), and if everything goes well, a certificate will be generated and stored under subfolder "pki/issued":

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File  Edit  View  Search  Terminal  Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa sign-req freeformat PCALICE


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a freeformat certificate for 3650 days:

subject=
    commonName                = PCALICE


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /home/user1/sw/easy-rsa/easyrsa3/openssl-1.0.cnf
Enter pass phrase for /home/user1/sw/easy-rsa/easyrsa3/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'PCALICE'
Certificate is to be certified until Dec 14 17:57:59 2023 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /home/user1/sw/easy-rsa/easyrsa3/pki/issued/PCALICE.crt

user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 39 - Confirming the signature of the CSR PCALICE

After you sign both CSRs, PCALICE and PCBOB, you should have the corresponding certificates ready in the pki/issued subfolder:

```
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ls pki/issued
PCALICE.crt  PCBOB.crt
```

Figure 40 - Location of certificates issued by the CA

Remember as well that the certificate of the CA, that you will also need to import into both systems later on, is located directly in the pki subfolder:

```
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ls pki/ca.crt
pki/ca.crt
```

Figure 41 - Location of the certificate of the CA itself

3.14 Step 13: Copy the newly generated certificates and that of the CA to the corresponding systems

Now that you have all the necessary certificates, you need to copy them from linuxhost to the appropriate systems.

Copy the following files from linuxhost to any folder (e.g. C:\tmp\) in PCALICE:

File name	Location in linuxhost	Description
ca.crt	$HOME/sw/easy-rsa/easyrsa3/pki/ca.crt	Certificate of the CA
PCALICE.crt	$HOME/sw/easy-rsa/easyrsa3/pki/issued/PCALICE.crt	Certificate of PCALICE

and copy the following files from linuxhost to PCBOB:

File name	Location in linuxhost	Description
ca.crt	$HOME/sw/easy-rsa/easyrsa3/pki/ca.crt	Certificate of the CA
PCBOB.crt	$HOME/sw/easy-rsa/easyrsa3/pki/issued/PCBOB.crt	Certificate of PCBOB

3.15 Step 14: On PCALICE, import the CA certificate (ca.crt) and its own certificate (PCALICE.crt)

On PCALICE, go back to the "Certificates (Local Computer)" snap-in of the Microsoft Management Console (re-open it following the steps described earlier, if you closed it):

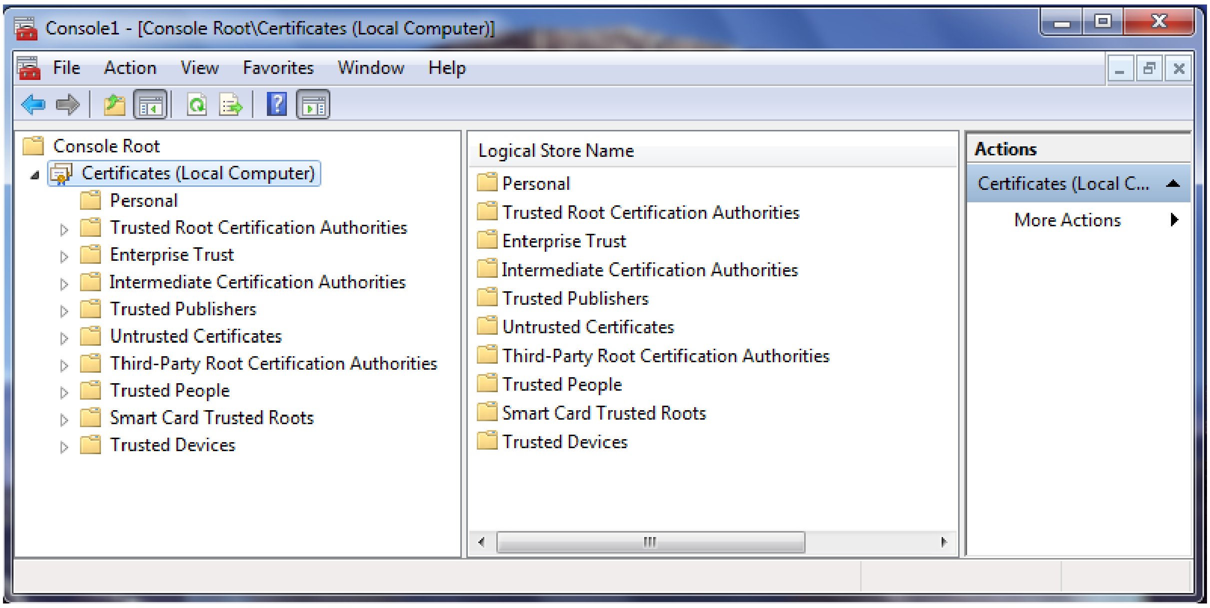


Figure 42 - Certificates (Local Computer) MMC snap-in in PCALICE

First, import the certificate of the CA into the Trusted Root Certification Authorities store. To do so:

Select the "Trusted Root Certification Authorities", right click on it, and select "All Tasks -> Import...". That will launch the Certificate Import Wizard:

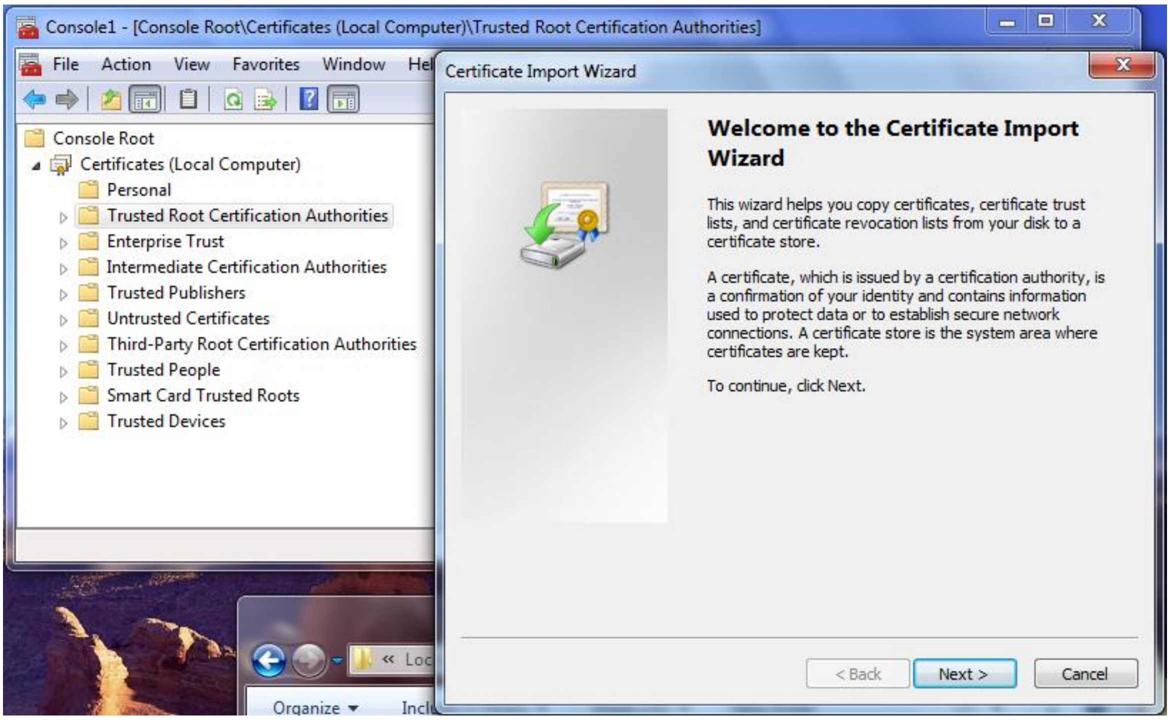


Figure 43 - Certificate Import Wizard

Click Next, and fill in the full path to the file containing the certificate from the CA (e.g.: C:\tmp\ca.crt):

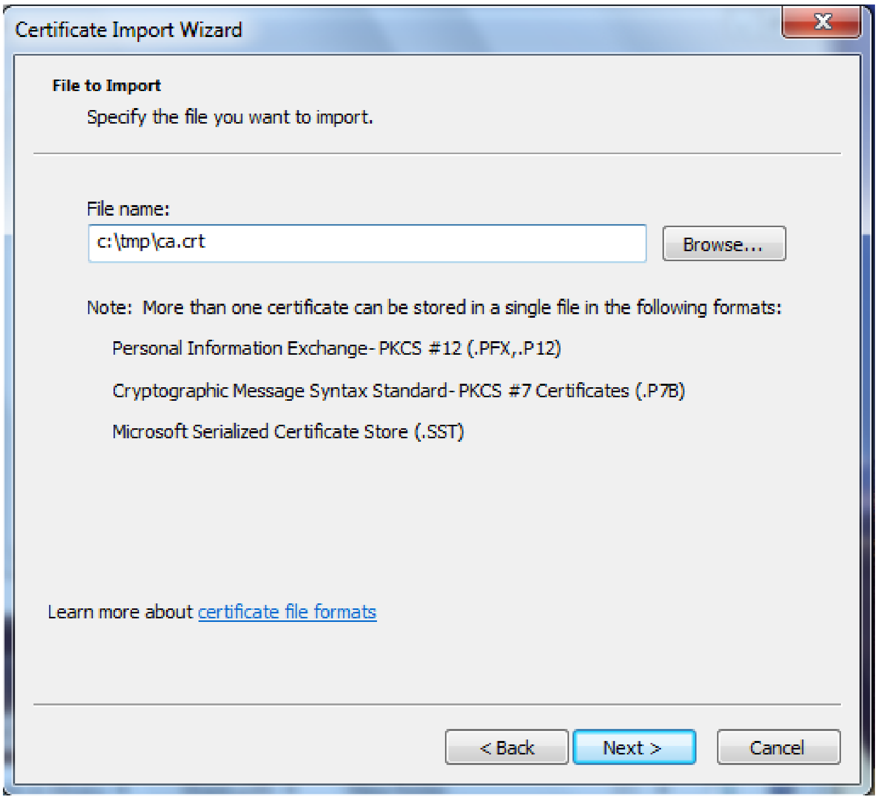


Figure 44 - Importing ca.crt

Click Next, and on the next window verify that the certificate store where the certificate will be placed is "Trusted Root Certification Authorities":

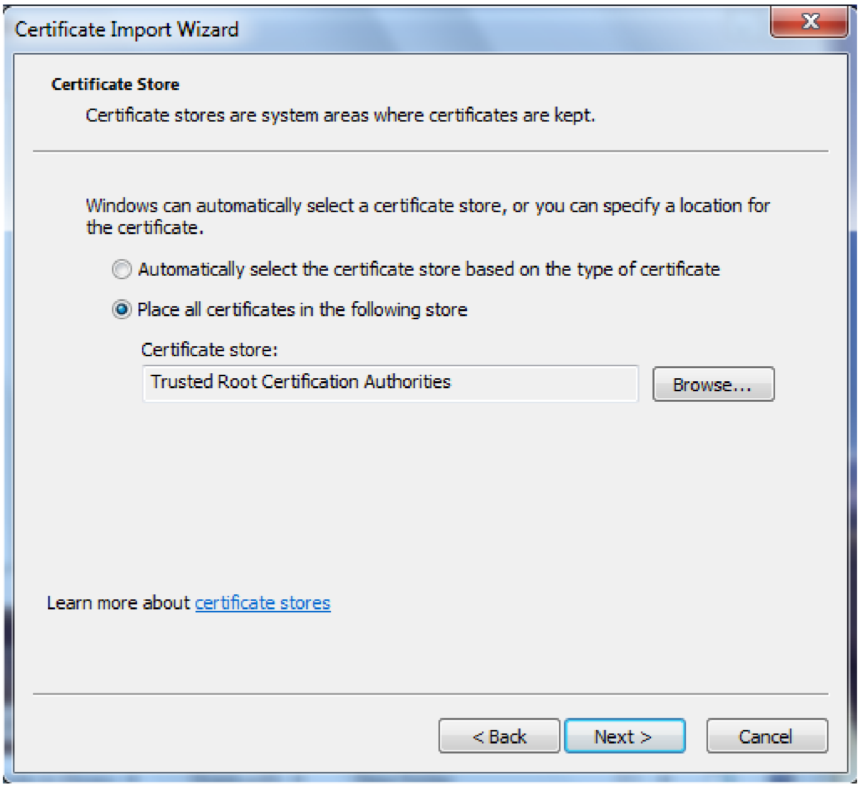


Figure 45 - Selection of the certificate store where the certificate will be placed

Click Next and finally Finish, to complete the import of the certificate. After a few seconds, you should get a pop-up window confirming that the import was successful:

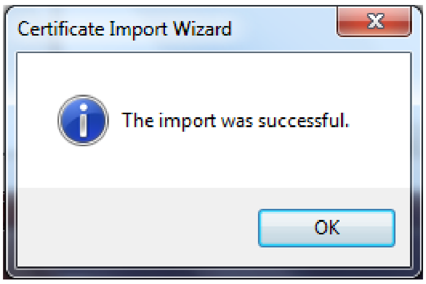


Figure 46 - Confirmation of a successful import of a certificate

Click OK to dismiss it.

You may then verify that the certificate was successfully imported by selecting the target container and checking that the certification authority (Example Root CA, in this example) appears on the list of trusted root CAs:

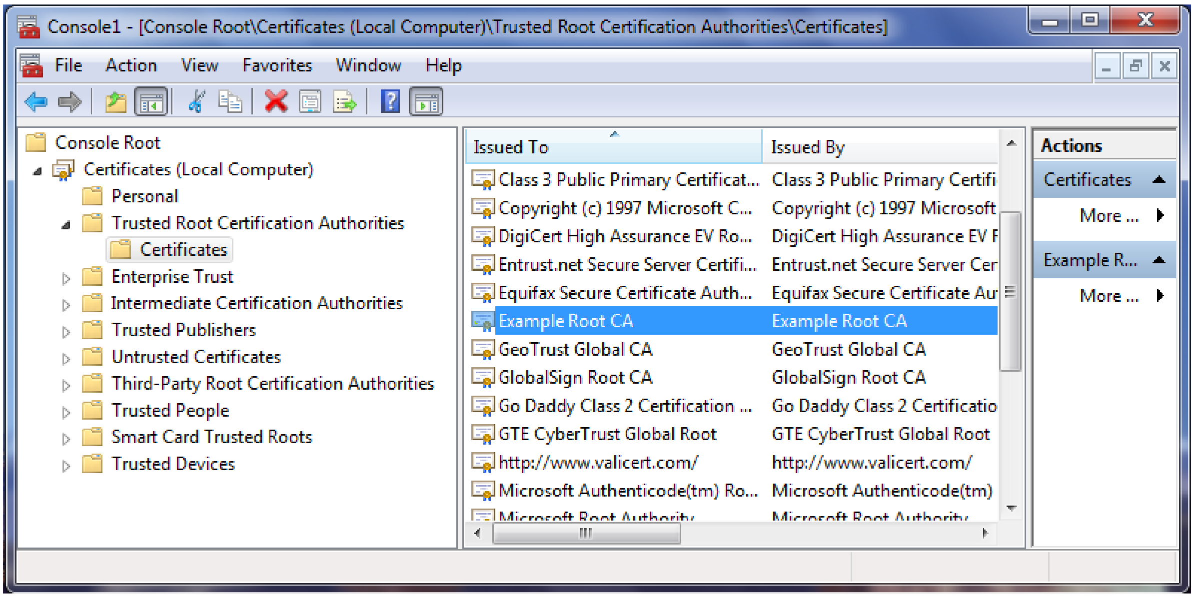


Figure 47 - Imported Root CA certificate

Then, import the certificate PCALICE.crt into the Personal store of PCALICE. To do so:

Select the "Personal" container, right click on it, and select "All Tasks -> Import...". That will launch the Certificate Import Wizard:

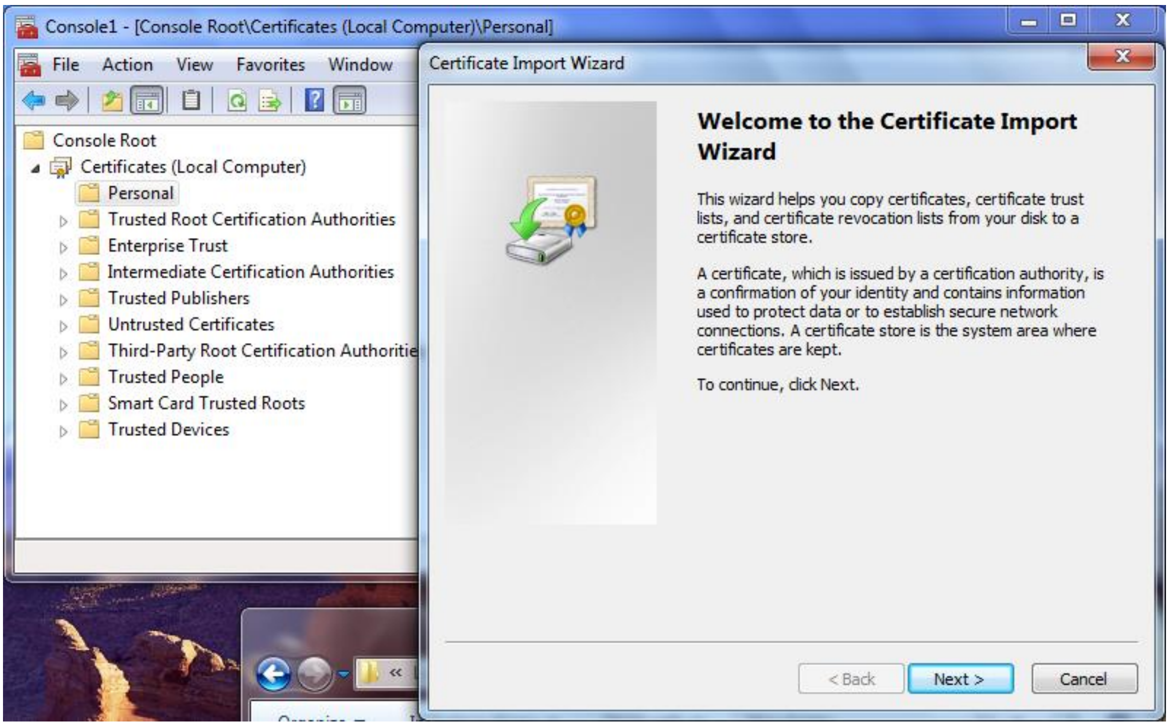


Figure 48 - Certificate Import Wizard to import the personal (computer) certificate of PCALICE

Click Next, and fill in the full path to the file containing the PCALICE's certificate (e.g.: C:\tmp\PCALICE.crt):

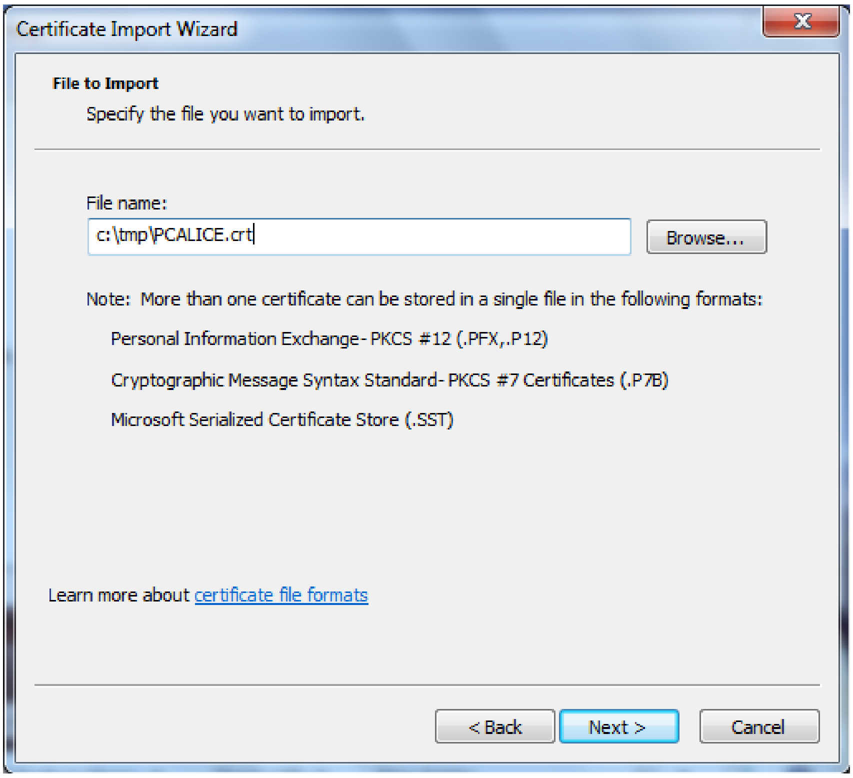


Figure 49 - Importing PCALICE.crt

Click Next and on the next window verify that the certificate store where the certificate will be placed is the "Personal" store:

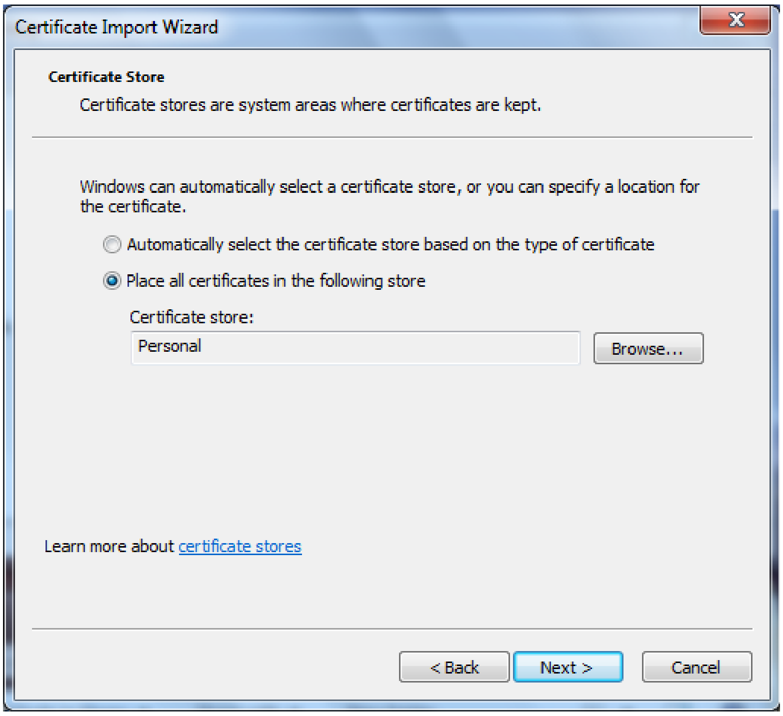


Figure 50 - Selection of the certificate store where the certificate will be placed

Click Next and finally Finish, to complete the import of the certificate. After a few seconds, you should get a pop-up window confirming that the import was successful:

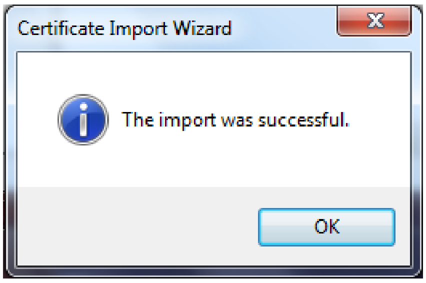


Figure 51 - Confirmation of a successful import of a certificate

Click OK to dismiss it.

