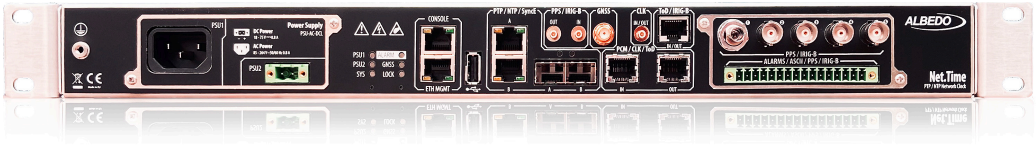

Net.Time: HTTPS Management with Custom SSL Certificates

Widely available safe communications are a key factor to enable e-commerce in the Internet era. The same protocols and procedures that provide ciphered and authenticated monetary transactions in the Internet can be reused in any environment when privacy between the communicating parties is required and also when the receiver must guarantee that the information has been generated by a legitimate source. A good example of this is the Albedo Net.Time network clock. Safe communications between Net.Time and remote users is necessary to avoid an intrusion. For this reason, Net.Time provides encryption and authentication in all management protocols. This document focuses in how to provide safe communications in the graphical management interface through a web application and, specifically, in how the client in a web session establishes trusted communications with Net.Time without any previous knowledge about its identity. We will see that this is only possible through the participation of an entity trusted by the client, a *Certification Authority* (CA), that guarantees that the server is who it claims to be.

Web traffic is delivered through the *Hyper-Text Transfer Protocol* (HTTP), a client-server protocol that enables the communicating parties to exchange information encoded by the the *Hyper-Text Markup Language* (HTML) or any of its extensions. HTML pages are displayed by web browsers allowing them to represent formatted text and graphics or even to deploy sophisticated applications. HTTP conveys web traffic without worrying about privacy, which means that any transmitted information could be read by anybody with access to the transmission medium. HTTP also lacks of authentication features. Attempts to replace a legitimate web server would be undetectable by the client entities. The *Secure Sockets Layer* (SSL), or in its more modern version, the *Transport Layer Security* (TLS), can be deployed on the top of HTTP to provide privacy and authentication. HTTP, when used together with SSL/TLS, is referred as HTTPS. A very common implementation of SSL, known as *OpenSSL* is used in the examples described in this application note. Open SSL enables users to generate keys and certificates required to deploy the HTTPS security infrastructure.

SSL/TLS makes extensive use of modern cryptographic techniques such as symmetric encryption, public /private key asymmetric encryption, and digital signatures. The following section provides brief introduction to these subjects.

1. Cryptographic Signatures

An important concept in cryptography is asymmetric encryption as opposed to symmetric encryption. Both of them are based on keys to encrypt and decrypt messages. Symmetric encryption allows anyone who knows the encryption key to also decrypt the message. This is not true when asymmetric encryption is used. In this case, encryption and decryption keys are different. Messages are encrypted by a key that does not need to be secret and, indeed, this key could be published to allow anyone to encrypt a message. For this reason, this key is known as the *public key*. In public / private key asymmetric encryption, only the key required to encrypt messages (*private key*) must be kept secret. Techniques derived from the number theory and other areas in Mathematics are the basis to public / private key encryption. The main idea behind these techniques it to make the computation of the private key from the public key computationally unfeasible.

For our purpose, it is important o analyze the result of processing the message by the private key instead of the public key. At first sight it seems that nothing useful is achieved in this case because anyone with

access to the public key could theoretically decrypt the message, but if we consider that only the private key owner is allowed to process a message with this key, we can conclude that this process enables the receiver to bind the message with the author identity. In other words, processing a message with a private key enables the author to *sign* the message.