You may then verify that the certificate was successfully imported by selecting the target container (Personal\Certificates) and checking that the certificate appears on the list:

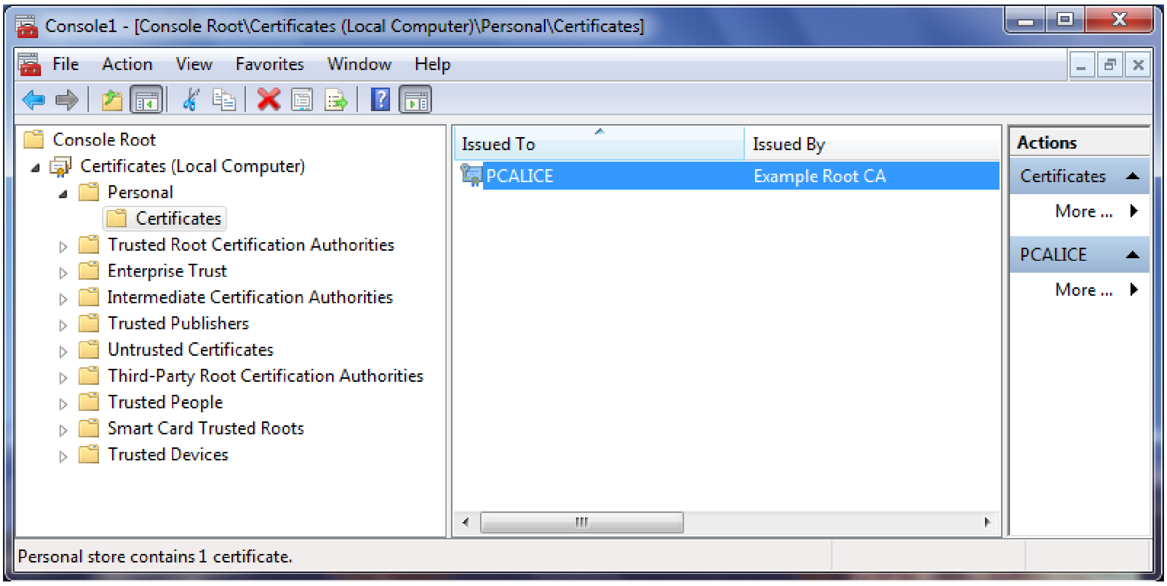


Figure 52 - Imported Personal (Local Computer\Personal) certificate

You may double click the certificate to display its contents:

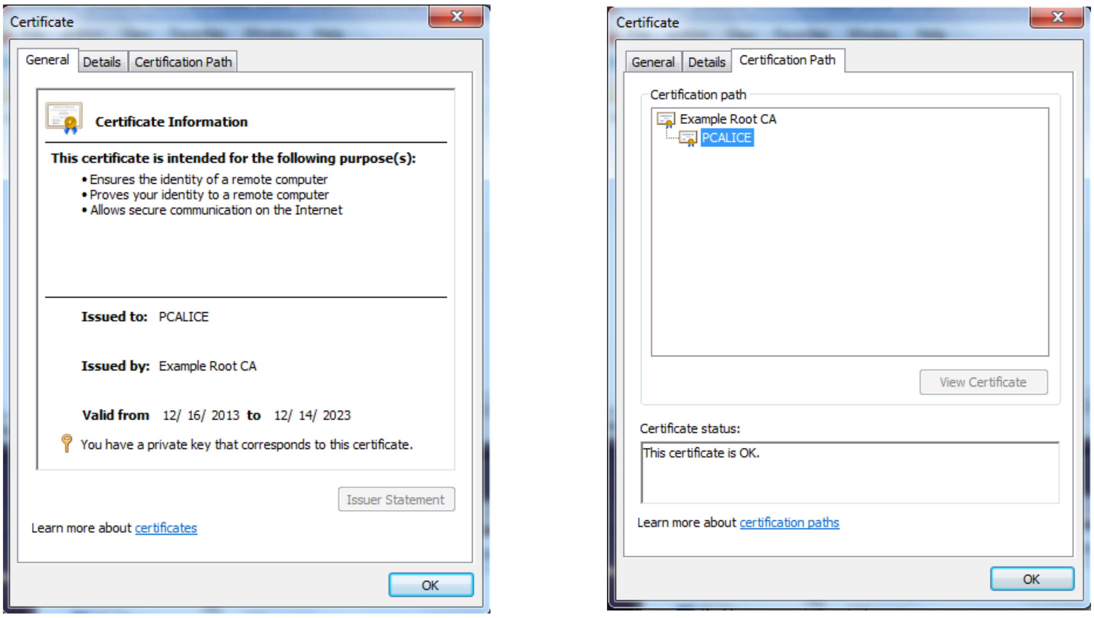


Figure 53 - PCALICE certificate

Note: In the General tab, among other information, the following sentence should be displayed: "You have a private key that corresponds to this certificate". If it is not, you probably imported the wrong certificate (PCBOB.crt into PCALICE or vice versa). Verify all previous steps.

3.16 Step 15: On PCALICE, configure default options for IPsec

Still on PCALICE, go back to the Windows Firewall with Advanced Security console snap-in (re-open it following the steps described earlier, if you closed it) and select the top level container:

Figure 54 - Top level container of Windows Firewall with Advanced Security console snap-in

Right click on the top level container you just selected, and select "Properties". The following window should appear:

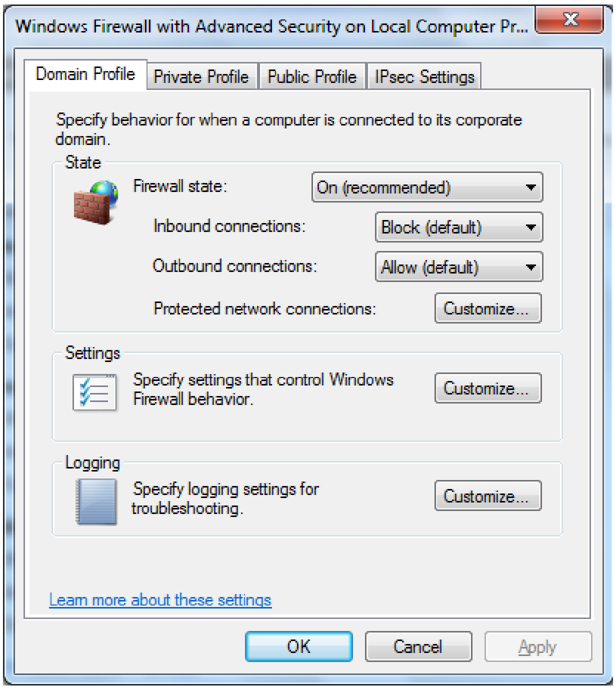


Figure 55 - Windows Firewall with Advanced Security properties

Select the "IPsec Settings" tab:

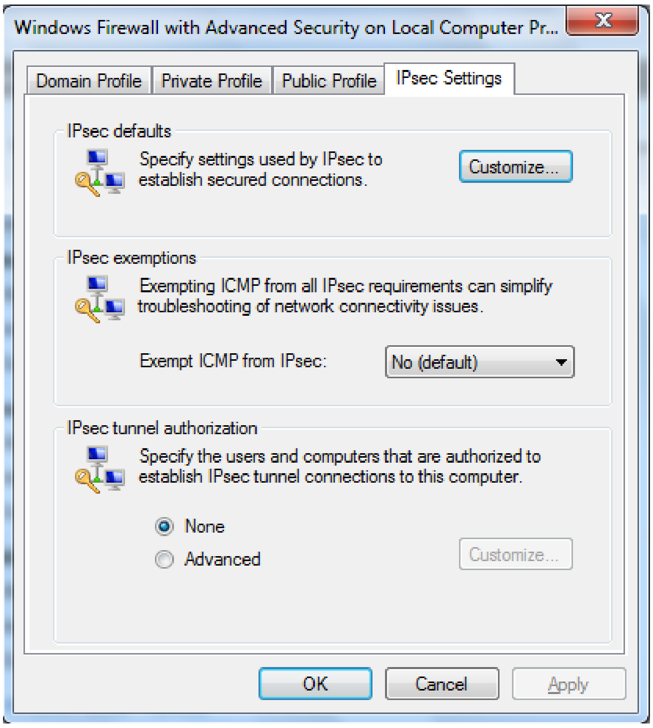


Figure 56 - IPsec Settings tab

Click the button "Customize..." of the section "IPsec defaults". The following window will appear:

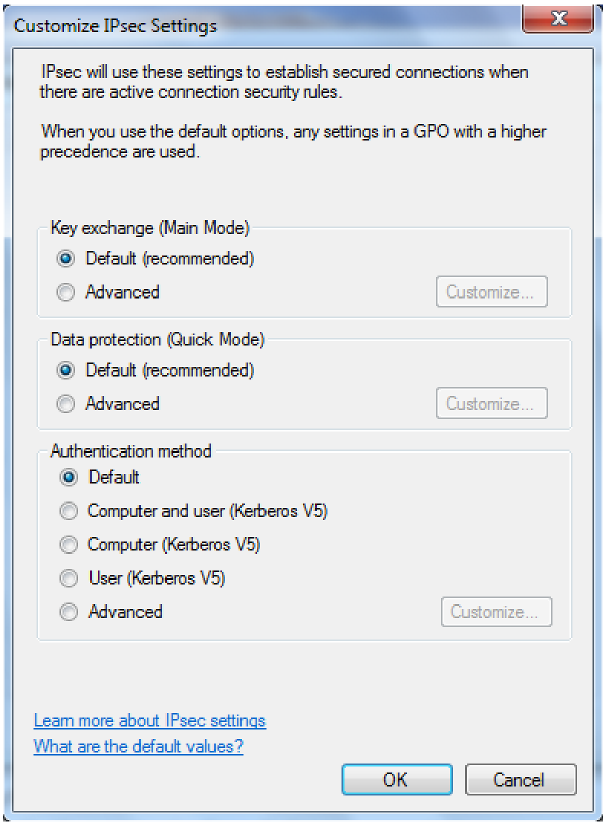


Figure 57 - Customize IPsec Settings window

Select the option "Advanced" in the "Key exchange (Main Mode)" section, and click its "Customize..." button. The following window will appear:

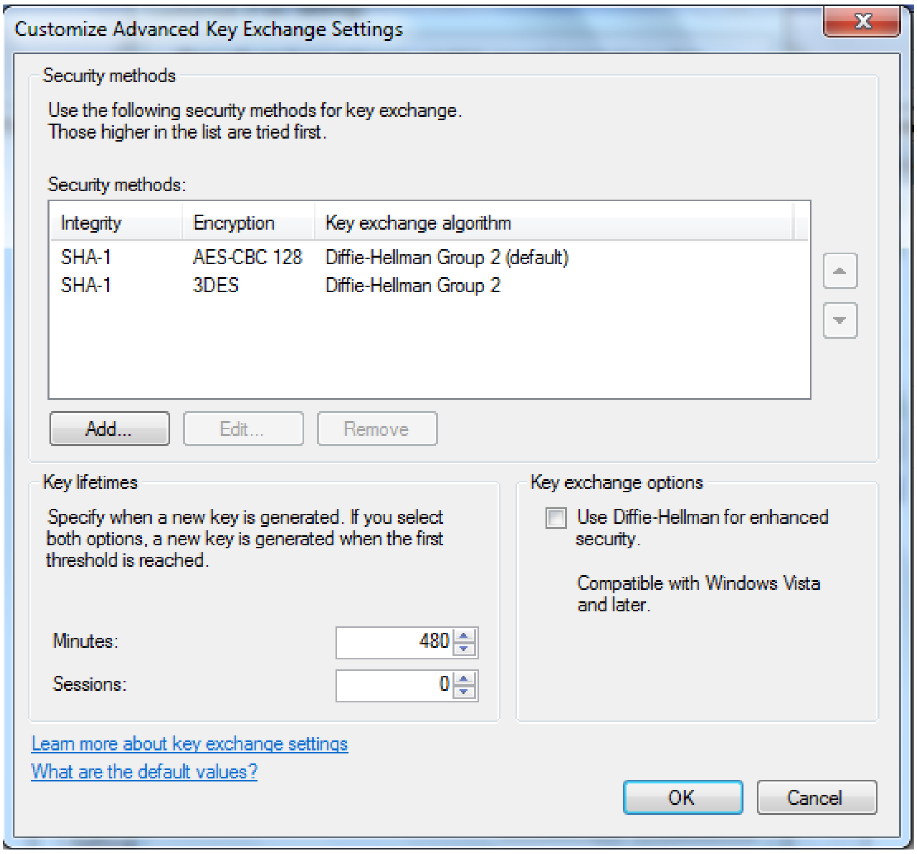


Figure 58 - Main Mode security methods, with the default configuration

First, remove all security methods, by selecting each method in turn and clicking the Remove button, until the "Security methods" list is empty:

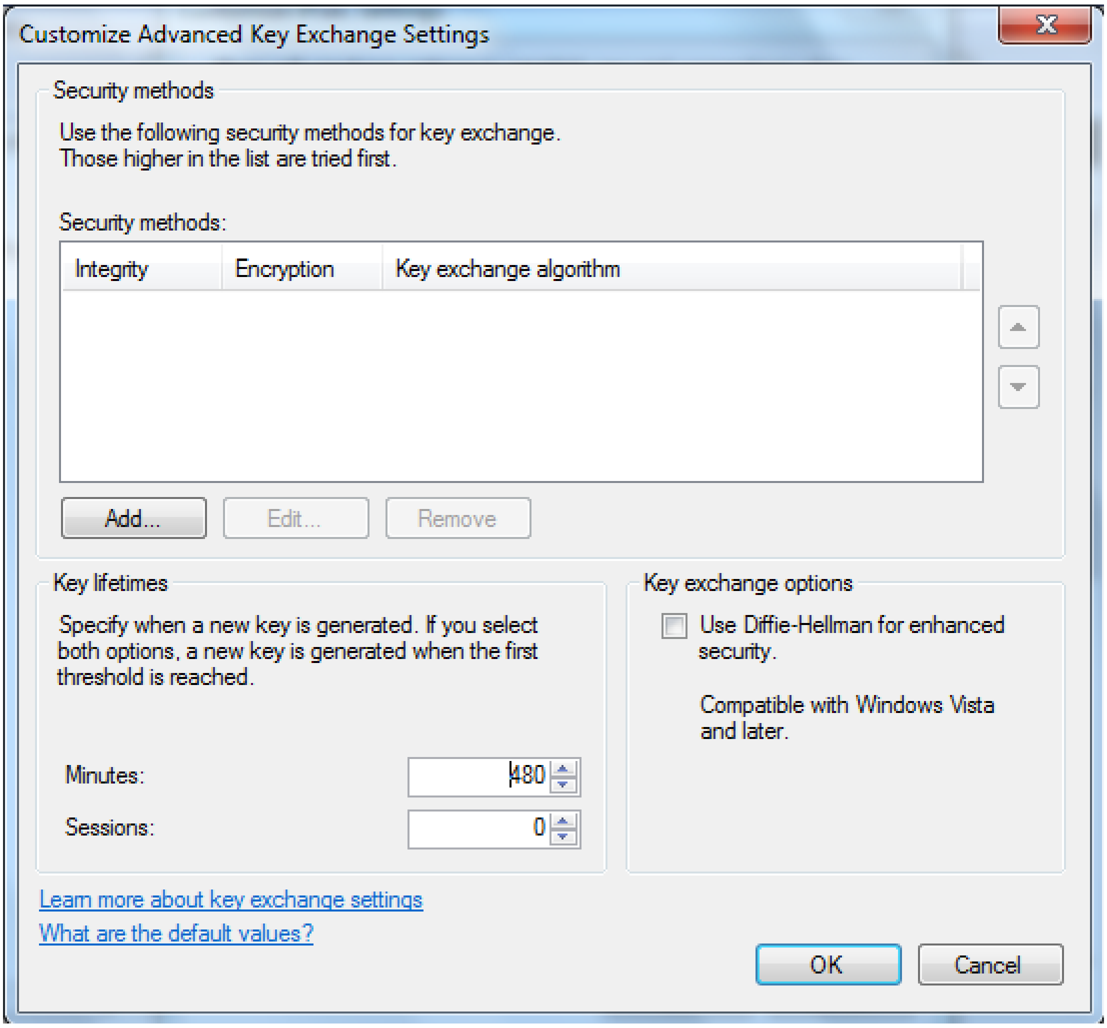


Figure 59 - Main Mode security methods. All security methods removed.

Then, add a single new method with the configuration of your liking. In this example we will use the following options:

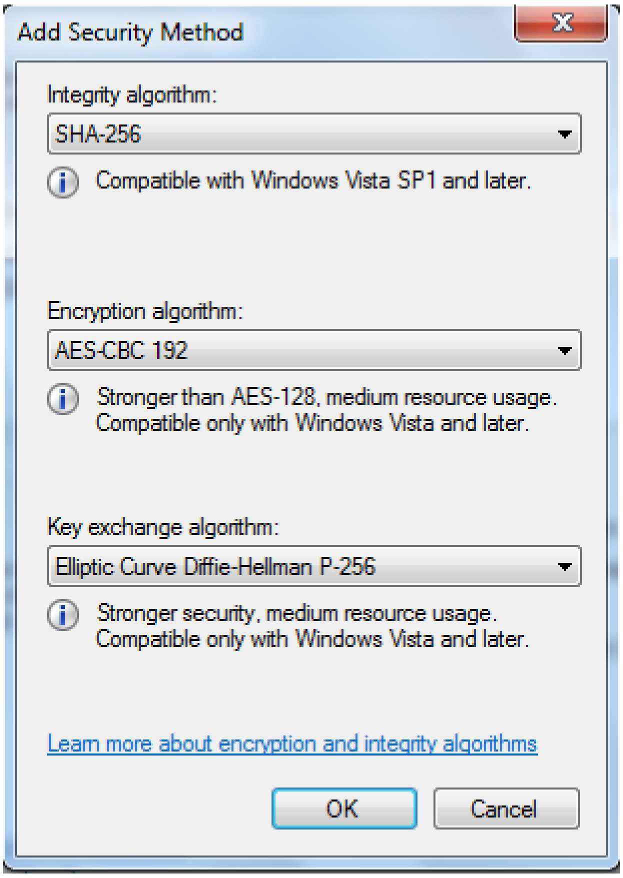


Figure 60 - New security method

After removing the default methods and adding the recommended method, the "Customize Advanced Key Exchange Settings" window should look like this:

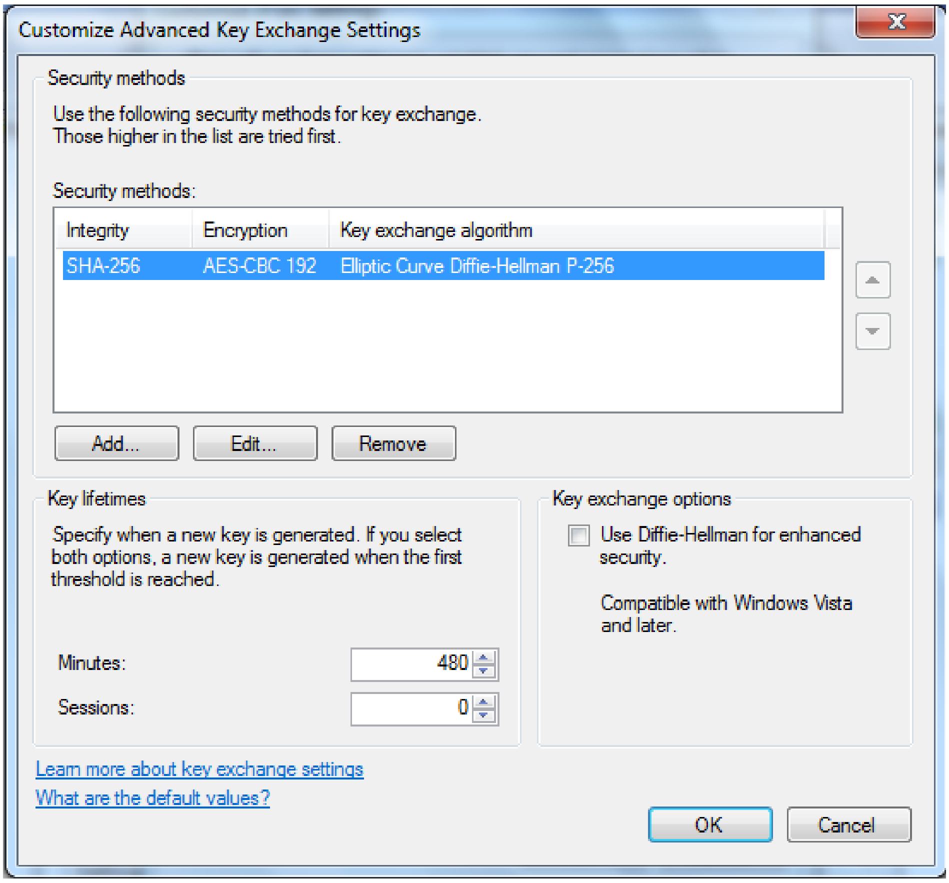


Figure 61 - New security method defined

Finally, mark the "Use Diffie-Hellman for enhanced security" check box in the Key exchange options. With this final change, the window should look like this:

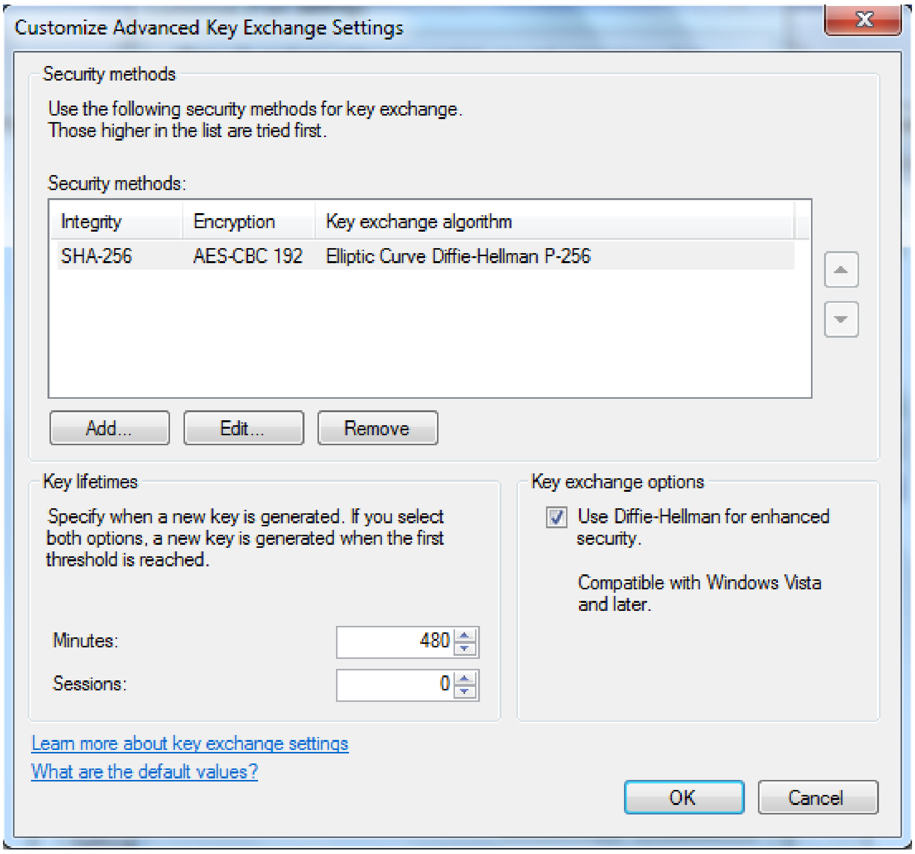


Figure 62 - Advanced Key Exchange Settings ready to be used

Click OK to close the "Customize Advanced Key Exchange Settings" window saving the changes.

Next, in the "Customize IPsec Settings" window, select the option "Advanced" in the "Data protection (Quick Mode)" section, and click its "Customize..." button. The following window will appear:

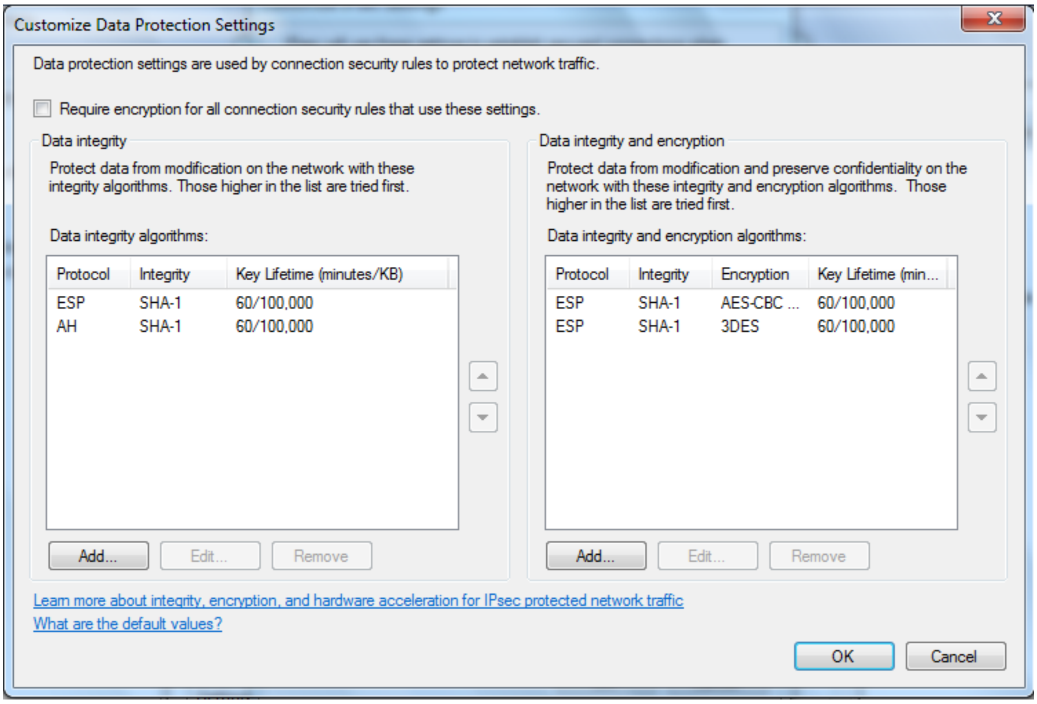


Figure 63 - Data Protection Settings, with the default configuration

Remove all algorithms from both sections ("Data integrity" and "Data integrity and encryption"), and then mark the option "Require encryption for all connection security rules that use these settings":

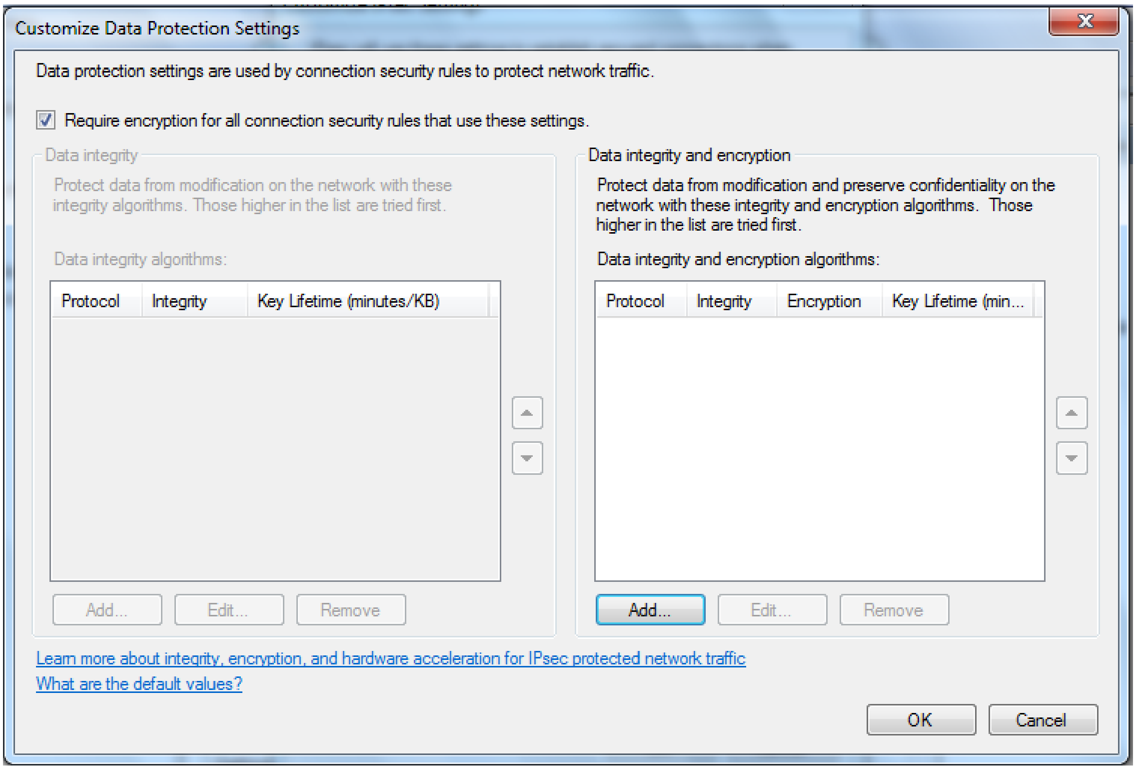


Figure 64 - Data Protection Settings, cleared

Then, click the "Add..." button of the "Data integrity and encryption" section, and select

Then, add a single new algorithm in the "Data integrity and encryption" section, with the configuration of your liking. In this example we will use the following options:

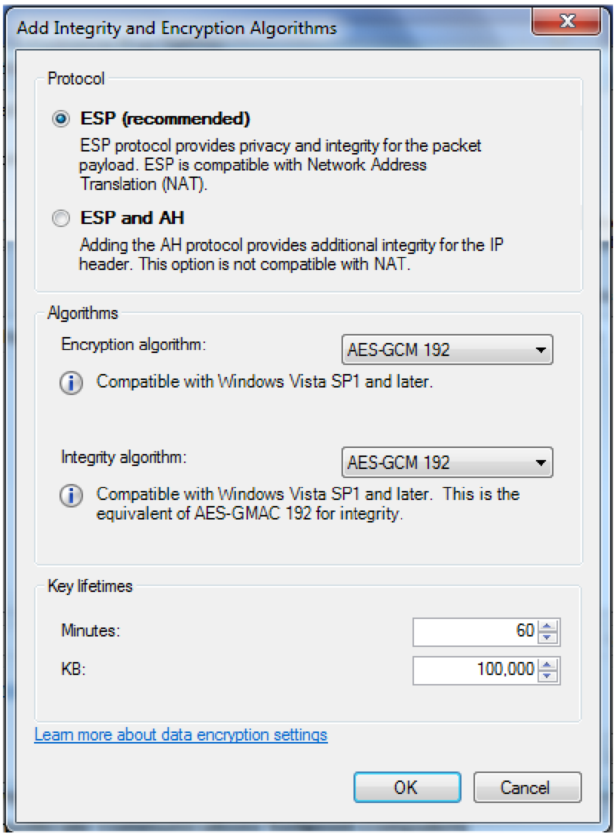


Figure 65 - New integrity and encryption algorithm

The final contents of the "Customize Data Protection Settings" window should be the following:

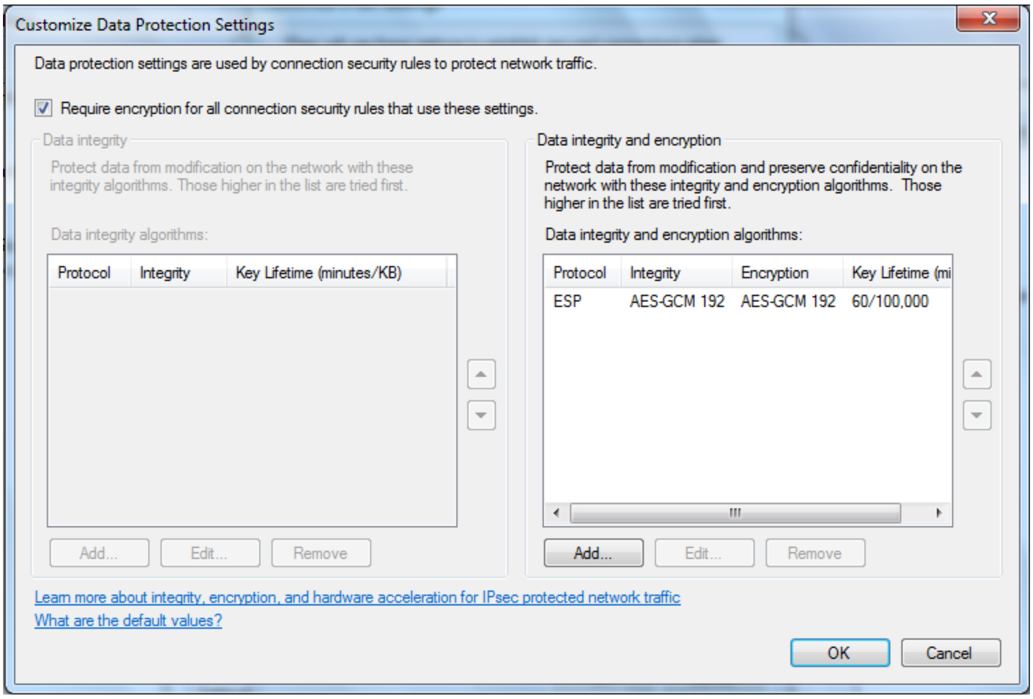


Figure 66 - Data Protection Settings, ready to be used

Click OK to close the "Customize Data Protection Settings" window saving the changes.

Next, in the "Customize IPsec Settings" window, select the option "Advanced" in the "Authentication method" section, and click its "Customize..." button. The following window will appear:

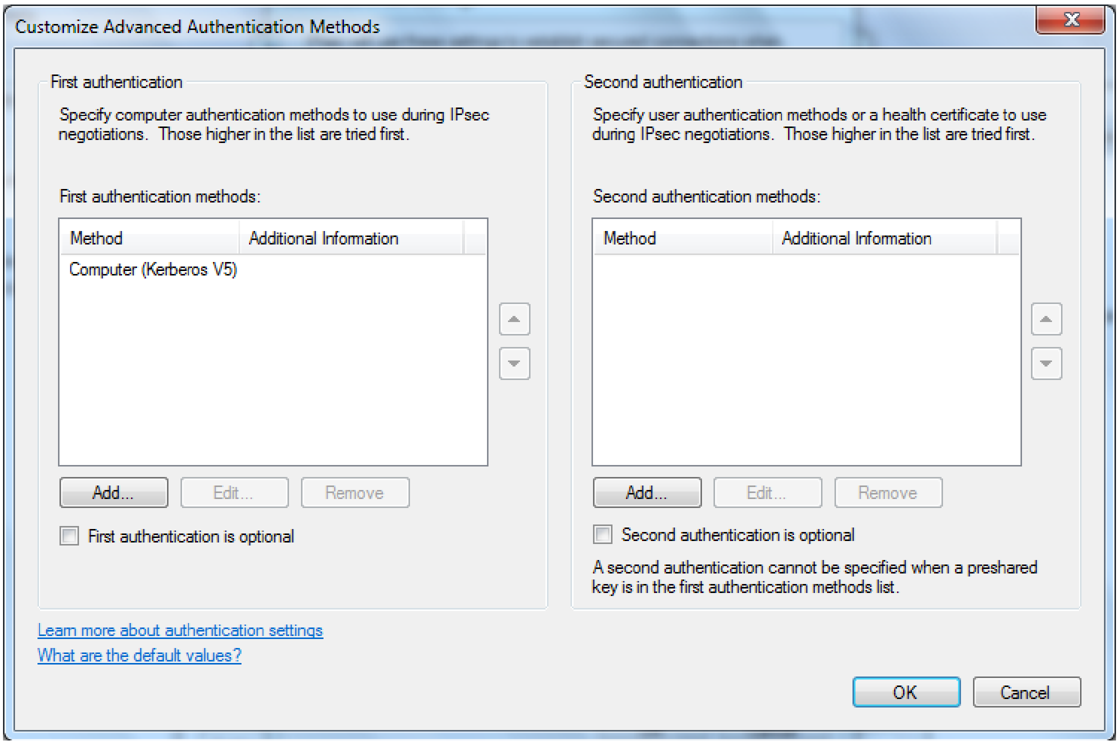


Figure 67 - Advanced Authentication Methods, with the default configuration

Remove all authentication methods from both sections ("First authentication" and "Second authentication"):

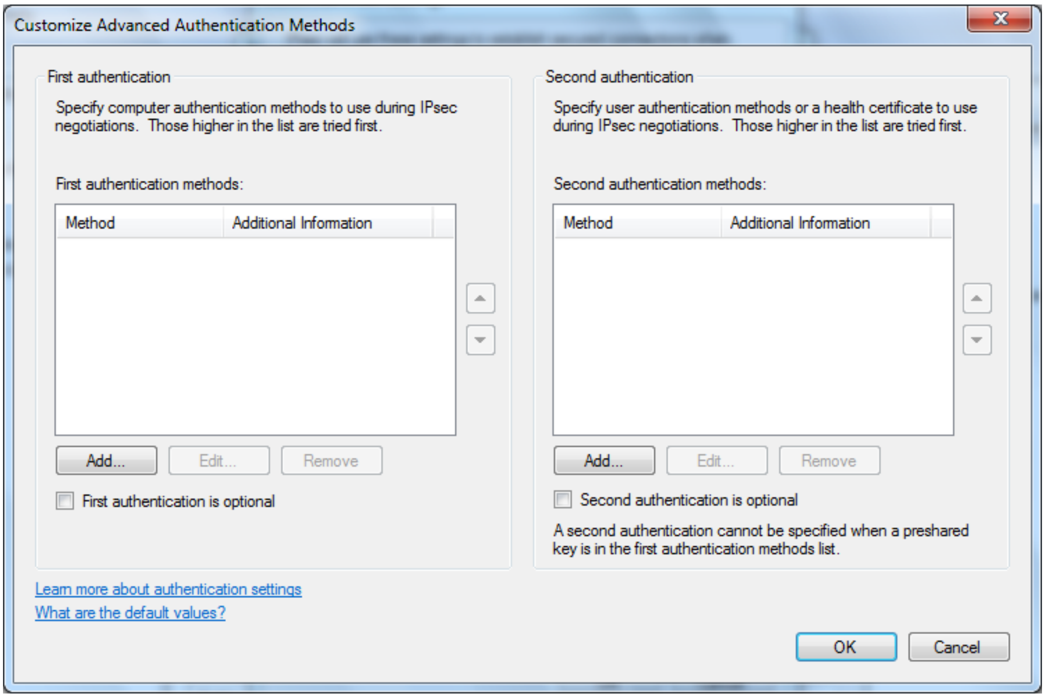


Figure 68 - Advanced Authentication Methods, cleared

Then, click the "Add..." button of the "First authentication" section, and in the popup window that will appear, select the option "Computer certificate from this certification authority (CA)", select "RSA (default)" as the signing algorithm and "Root CA (default)" as the certificate store type, and click the "Browse..." button and select the certificate corresponding to the root CA you used to sign the certificates (in this example, "Example Root CA"). The following figure shows the final contents that the "Add First Authentication Method" should have:

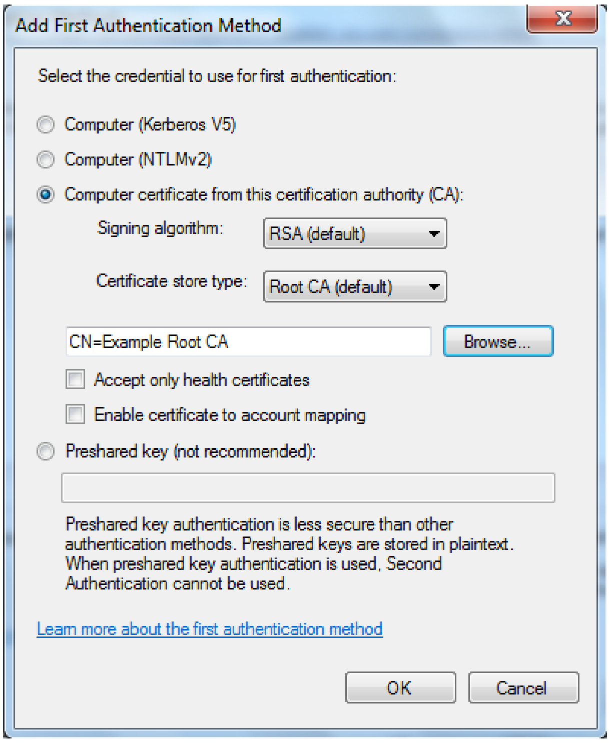


Figure 69 - New First Authentication Method

Click OK to close the "Add First Authentication Method" window saving the results. The "Customize Advanced Authentication Methods" window should now display that, and only that, authentication method based on computer certificates:

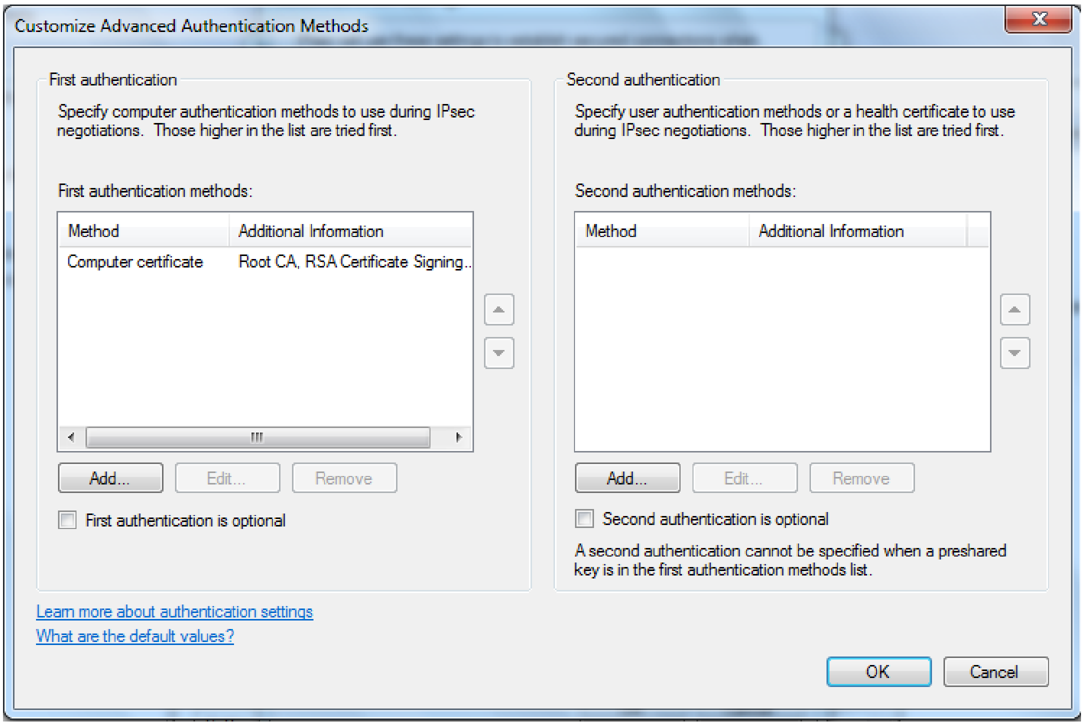


Figure 70 - Advanced Authentication Methods, ready to be used

Click OK to close the "Customize Advanced Authentication Methods" window saving the changes. You will be brought back to the "Customize IPsec Settings", now showing the option Advanced selected in all of its three sections:

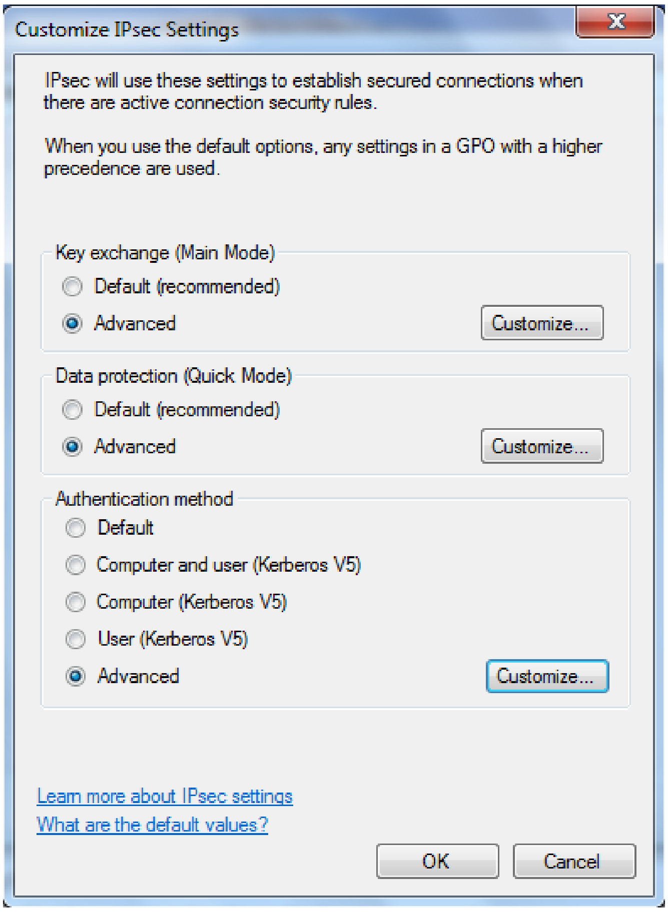


Figure 71 - IPsec Settings, configured

Click OK to close the "Customize IPsec Settings" window saving the changes. You will be brought back to the "IPsec Settings" tab of the "Windows Firewall with Advanced Security on Local Computer Properties" window.

Verify that in the section "IPsec exemptions", the field "Exempt ICMP from IPsec" has the option "No (default)" selected, as shown in the following figure:

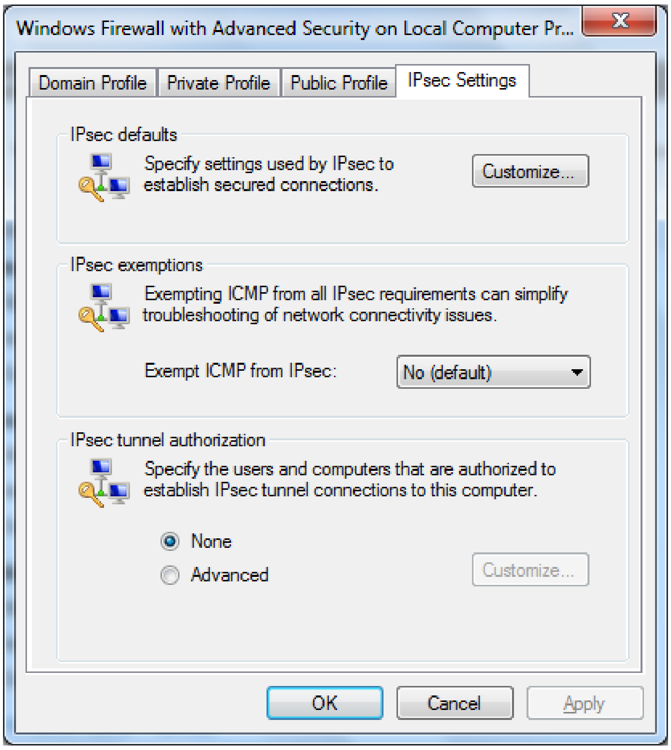


Figure 72 - IPsec Settings, ready to be used (defaults configured, ICMP not exempt from IPsec)

Leave the option "None" selected in the IPsec tunnel authorization.

Finally, click OK to close the "Windows Firewall with Advanced Security on Local Computer Properties" window saving the changes.

3.17 Step 16: On PCALICE, create a connection security rule (IPsec) for traffic exchanged with PCBOB

Still on PCALICE, and still on the "Windows Firewall with Advanced Security" console snap-in, select the container "Connection Security Rules" that hangs from the top level container (NOT the one under the Monitoring container). Initially, the container will be empty:

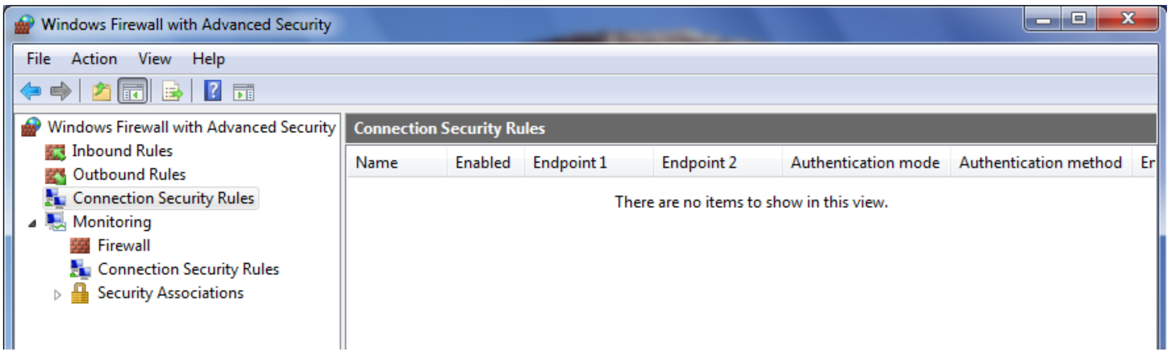


Figure 73 - Connection Security Rules. Initially: none.

Right click on the container and select "New Rule...". This will launch the "New Connection Security Rule Wizard":

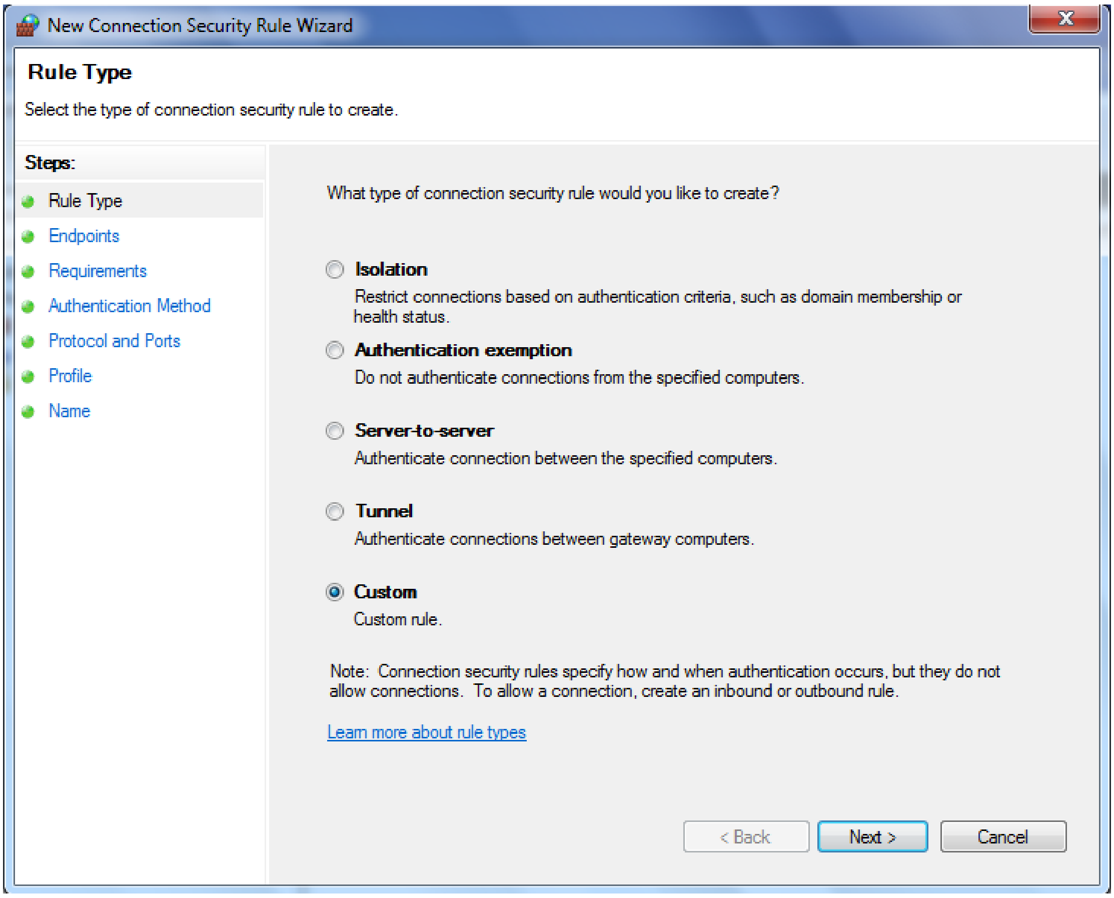


Figure 74 - New Connection Security Rule Wizard. Step: Rule Type.

In the "Rule Type" step (steps are listed on the left column of the window), select the option "Custom" (Custom rule), as shown in the previous figure, and click Next. That will advance the wizard to the next step, "Endpoints".

In the "Endpoints" step, select option "Any IP address" for Endpoint 1, select option "These IP addresses:" for Endpoint 2, and enter the IP address of PCBOB (192.168.108.20) in the text box of that option using the "Add..." button and the option "This IP address or subnet" on the popup window that will appear. The final contents of this step must be the following:

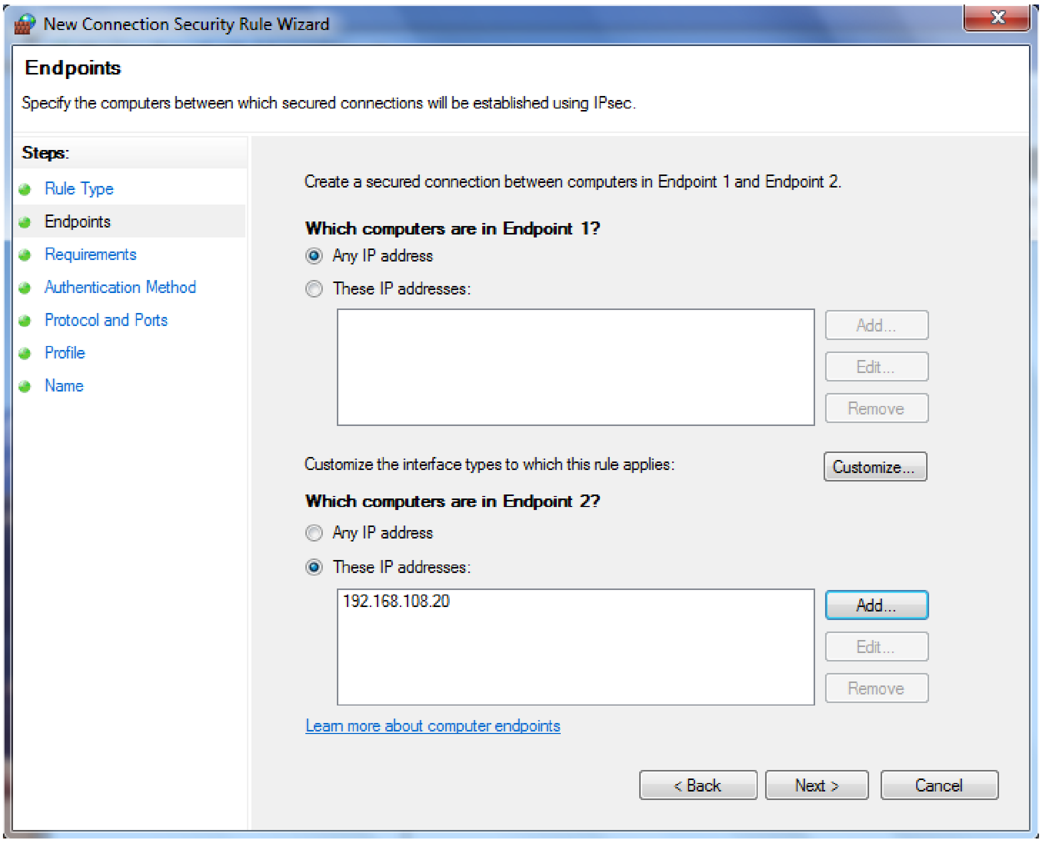


Figure 75 - New Connection Security Rule Wizard. Step: Endpoints

Click Next.

In the "Requirements" step, select the option "Require authentication for inbound and outbound connections":

Note: Make sure you select the option that begins with "Require..." and not with "Request..."

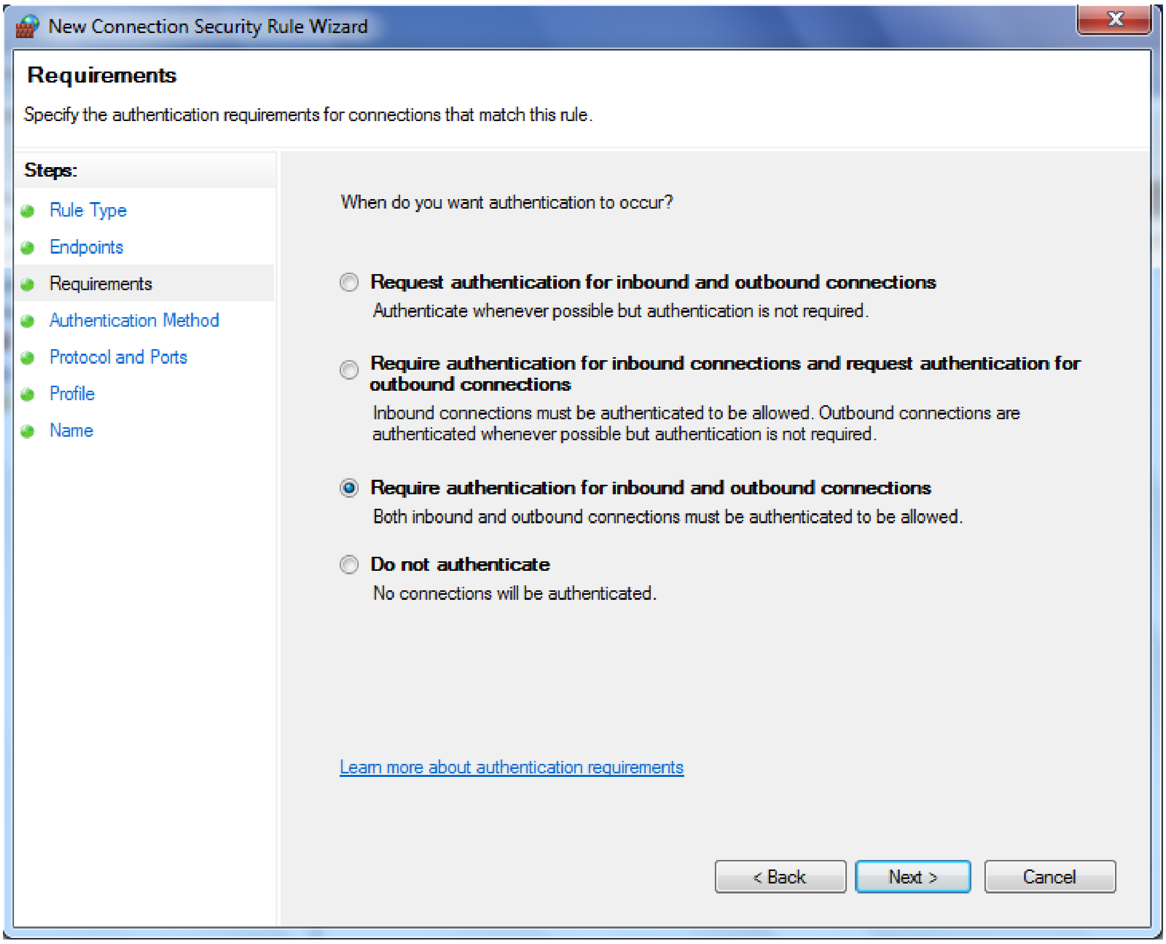


Figure 76 - New Connection Security Rule Wizard. Step: Requirements

Click "Next".