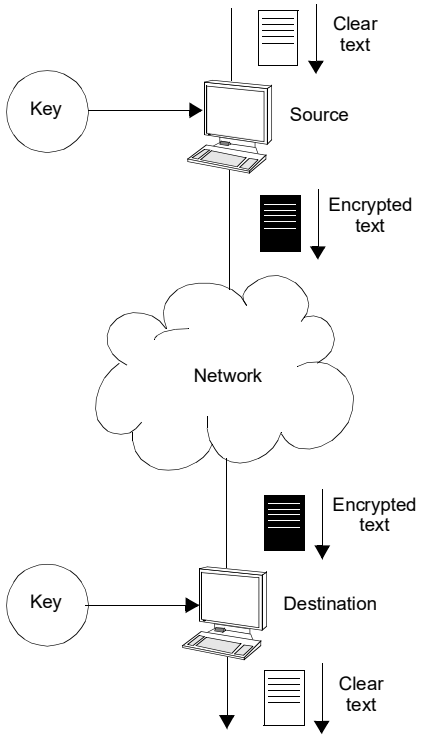

Figure 1 *Operation principle of symmetric encryption*

Actually, it is not necessary to process the whole message with the private key. In practical applications only a short, fixed length, bit sequence computed from the message needs to be processed by the private key. This bit sequence is known as a cryptographic hash. When the hash is signed it becomes a digital signature that can be appended to the message to prove that it has been generated by a legitimates source. A cryptographic hash is useful only if it meets certain requirements. The most important is that it must be computationally unfeasible to find two different messages providing the same hash. Otherwise, it would be possible to reuse digital signatures in messages from non-legitimate sources

Digital signatures are basis of modern authentication mechanisms based on certificates. A digital certificate contains information about the message generator identity and a signature from a CA which is trusted by all the potential message receivers. The role of the CA in the digital certificate is to provide trust by binding identity data (domain names, IP addresses or any other) to their legitimate owners. In other words, the CA ensures that the message generators are who they claim to be.

2. Obtaining a Certificate Signed by a CA

The purpose of the the certification process it to provide a means to prove the authenticity of your Net.Time servers to anybody accessing to them through the HTTPS protocol. In order to do that, an external entity, a CA accepted by all potential web clients, provides trust by signing a certificate supplied by your web site when these clients access to it. Clients can extend the trust to your site because they trust the CA that has signed the certificate.

In practical terms, getting a signed certificate requires that you provide a *Certificate Signing Request* (CSR) file to the CA, which is a file that must contain at least the following information:

- A digital signature generated by the private key from the organization that generates the CSR. Since only this organization is able generate such signature, the CSR can be use as a proof of identity for the organization.
- A public key that, together with the digital signature identifies the organization that issues the CSR. Indeed, since there is no need to keep public keys secret, this part of the CSR could be obtained by a different means. Could be distributed by e-mail or published in the Internet, for example.
- Identities corresponding to the objects or entities to be certified. They are typically IP addresses, host names or domain names. A single CSR file could be used to certify the identity of several objects at the same time.