In the "Authentication Method" step, select the option "Default". This will make this particular rule use the authentication methods defined before in the IPsec settings:

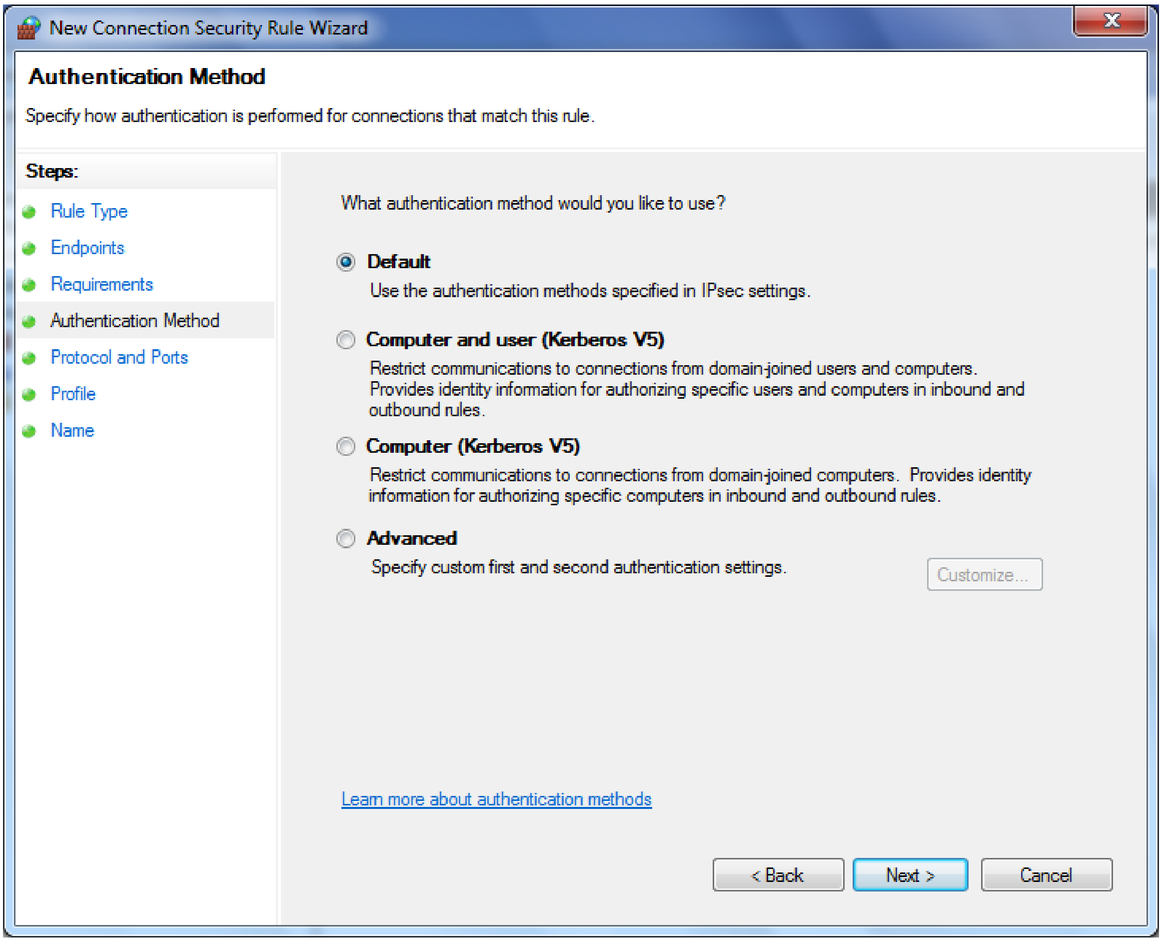


Figure 77 - New Connection Security Rule Wizard. Step: Authentication Method

Click "Next".

In the "Protocols and Ports", select the IP protocols (and ports, if applicable) to which you want this rule to apply. In this example, we will make the rule apply to all IP traffic, selecting the option "Any" in the "Protocol Type" field:

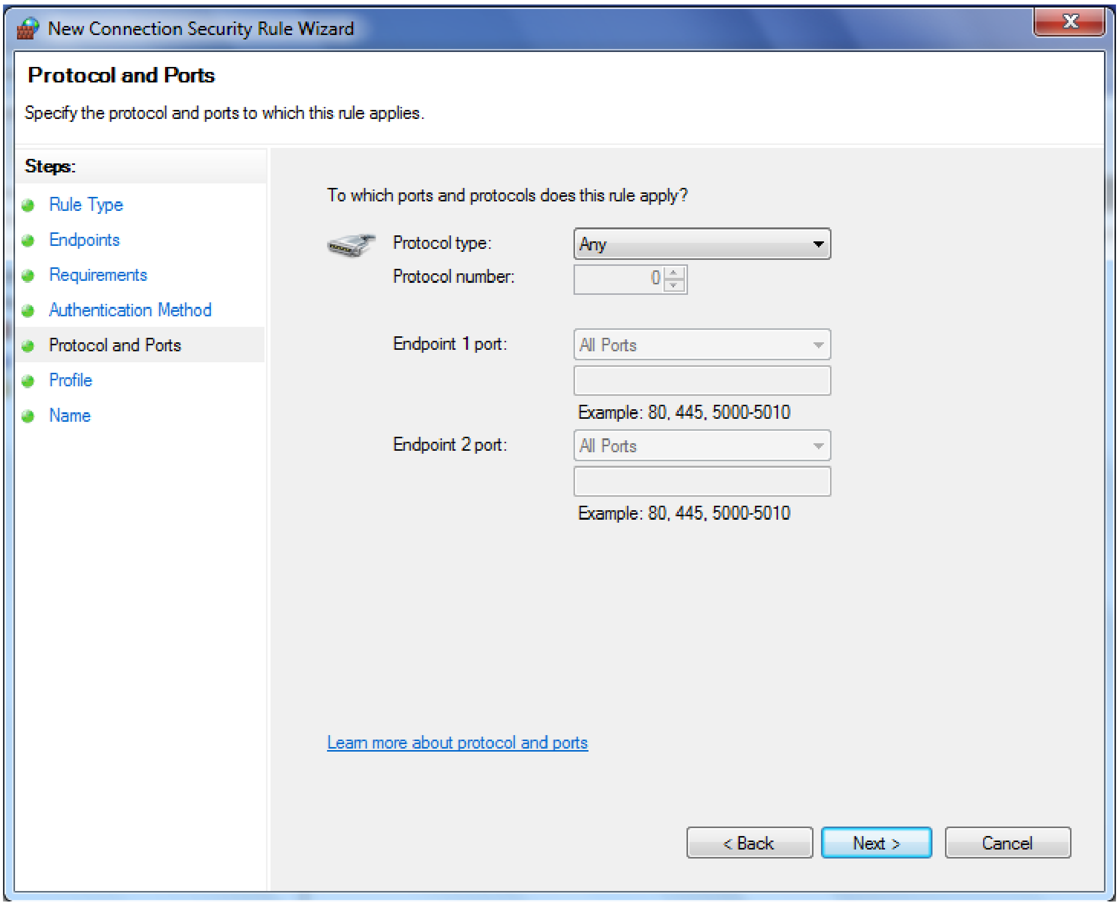


Figure 78 - New Connection Security Rule Wizard. Step: Protocols and Ports

Click "Next".

In the "Profile step", select the network profiles you want this rule to apply to. In this example we will make the rule apply to all three profiles by selecting all of them (Domain, Private and Public):

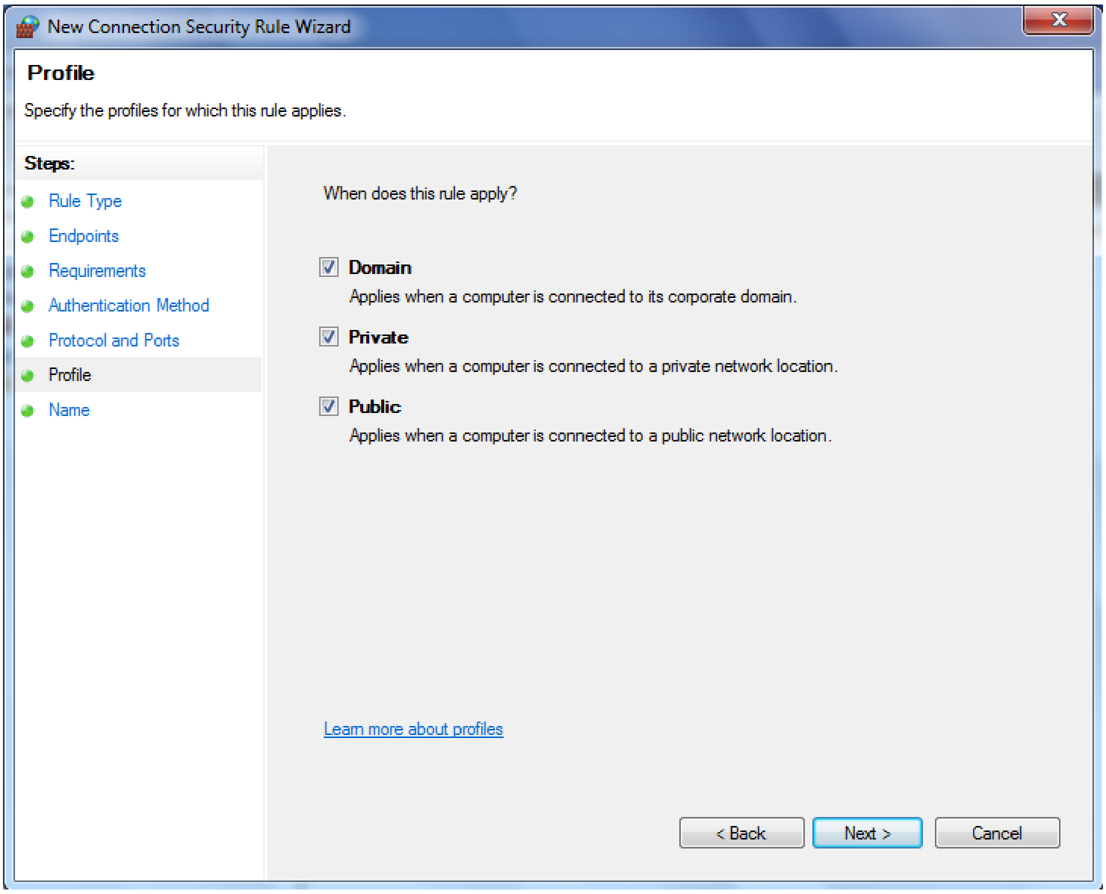


Figure 79 - New Connection Security Rule Wizard. Step: Profile

Click "Next".

In the "Name" step, give the rule a name. In this example we will name it "PCBOB - All traffic":

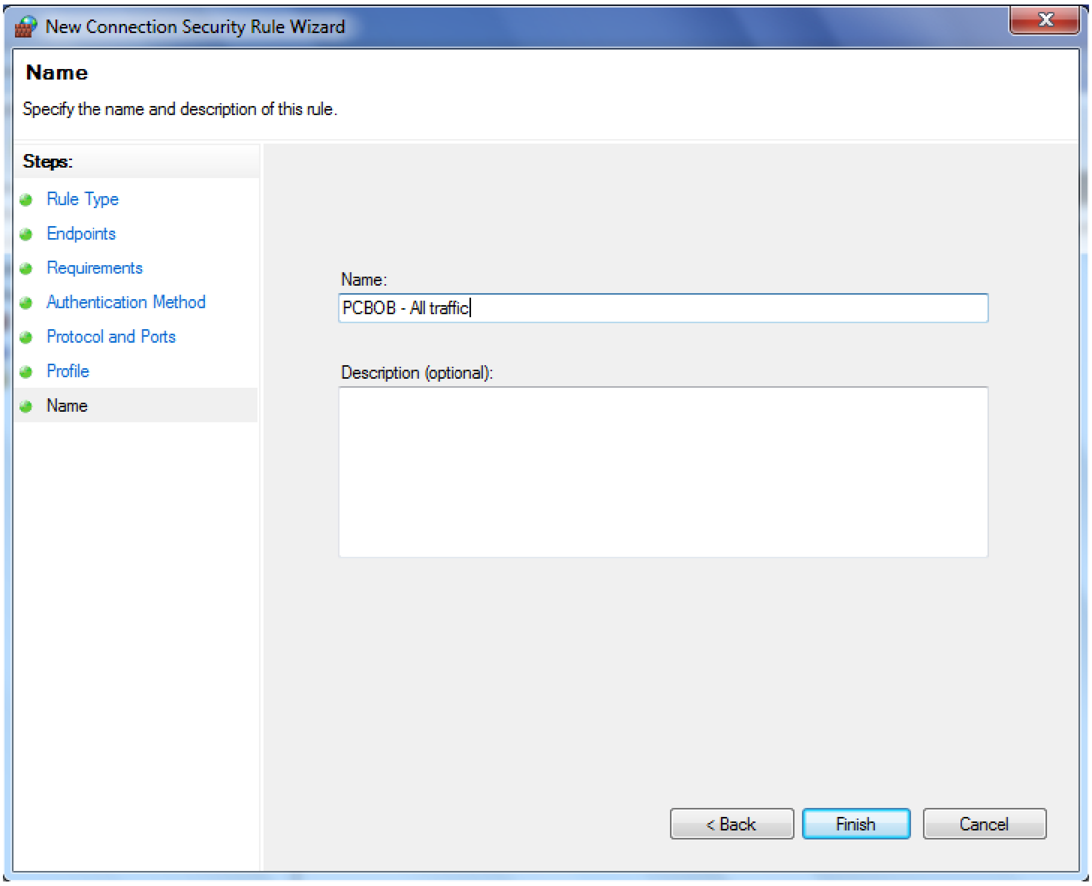


Figure 80 - New Connection Security Rule Wizard. Step: Name

Click "Next". The wizard will close and the rule will appear in the "Connection Security Rules" container:

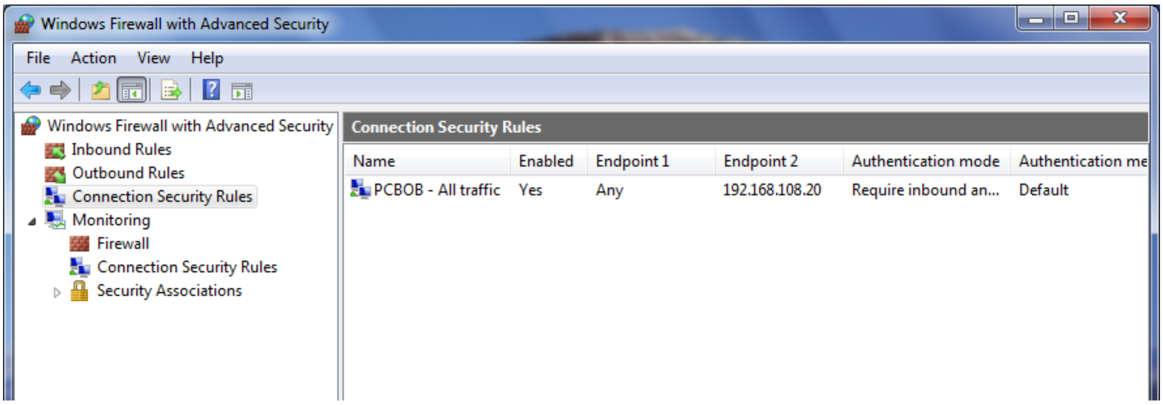


Figure 81 - Connection Security Rules. Rule configured to protect all traffic exchanged with PCBOB.

At this point, PCALICE is ready to protect with IPsec all traffic exchanged with PCBOB. Note, however, that PCBOB is not ready yet, and therefore, if you try to ping PCBOB from PCALICE at this point, the pings will fail trying to negotiate an IPsec security association using the ISAKMP protocol, as shown in the following figure:

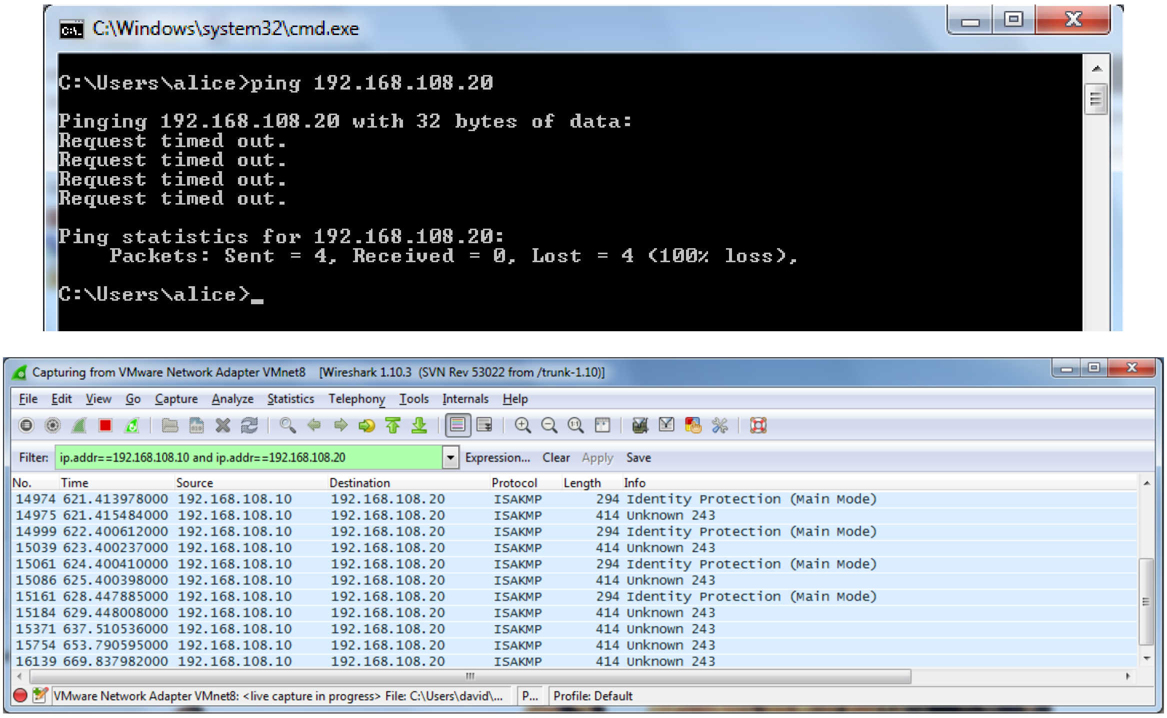


Figure 82 - Ping fails because PCBOB is not ready yet to use IPsec

3.18 Step 17: On PCBOB, mirror steps 14 to 16

On PCBOB, mirror the steps 14 to 16 that you just performed on PCALICE, but swapping any references to PCALICE and PCBOB, and using the IP address of PCALICE as the remote endpoint when configuring the connection security rule on PCBOB.

This will get PCBOB ready to exchange IPsec traffic with PCALICE, and that is what you will verify in the next step.

3.19 Final step: Exchange traffic between PCALICE and PCBOB and verify it gets protected with IPsec

At this point, PCALICE and PCBOB are completely configured to protect with IPsec any IP traffic that they exchange.