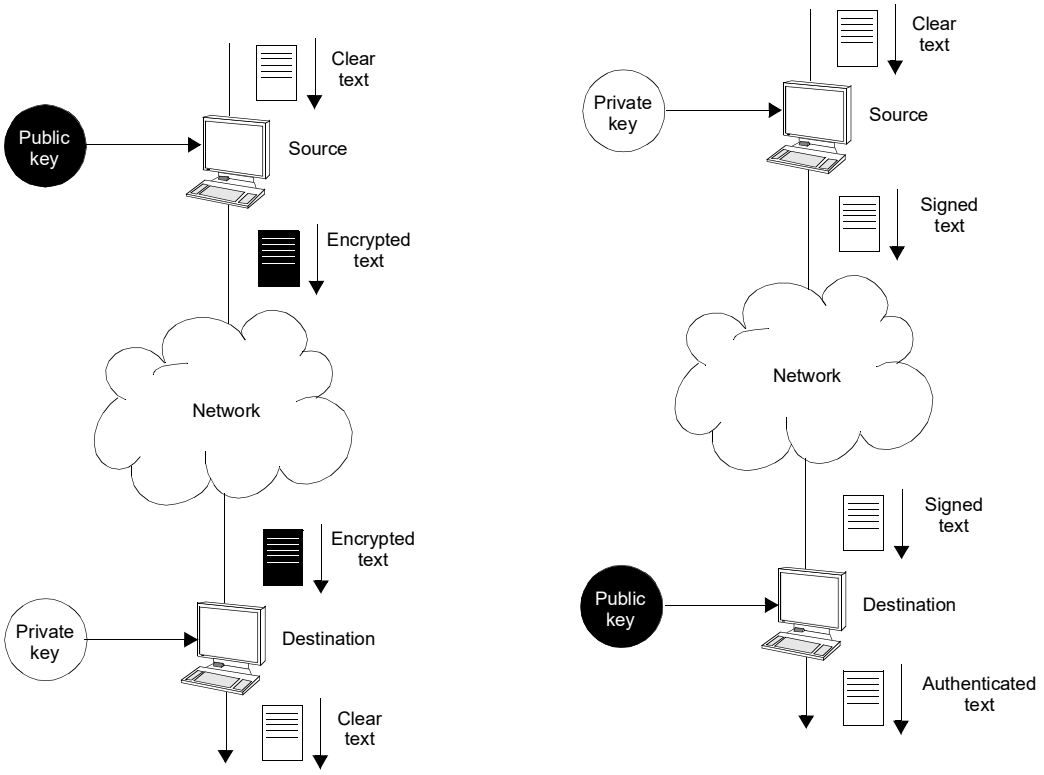

Figure 2 *Public / private key cryptography: (a) Operation principle of public key encryption, (b) operation principle of public key signatures.*

The purpose of the certification process is to bind the objects (IP addresses, host names, domain names) specified in the CSR to the organization that issues the certification request. At the end of the process, the organization receives a CRT file that contains basically the same data than the CSR and the signature from the CA.

To generate a CSR for the CA of your choice using OpenSSL, you need to follow these steps:

1. Create a private key. This key will be used to sign your certificate. Since this key contains sensitive information it is a good idea to encrypt its content. In the command line, the key is ciphered with the symmetric 128-bit *Advanced Encryption Standard* (AES) by means the *-aes128* option. You will be prompted to add a password to the key before it is created.

```
user@host:~$ openssl genrsa -aes128 -out ntime-mkt.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
.....................................................................................+++++
e is 65537 (0x010001)
Enter pass phrase for ntime-mkt.key:
Verifying - Enter pass phrase for ntime-mkt.key:
```

2. The previous command generates a base-64 encoded text message that you can display with the following command.

```
 user@host:~$ cat ntime-mkt.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
```

```
DEK-Info: AES-128-CBC,AB2E759D3D290EF072589CF27AFE7B4B

hL9VjFdI2PYO/EudI3wkpJq4LRWfe+dJmUc452sHSTEMdGes0RPAEMdgBzTyUNDW
+32/6AYqqLlKavBY3pfLJN8VR/LLRRmIXCqEFuAiwpR290WoDhNJ2J7Pxl5gNKgW
zDAGaKECWTHnhYos4HXsNGkoHTPnsISFoDDFwtIgbwGTSXgBA4vCMAH5TljKrUrc
uhXXXTO3Empic59bEt2vaDd3EyY/D5ndqpvxLKk4KgkDqpmyjk302uW+Ngkj1G4c
2aHU+SqC/9h1bk6R+H9nhxvd2maAf2yFroF0OMQyfODdIYMRmpoNyXtfNKUckJ7P
DcoZeE9S1ADKu119ZhAAcnQjhde6zGuQJOBnZsc+MVXVbDXylGXJi/tp8cNSvNRq
4v//aKrMvqkeaW3LrL/lRfxFnmg0dy8aoX/SGpzS0FGOtRoVDchnKkdfDA+pl7Rk
/FIhyn4BBuNCHFNbSsKqNRfLEQqByDkHvLufAfOMWlnslk1wLXh6Gzq4lyLzVEnh
0eUEi701YQzAN0W+O3g/M+9WMlfBoCC6kcM71m64vMnCGZlMkeQA3GVIxqvWpNyB
o6zmdi/PRiBP21nJxSncjxVshfvWVIQhEqfxRoDgBRqjWhxkgMpp6yigARmTk+xm
i510J0VmKWmsBXkMiNOpJqoz2DHAFGdhoL4+LT0no2koHdVfnnBVWVlVbL2M5yuI
VjxMhD/Uig22XbJ3j/pxaKoJEPFSpIwjIHqDPEVVReYlg39CLytbKittyI1FevPZ
vpQ5WyNW1AKwwNPUWXMErHeJqjAcSI0TKMP7CWehVJsfNNYllWFvu3tZUsg9lK1v
wJ/XZK7xxymQzCQN+XUiXl+GAlCKP69fSdJduW6LF2X9IWlUfoceuE/+3Ny/WFrd
ejqJFmvXrg44G8rIJqRLZuEyN6CR1Gf2arhp9pqQZz1CmtlxQkWZ7QDMGG6RKlHe
Io08iB+zgPwP9f8meYs6gQS7F3knBAzaMXtvE5uNEFUp6W2ZHWEFaWH0z+m75Dht
Uqhf0OU3cUiO+YylCXasCwYc3v4tgrfC/lddN3h0L7K4V0Fx5r+KFpIQPWgcmf4O
fSVYt+It72QfBFT3zFlPfwLW9A+TDqY9ff0uMDuNdXt8nqhzFrCQj17q3MyNymtb
Rk5TtKl8koP4SkzizvuQKiRG8ghuOcY3O0ZO3ah4fhzNjvk/dGf/BnjURCp8mu3E
7Ja4rglYm4eTM7gJ7Nda5zRD6aQQU25E7d9SJVgMpImMv9D2IUmnNLMmN0Z2RKs+
3aBS5K/8SL1CcK0p2XvvDxXghtwK6I6K10i+HF1RLKFzjr1My2krOZrK0w8gDnqs
AWVx6wUT5jA5mkSWy7qTJiSz0SpCGiya3qubcTcA7Xr5kkEm52b+i2ZpljOUybxl
m8gb+85Q/efgLJVKxEGmo1TfF1kqU75YXAv+SYyaC8ta5QxijSIpcixit8aPAaGy
lO6FGp3WTbzudwWP/WCwCvsMf7B4MCcMJDGitljfY2Oa7xFcBeqVR+6vk7h6O3O7
FupxipnLdzAAWI9tRihvq1CablMHL7wgDcj2ier9TJpvxnNefzYISYzi96s9eW9x
-----END RSA PRIVATE KEY-----
```