In order to verify this, please repeat the network connectivity tests you performed on step 3 (pinging each other while monitoring the network traffic with Wireshark), and check that the PINGs do get responses, and that the traffic is indeed encrypted using IPsec, looking at the output from Wireshark. In the network trace you should see a few ISAKMP packets, while the parameters of the secure connection are being negotiated, and then ESP packets, that are the actual data packets encrypted.

As an example of the output you should see if everything is working fine, the following figure shows a ping from PCALICE to PCBOB, as seen in the CMD console and in Wireshark:

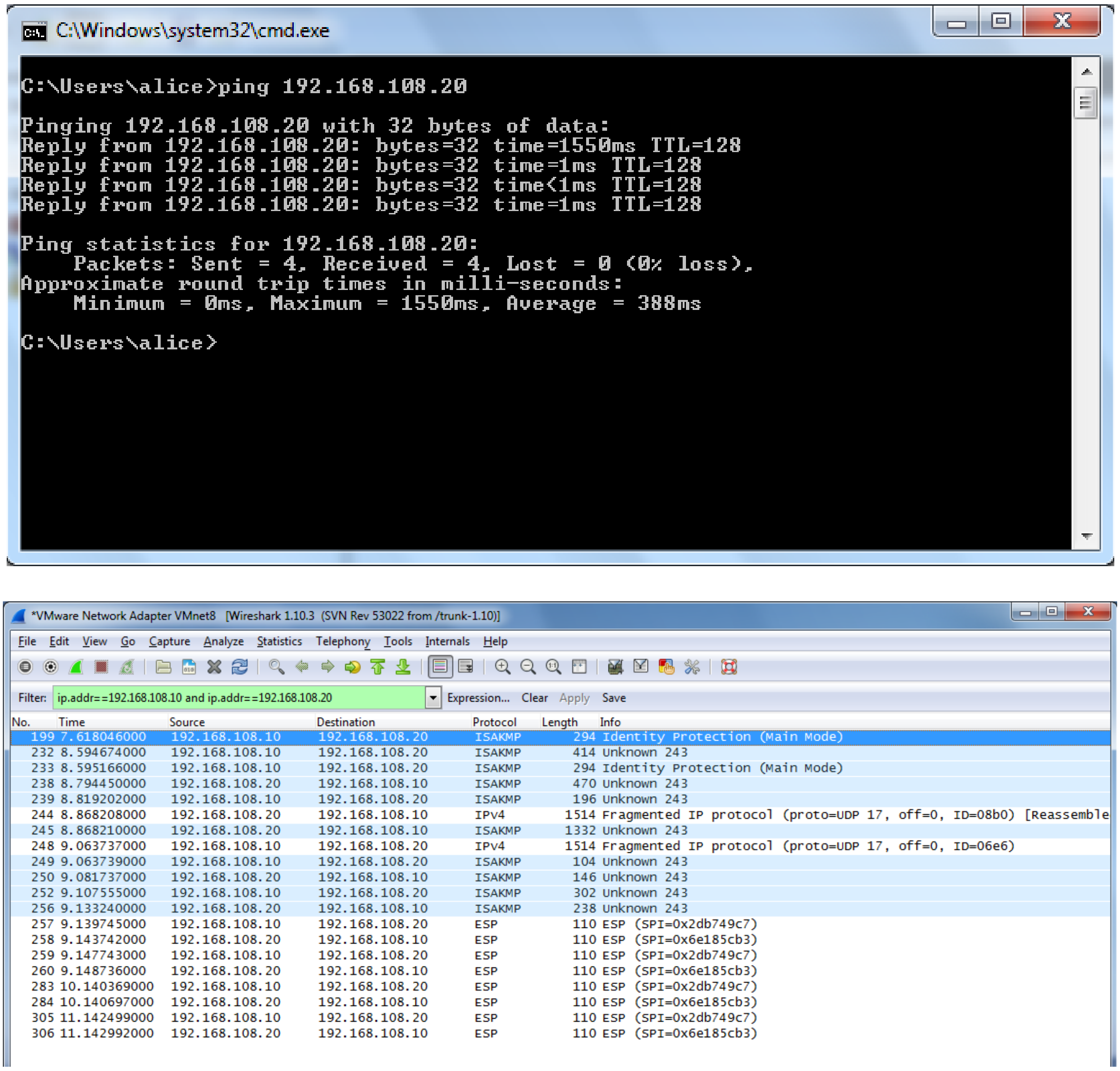


Figure 83 - Ping from PCALICE to PCBOB, protected by IPsec

Please note that the ISAKMP negotiation will only take place the first time these two systems exchange traffic (and every time a certain timeout expires), in order to establish a security association between them. After the security association gets created, traffic will be protected using ESP as defined in such security association, with no need to re-negotiate it each time.

You can see the security associations that are active at any given time in a system by expanding the sub-containers "Main Mode" and "Quick Mode" under the container "Monitoring -> Security Associations", in the "Windows Firewall with Advanced Security" console snap-in:

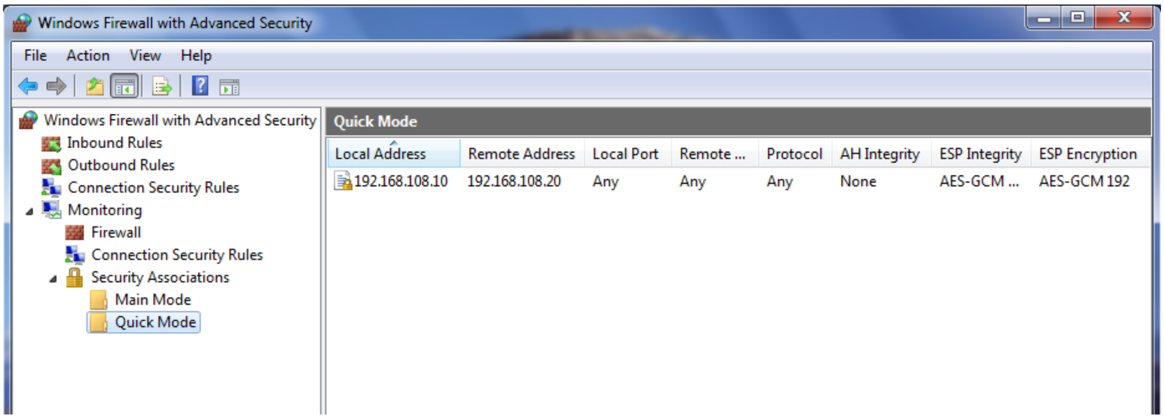


Figure 84 - Security associations monitoring

4 Step-by-step guide #2: Generate all CSRs centrally and without desired extensions

This section constitutes another step by step guide to generate, install and use digital certificates to communicate via IPsec two standalone Windows 7 systems, using easy-rsa to create the certificates. The start, goals and many of the steps will be common to the previous guide. However, the method to generate the CSRs and the certificates will be different. All CSRs will be generated in the Linux host and they will not contain information about any desired extensions. Then, the CSRs will be signed using easy-rsa to add the appropriate extensions to the corresponding certificates, and then the certificates, including their corresponding private keys, will be distributed to the Windows systems.

Some steps are equal to steps of the previous guide. In those cases, a simple reference will be indicated, rather than reprinting their contents.

4.1 Step 0: Starting point and goal

See section *3.1, Step 0: Starting point and goal*.

4.2 Step 1: Configure the Windows firewall of PCALICE to allow incoming traffic from PCBOB

See section *3.2, Step 1: Configure the Windows firewall of PCALICE to allow incoming traffic from PCBOB.*

4.3 Step 2: Configure the Windows firewall of PCBOB to allow incoming traffic from PCALICE

See section *3.3, Step 2: Configure the Windows firewall of PCBOB to allow incoming traffic from PCALICE.*

4.4 Step 3: Test the network connectivity between PCALICE and PCBOB

See section *3.4, Step 3: Test the network connectivity between PCALICE and PCBOB*.

4.5 Step 4: Install easy-rsa prerequisites on "linuxhost"

See section 3.7, Step 6: Install easy-rsa prerequisites on "linuxhost"

4.6 Step 5: Configure easy-rsa on "linuxhost" to generate certificates that will be valid for Windows IPsec

As previously stated, extensions in a certificate are fields that, among other things, may declare the uses for which such certificate should be trusted. Examples are "Server authentication", "IP security end system" or "Code Signing".

Each extension is identified by an object identifier (OID) and/or its associated name (if there is one). The contents of an extensions may be simple text, or other OIDs.

For a certificate to be valid for Windows IPsec authentication, it needs to include the following intended uses:

- Server Authentication (OID: 1.3.6.1.5.5.7.3.1)
- Client Authentication (OID: 1.3.6.1.5.5.7.3.2)
- IP security end system (OID: 1.3.6.1.5.5.7.3.5)

Knowing this, one might feel tempted to include those uses in the well known "Enhanced Key Usage" extension and be done with it. However, that would not be enough: Windows needs those uses declared in a different extension, "Application Policies", regardless of whether they are also declared in "Enhanced Key Usage" or not.

Note: When a CSR is generated in Windows including those intended uses, the CSR includes them in both extensions, "Enhanced Key Usage" and "Application Policies", but a certificate containing only the "Application Policies" extension is also acceptable to Windows.

Easy-rsa, in its default configuration, does not contain a certificate type definition that includes the above extensions and intended uses. But, fortunately, defining a new type of certificate in easy-rsa is pretty simple. All you have to do is creating a new file in the subdirectory "x509-types" of easy-rsa (easy-rsa/easyrsa3/x509-types/), naming it with the same name you want easy-rsa to use to refer to the new type of certificate. For example, you may create a file with the name "WindowsIPsecComputer". The contents of the file must be the extensions that you want easy-rsa to add to all certificates generated using this type definition, overriding the values for those contained in the CSR, if present.

The extension "Enhanced Key Usage" is a simple comma separated list of OIDs (or their corresponding names) that indicate the intended uses of the certificate. Therefore, the definition of this extension that will need to be included in the new type file is pretty straightforward:

```
extendedKeyUsage=1.3.6.1.5.5.7.3.2, 1.3.6.1.5.5.7.3.2, 1.3.6.1.5.5.7.3.5
```

The format of the extension "Application Policies" (OID 1.3.6.1.4.1.311.21.10), however, is a bit more complex, because it must be defined as a list of lists (even though the inner lists may be composed of just one element, as in this case). The syntax to use in this case is:

```
1.3.6.1.4.1.311.21.10=ASN1:SEQUENCE:application_policies_section

[ application_policies_section ]
dummylabel01 = SEQ:intended_use_01
dummylabel02 = SEQ:intended_use_02
```

```
dummylabel03 = SEQ:intended_use_03

[ intended_use_01 ]
# Server Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.1

[ intended_use_02 ]
# Client Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.2

[ intended_use_03 ]
# IP security end system
dummylabel01 = OID:1.3.6.1.5.5.7.3.5
```

Apart from these two extensions, you will probably want to include the following, to make sure that the certificates cannot be used to sign other certificates, and to include in the certificates some common identification information:

```
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
```

Putting it all together, what you need to do is create a file named "WindowsIPsecComputer" in the "easy-rsa/easyrsa3/x509-types/" folder, with the following contents:

```
# X509 extensions for a WindowsIPsecComputer certificate
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage=1.3.6.1.5.5.7.3.1, 1.3.6.1.5.5.7.3.2, 1.3.6.1.5.5.7.3.5
1.3.6.1.4.1.311.21.10=ASN1:SEQUENCE:application_policies_section

[ application_policies_section ]
dummylabel01 = SEQ:intended_use_01
dummylabel02 = SEQ:intended_use_02
dummylabel03 = SEQ:intended_use_03

[ intended_use_01 ]
# Server Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.1

[ intended_use_02 ]
# Client Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.2

[ intended_use_03 ]
# IP security end system
dummylabel01 = OID:1.3.6.1.5.5.7.3.5
```

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3/x509-types
File  Edit  View  Search  Terminal  Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3/x509-types$ ls
ca  client  COMMON  freeformat  server  WindowsIPsecComputer
user1@linuxhost:~/sw/easy-rsa/easyrsa3/x509-types$ cat WindowsIPsecComputer
# X509 extensions for a WindowsIPsecComputer certificate
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage=1.3.6.1.5.5.7.3.1, 1.3.6.1.5.5.7.3.2, 1.3.6.1.5.5.7.3.5
1.3.6.1.4.1.311.21.10=ASN1:SEQUENCE:application_policies_section

[ application_policies_section ]
dummylabel01 = SEQ:intended_use_01
dummylabel02 = SEQ:intended_use_02
dummylabel03 = SEQ:intended_use_03

[ intended_use_01 ]
# Server Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.1

[ intended_use_02 ]
# Client Authentication
dummylabel01 = OID:1.3.6.1.5.5.7.3.2

[ intended_use_03 ]
# IP security end system
dummylabel01 = OID:1.3.6.1.5.5.7.3.5
user1@linuxhost:~/sw/easy-rsa/easyrsa3/x509-types$
```

Figure 85 - Creation of file easy-rsa/easyrsa3/x509-types/WindowsIPsecComputer

Note: If you want to save these edits using git, you may do so by executing:

```
cd $HOME/sw/easy-rsa/
git add easyrsa3/x509-types/WindowsIPsecComputer
git commit -m "Add WindowsIPsecComputer certificate type"
```

Note: Do not worry if your instance of easy-rsa does not include the file "freeformat" that appears in the previous figure, because it is simply a file that would get created if you followed the previous step by step guide, as described in section 3.9. It does not matter if it is present or not for this step by step guide.

Note: If you followed section 3.9 and therefore added the variable "copy_extensions" to file openssl-1.0.cnf, you may want to comment it out now, because it is not needed for this method, and it may pose a security risk in the future if you forget about it.

4.7 Step 6: Create your own root CA using easy-rsa

See section 3.10, Step 9: Create your own root CA using easy-rsa. Only do this differently:

- Give the CA the name "Example2 Root CA" as its Common Name

4.8 Step 7: Generate CSRs (and private keys) for PCALICE and PCBOB using easy-rsa on "linuxhost"

Execute the following commands to create a CSR for PCALICE:

```
cd $HOME/sw/easy-rsa/easyrsa
./easyrsa gen-req PCALICE
```

Easy-rsa will first generate a private/public key pair, and ask you to select a pass phrase to protect the private key. Choose a pass phrase of your liking.

Then you will be asked to enter a Common Name for the CSR, and be offered the name you used when invoked the command, as the default value. Accept the default value (PCALICE).

Easy-rsa will then inform you of the files it just created, which will be PCALICE.key, containing the private key for PCALICE, and PCALICE.req, containing the CSR for PCALICE (which includes its public key), and exit.

At this point, the private key and CSR of PCALICE are ready to be used.

The following figure shows a screenshot of the process just described.

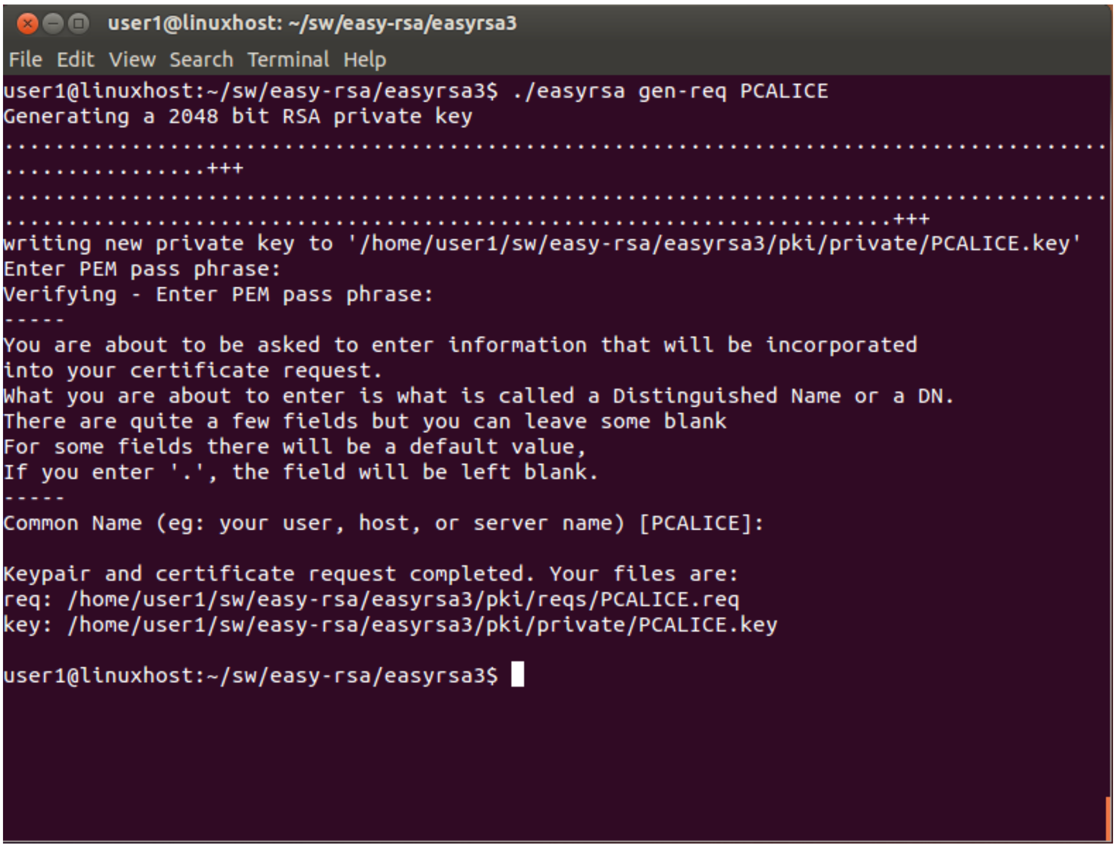


Figure 86 - Generation of the CSR (and private key) for PCALICE

Then, repeat the same process, but using the name PCBOB instead of PCALICE:

```
cd $HOME/sw/easy-rsa/easyrsa
./easyrsa gen-req PCBOB
```

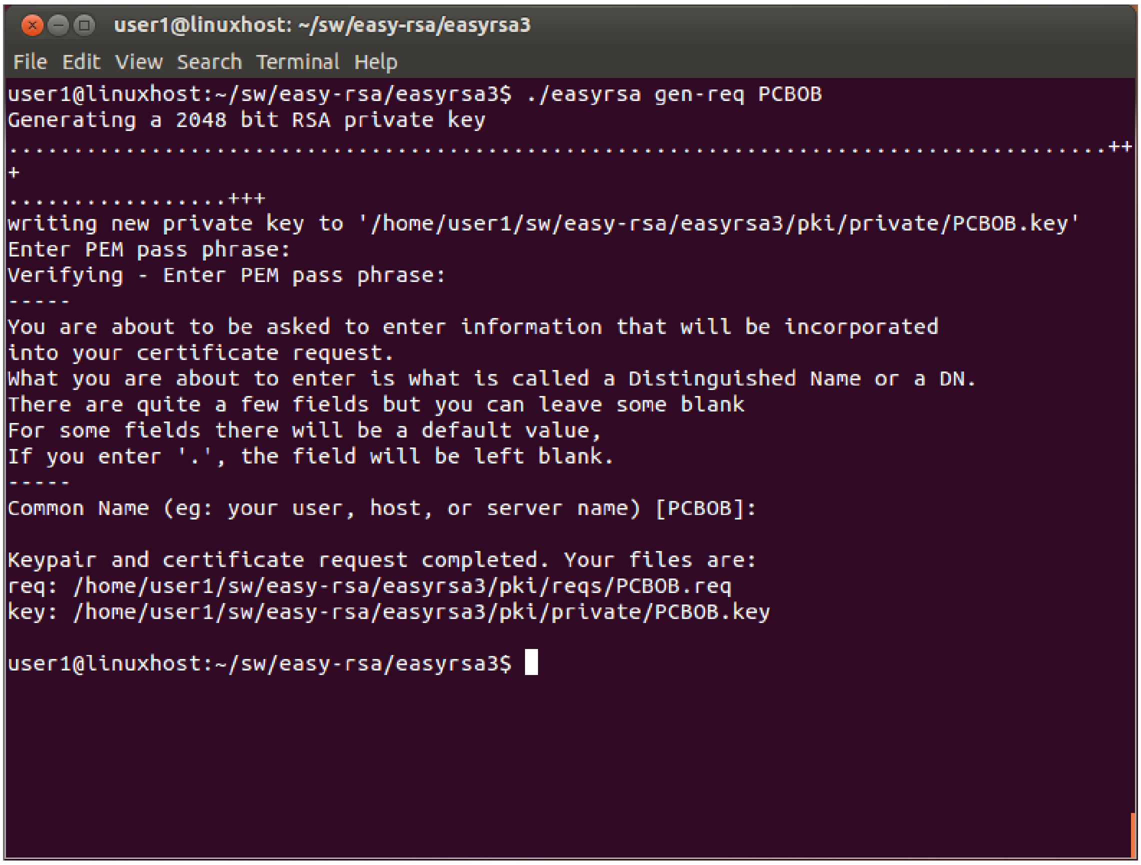


Figure 87 - Generation of the CSR (and private key) for PCBOB

4.9 Step 8: Generate the certificates by signing the CSRs using easy-rsa on "linuxhost"

Execute the following command to sign the CSR of PCALICE generating a certificate of the type "WindowsIPsecComputer" that you defined earlier:

```
./easyrsa sign-req WindowsIPsecComputer PCALICE
```

Easy-rsa will display the Common Name contained in the CSR and ask you to confirm that you do want to sign that CSR. Type "yes" (without the quotes) to confirm.

Then, easy-rsa will ask you to enter the pass phrase that protects the private key of the CA (ca.key). You will need to type the pass phrase that you selected when you created the CA.

Finally, easy-rsa will create the certificate and exit after informing you of the location of the newly issued certificate:

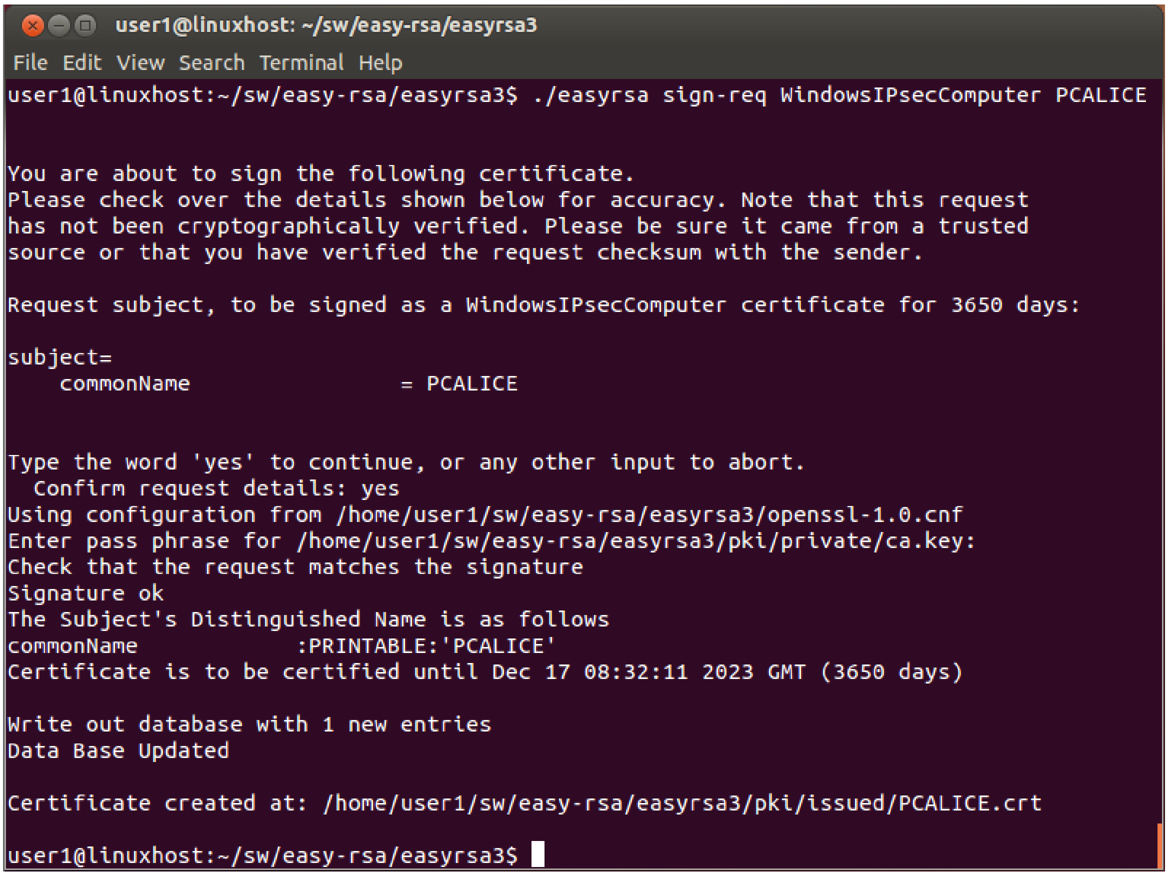
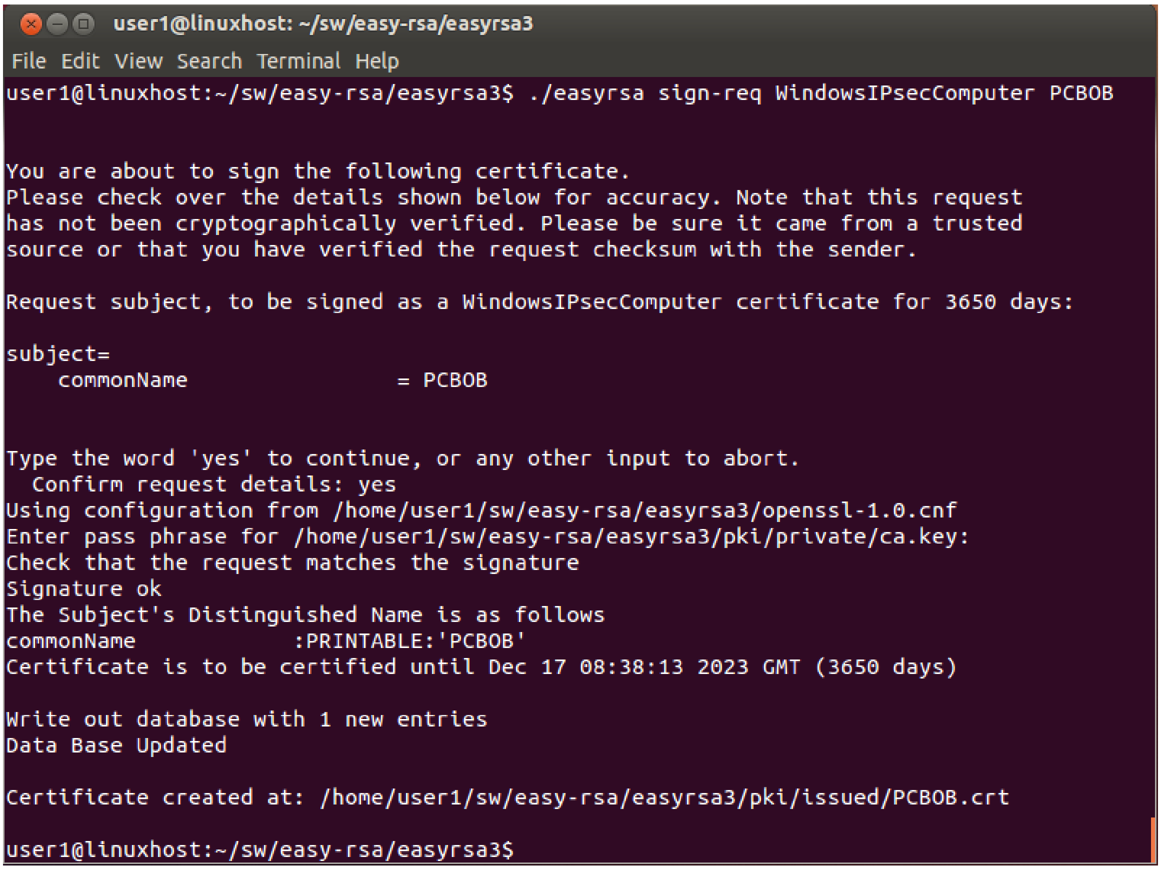


Figure 88 - Signing PCALICE's CSR, thus generating its certificate

Then, repeat the same process, but using the name PCBOB instead of PCALICE:

```
./easyrsa sign-req WindowsIPsecComputer PCBOB
```

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File Edit View Search Terminal Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa sign-req WindowsIPsecComputer PCBOB


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a WindowsIPsecComputer certificate for 3650 days:

subject=
    commonName                = PCBOB


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /home/user1/sw/easy-rsa/easyrsa3/openssl-1.0.cnf
Enter pass phrase for /home/user1/sw/easy-rsa/easyrsa3/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :PRINTABLE:'PCBOB'
Certificate is to be certified until Dec 17 08:38:13 2023 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /home/user1/sw/easy-rsa/easyrsa3/pki/issued/PCBOB.crt

user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 89 - Signing PCBOB's CSR, thus generating its certificate

4.10 Step 9: Pack each certificate with its corresponding private key and the certificate of the CA into a PKCS #12 (.p12) file using easy-rsa on "linuxhost"

Execute the following command to pack together in a single file, with PKCS #12 format (.p12 extension), the certificate of PCALICE, its private key, and the certificate of the CA (Example2 Root CA):

```
./easyrsa export-p12 PCALICE
```

Easy-rsa will first ask you to enter the pass phrase that protects the private key of PCALICE (which you selected when you created its CSR).

Then, it will ask you to choose an "Export Password". This is a pass phrase that will protect the private key inside the .p12 file.

Finally, easy-rsa will exit after informing you of the location of the output file:

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File Edit View Search Terminal Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa export-p12 PCALICE
Enter pass phrase for /home/user1/sw/easy-rsa/easyrsa3/pki/private/PCALICE.key:
Enter Export Password:
Verifying - Enter Export Password:

Successful export of p12 file. Your exported file is at the following
location: /home/user1/sw/easy-rsa/easyrsa3/pki/private/PCALICE.p12

user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 90 - Exporting PCALICE's certificate and private key, and the certificate of the CA, into a single .p12 file

Then, repeat the same process, but using the name PCBOB instead of PCALICE:

```
./easyrsa export-p12 PCBOB
```

```
user1@linuxhost: ~/sw/easy-rsa/easyrsa3
File Edit View Search Terminal Help
user1@linuxhost:~/sw/easy-rsa/easyrsa3$ ./easyrsa export-p12 PCBOB
Enter pass phrase for /home/user1/sw/easy-rsa/easyrsa3/pki/private/PCBOB.key:
Enter Export Password:
Verifying - Enter Export Password:

Successful export of p12 file. Your exported file is at the following
location: /home/user1/sw/easy-rsa/easyrsa3/pki/private/PCBOB.p12

user1@linuxhost:~/sw/easy-rsa/easyrsa3$
```

Figure 91 - Exporting PCBOB's certificate and private key, and the certificate of the CA, into a single .p12 file

4.11 Step 10: Copy each PKCS #12 (.p12) file to the corresponding system

Now that you have all the necessary certificates, you need to copy them from linuxhost to the appropriate systems.

Copy the following file from linuxhost to any folder (e.g. C:\tmp\) in PCALICE:

File name	Location in linuxhost	Description
PCALICE.p12	$HOME/sw/easy-rsa/easyrsa3/pki/private/PCALICE.p12	PCALICE's .p12 file

Then, copy the following file from linuxhost to any folder (e.g. C:\tmp\) in PCBOB:

File name	Location in linuxhost	Description
PCBOB.p12	$HOME/sw/easy-rsa/easyrsa3/pki/private/PCBOB.p12	PCBOB's .p12 file

4.12 Step 11: On PCALICE, import the certificates and the private key contained in its PKCS #12 file

On PCALICE, launch an instance of the Microsoft Management Console (mmc.exe) with administrative privileges:

Start -> Type "mmc" -> Right click on the mmc program -> Run as administrator

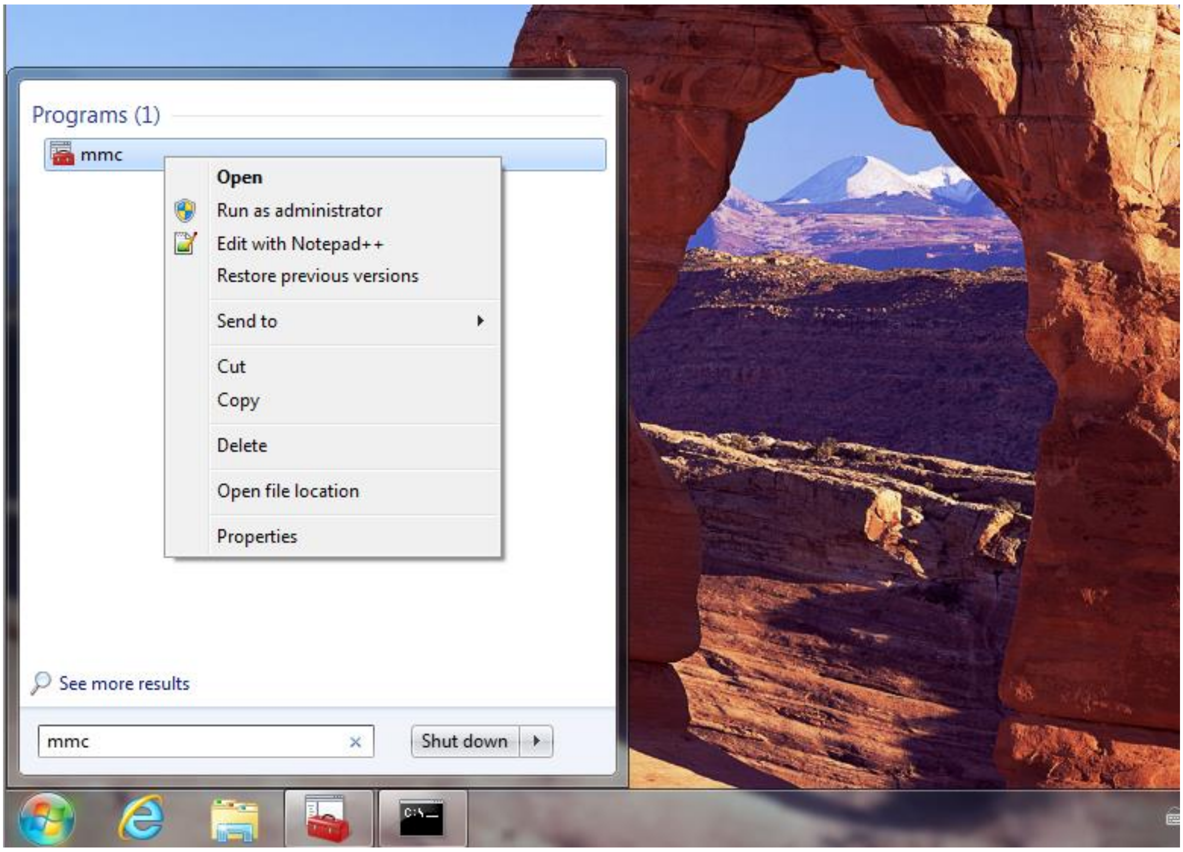


Figure 92 - Launching Microsoft Management Console (mmc) with administrative privileges

The following window should appear:

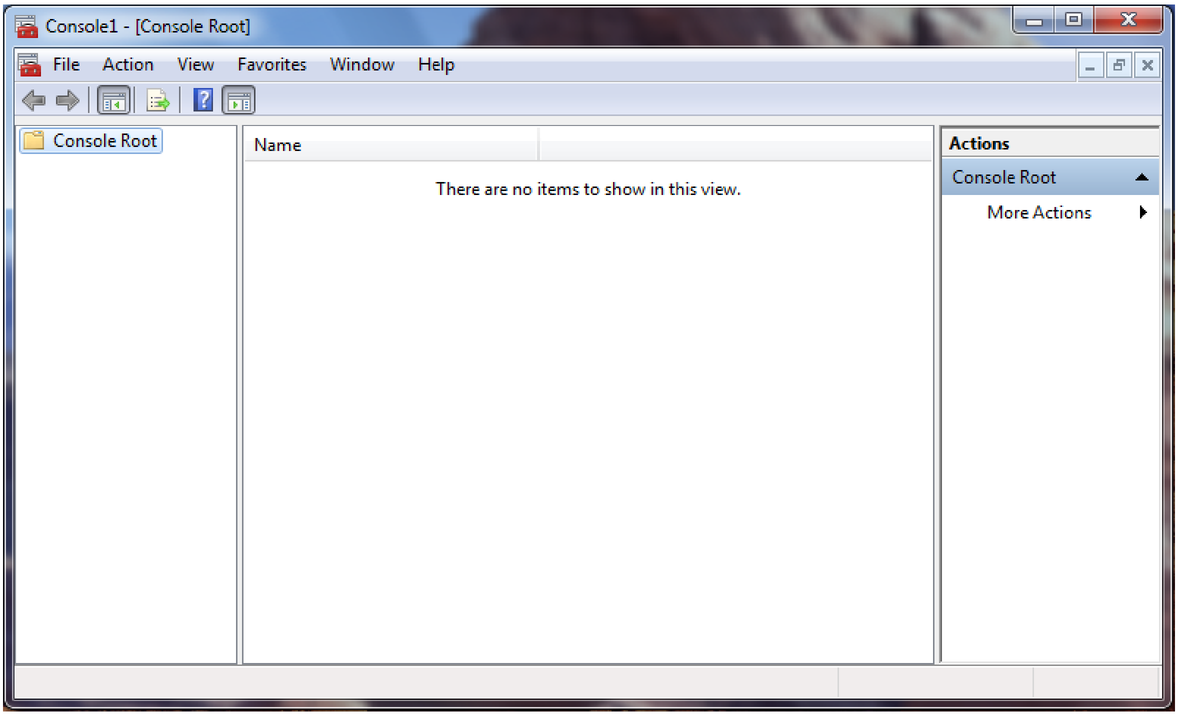


Figure 93 - Microsoft Management Console (mmc)

Note: At this point the window will not show any sign of being running with administrative privileges, but the following steps will not work as expected if you fail to start mmc.exe with administrative privileges.

Then, add the Certificates snap-in under the identity of the local computer, as follows:

File -> Add/Remove Snap-in... -> Select "Certificates" and click "Add >" -> Select "Computer account" in the pop-up window that will appear and then click "Next" in that window -> Select "Local computer: (the computer this console is running on)" and click "Finish" (still in the pop-up window, to return to the "Add or Remove Snap-ins" window).

At this point your Add or Remove Snap-ins window should show "Certificates (Local Computer)" in the "Selected snap-ins:" section:

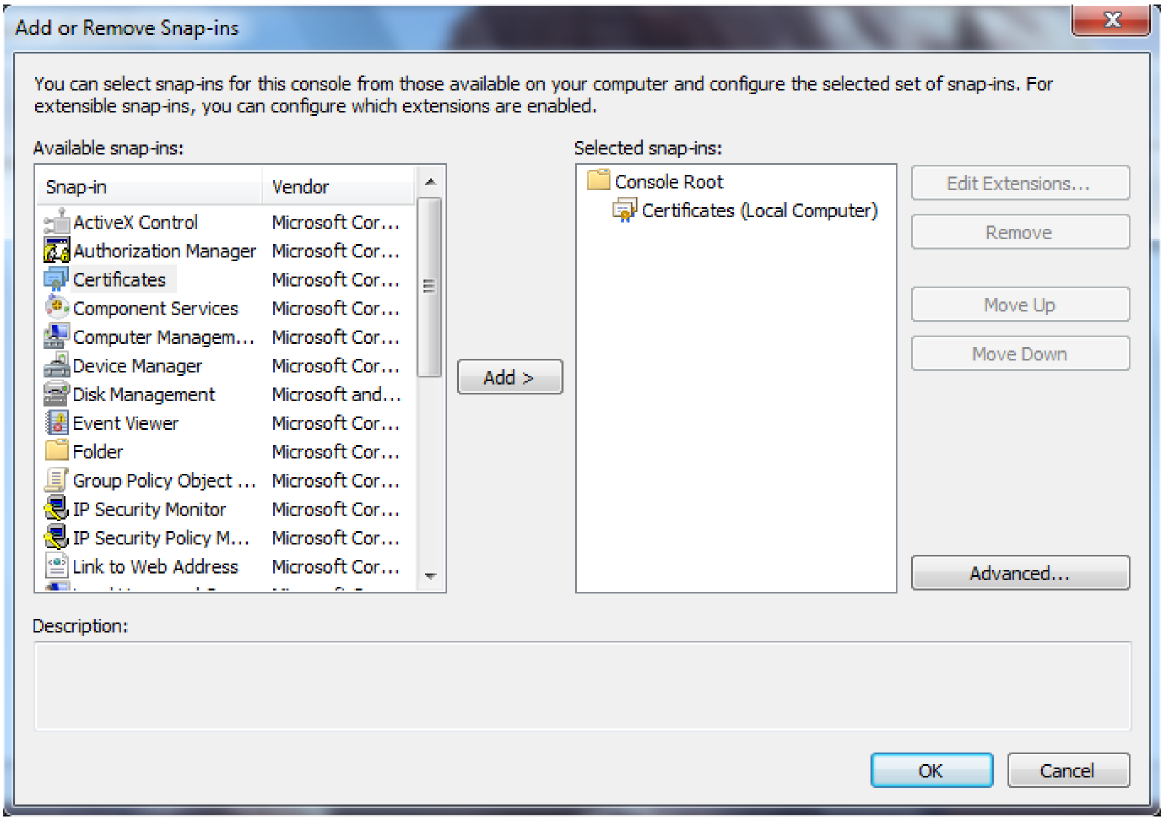


Figure 94 - Certificates (Local Computer) snap-in selected

Click OK to close the Add or Remove Snap-ins window and return to the MMC console, which should show the selected snap-in inside:

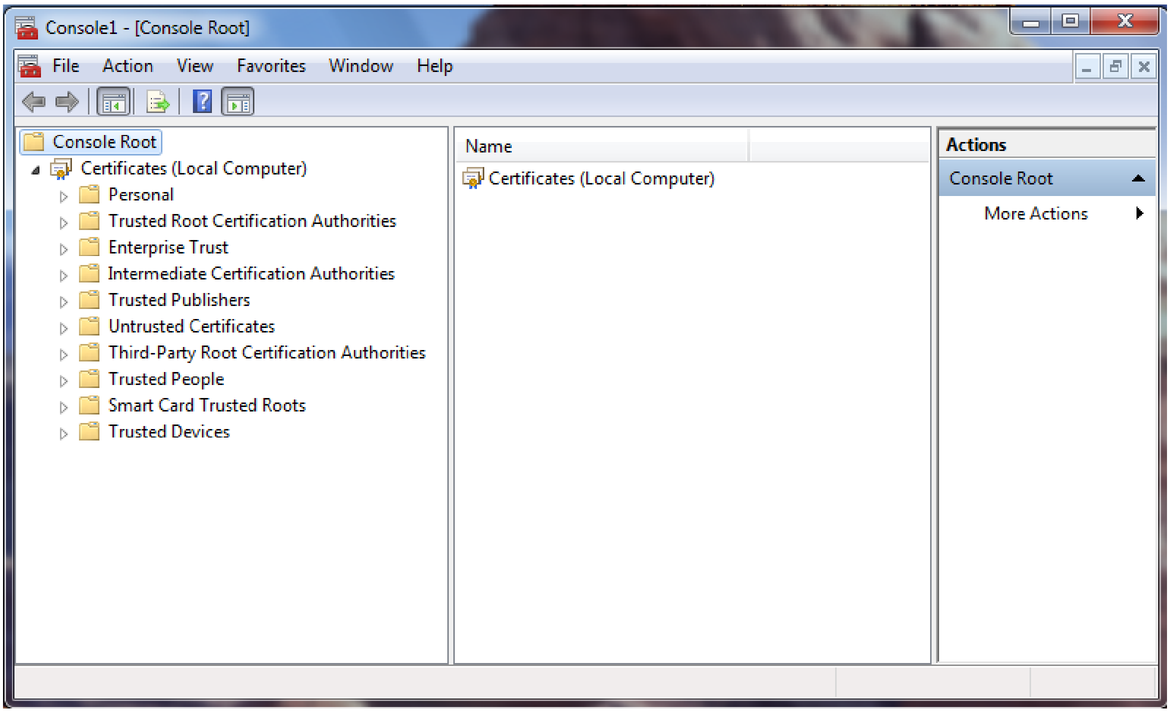


Figure 95 - Certificates (Local Computer) snap-in open in MMC

Note: The open snap-in must display "Certificates (Local Computer)" at this point. If it shows "Certificates - Current User" or anything else, you will need to stop, close it, and go back to repeat the above steps, but making sure this time that you perform them exactly as indicated.

Then, select the Personal container, and launch the Certificate Enrollment wizard by invoking:

Personal -> Right click -> All Tasks -> Import...

The following window should appear:

Figure 96 - Certificate Import Wizard

Click Next, and fill in the full path to the file containing the .p12 file (e.g.: C:\tmp\PCALICE.p12):

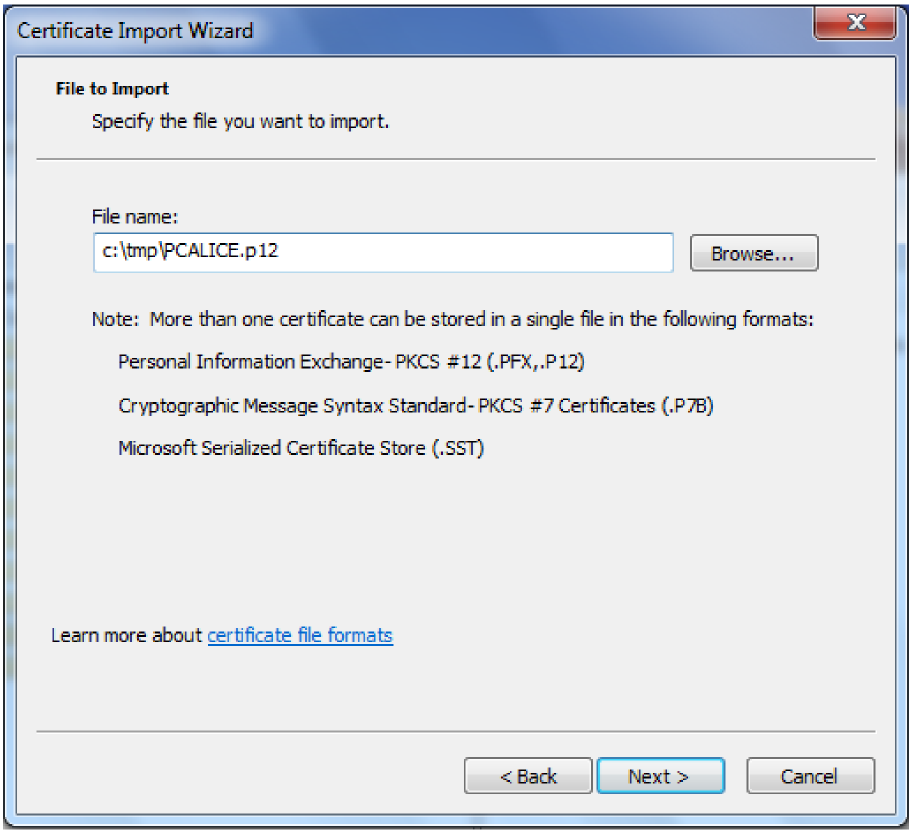


Figure 97 - Importing the contents of PCALICE.p12

Click Next. In the following window, enter the password (pass phrase) protecting the private key of PCALICE inside PCALICE.p12 (you selected that pass phrase when you created the .p12 file) and select the option "Include all extended properties":

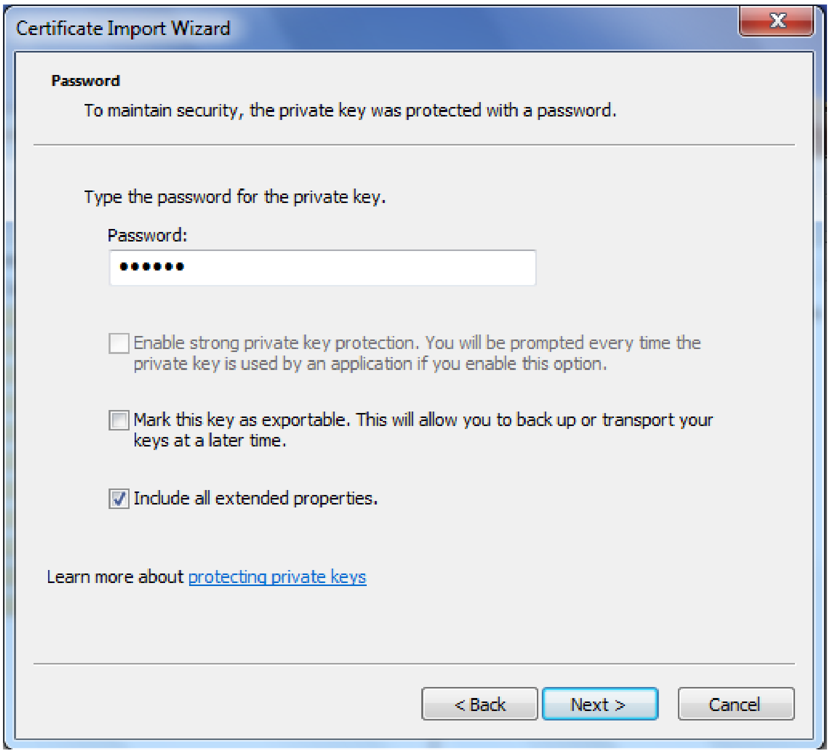


Figure 98 - Selection of the certificate store where the certificate will be placed

Note: You may as well select the option "Mark this key as exportable" if you want to be able to back it up from that system later on.

Click Next, and on the next window verify that the certificate store where the certificate will be placed is "Personal":

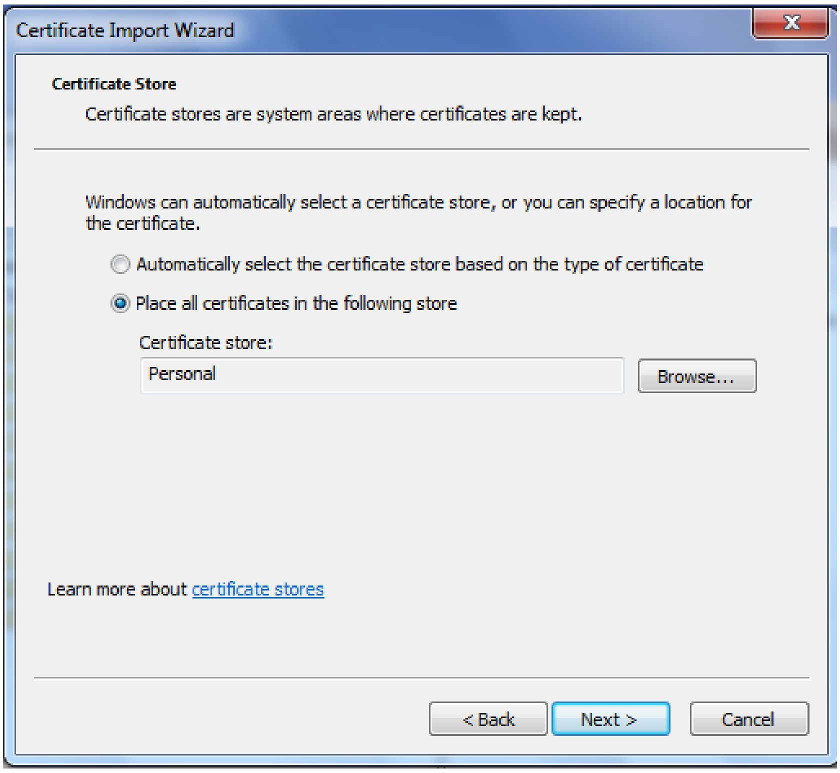


Figure 99 - Selection of the certificate store where the certificate will be placed

Click Next and finally Finish, to complete the import of the certificate. After a few seconds, you should get a pop-up window confirming that the import was successful:

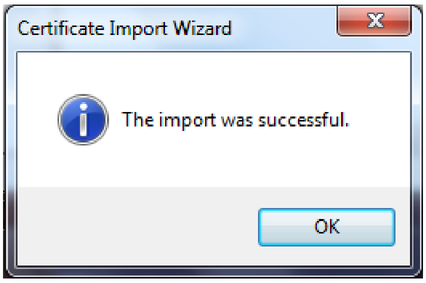


Figure 100 - Confirmation of a successful import of a certificate

Click OK to dismiss it.

Then, expand the target container (Certificates (Local Computer) \ Personal \ Certificates), and verify that two certificates were successfully imported there: one issued to PCALICE by "Example2 Root CA", and the other issued by "Example2 Root CA" to itself:

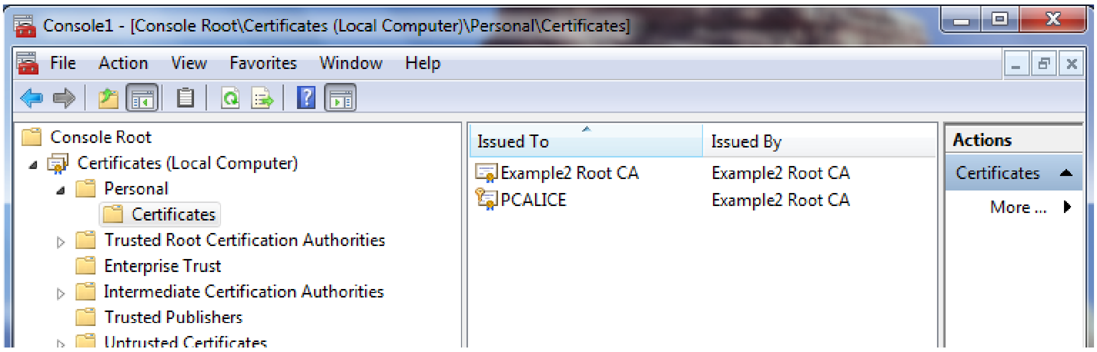


Figure 101 - Both certificates imported into the Personal\Certificates container

The certificate issued to PCALICE must stay in that container, but the certificate issued to "Example2 Root CA", which is the certificate of the CA itself, must be moved to the container "Trusted Root Certification Authorities\Certificates". You may do so cutting the certificate here (select certificate > right click > Cut) and then pasting it in the "Trusted Root Certification Authorities\Certificates" container (select the target container > right click > Paste), or you may simple drag&drop the file from one container to the other.

The end result must be that the certificate issued to PCALICE must stay in the "Personal\Certificates" container, while the certificate issued to "Example2 Root CA" must be located in the "Trusted Root Certification Authorities\Certificates" container:

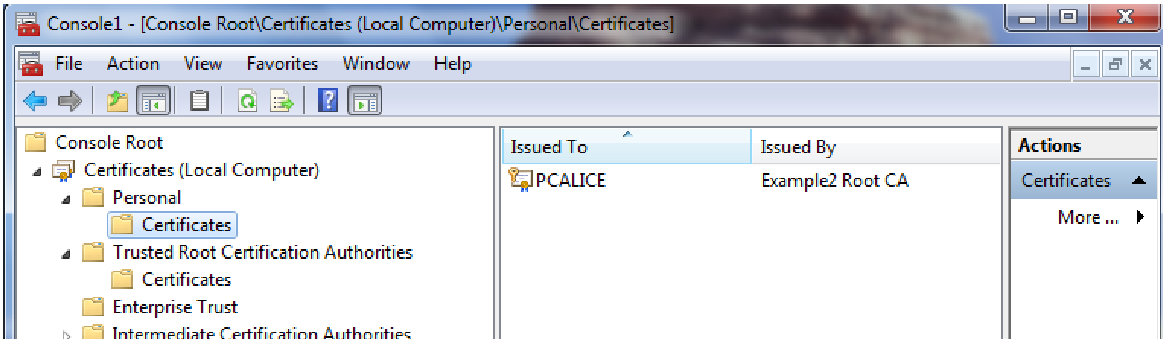


Figure 102 - PCALICE's certificate must stay in the Personal\Certificates container

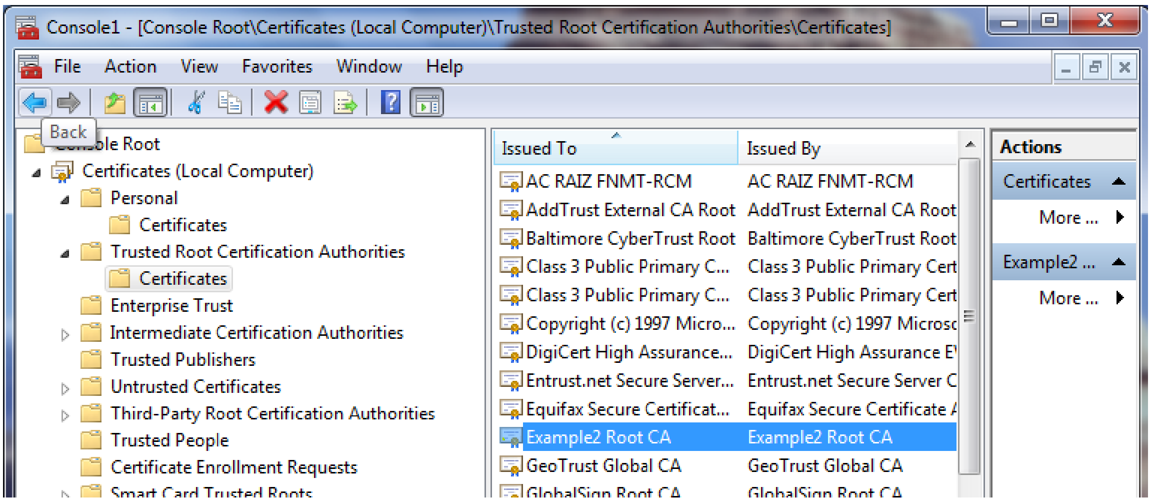


Figure 103 - The CA certificate must be located in the Trusted Root Certification Authorities\Certificates container

You may double click PCALICE's certificate to display their contents and verify its intended purposes:

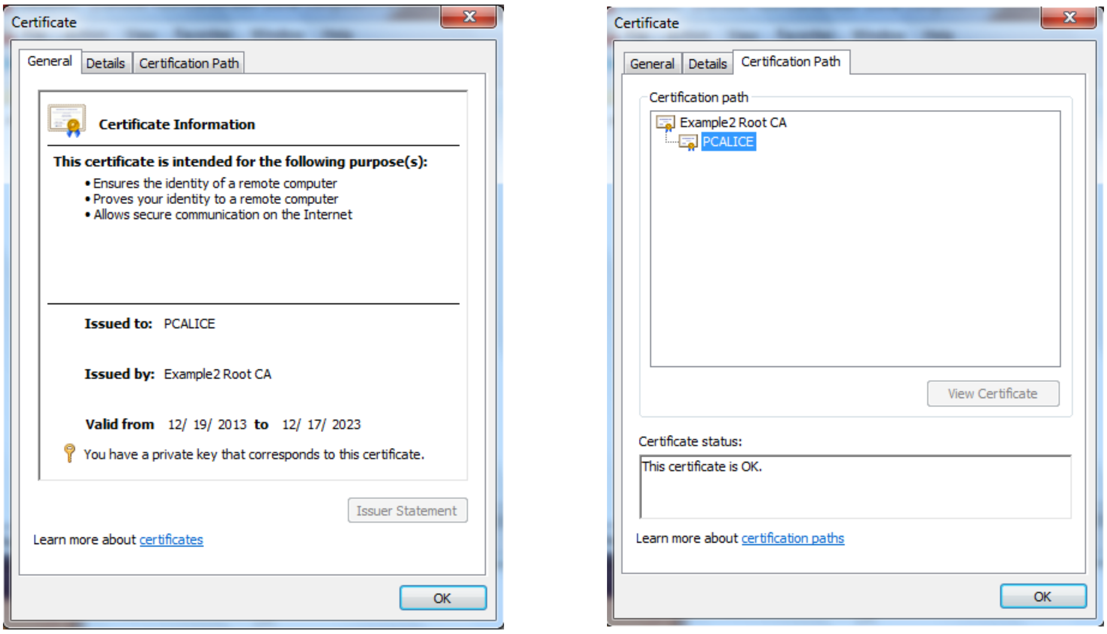


Figure 104 - PCALICE certificate

4.13 Step 12: On PCALICE, configure default options for IPsec

See section *3.16, Step 15: On PCALICE, configure default options for IPsec*.

4.14 Step 13: On PCALICE, create a connection security rule (IPsec) for traffic exchanged with PCBOB

See section *3.17, Step 16: On PCALICE, create a connection security rule (IPsec) for traffic exchanged with PCBOB.*

4.15 Step 14: On PCBOB, mirror steps 11 to 13

On PCBOB, mirror the steps 11 to 13 that you just performed on PCALICE, but swapping any references to PCALICE and PCBOB, and using the IP address of PCALICE as the remote endpoint when configuring the connection security rule on PCBOB.

This will get PCBOB ready to exchange IPsec traffic with PCALICE, and that is what you will verify in the next step.

4.16 Final step: Exchange traffic between PCALICE and PCBOB and verify it gets protected with IPsec

See section *3.19, Final step: Exchange traffic between PCALICE and PCBOB and verify it gets protected with IPsec.*