3. Generate a configuration file with the information you need to include in the CSR. You can include domain names and IP addresses from the Net.Time clocks that you need to include in the certificate. The example generates a certificate for two clocks: *ntime01* (*ntime01.albedo.biz*) and *ntime02* (*ntime02.albedo.biz*). The file, named *ntime-mkt-01.cfg*, could have the following content that it may be adapted to your own requirements:

```
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth, clientAuth

[req]
distinguished_name = req_distinguished_name
req_extensions = req_ext

[req_distinguished_name]
countryName                     = Country Name (2 letter code)
countryName_default             = ES
countryName_min                 = 2
countryName_max                 = 2
stateOrProvinceName             = State or Province Name (full name) ## Print this message
stateOrProvinceName_default     = CATALONIA
localityName                    = Locality Name (eg, city)
localityName_default            = BARCELONA
0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = ALBEDO Telecom SL
organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Marketing
commonName                      = Common Name (eg, your name or your server hostname)
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 64

[req_ext]
subjectAltName = @alt_names

[alt_names]
```

```
DNS.1 = ntime01
DNS.2 = ntime01.albedo.biz
DNS.3 = ntime02
DNS.4 = ntime02.albedo.biz
```

4. Generate the CSR with the following command. During the process you will be prompted to to enter the information required to generate the file. If the private key is encrypted you will need to enter the password assigned to it.

```
user@host:~$ openssl req -new -key ntime-mkt.key -out ntime-mkt-01.csr -config ntime-mkt-01.cfg
Enter pass phrase for ntime-mkt.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [CATALONIA]:
Locality Name (eg, city) [BARCELONA]:
Organization Name (eg, company) [ALBEDO Telecom SL]:
Organizational Unit Name (eg, section) [Marketing]:
Common Name (eg, your name or your server hostname) []:ntime.albedo.biz
Email Address []:info@albedotelecom.com
```

5. At the end of the process, the CSR, a file with the name *ntime-mkt-01.csr* is generated. You can also display the content of the newly generated text file with the following command:

```
user@host:~$ cat ntime-mkt-01.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIDrTCCApUCAQAwgacxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDQVRBTE9OSUEx
EjAQBgNVBAcMCUJBUkNFTE9OQTEaMBgGA1UECgwRQUxCRURPIFRlbGVjb20gU0wx
EjAQBgNVBAsMCU1hcmtldGluZzEZMBcGA1UEAwwQbnRpbWUuYWxiZWRvLmJpejEl
MCMGCSqGSIb3DQEJARYWaW5mb0BhbGJlZG90ZWxlY29tLmNvbTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAO7Rk1q97bOmCgoTCNW9vvaDePUHfEdWFuEy
AfLRJLy4vbiPbU+QGRyGYWxSCSk80EbYDIljdTfA2y/P5CvRv/HyPhbo6Vp+zDh8
dMBKcM7gRsoBGSKWVJRqjGvmiiCXM7cVwW7OaX2PtWudxbFe8mwSvV+nlEJGZCWe
9v0kPal/TFB7KmXOO7zP9IftPA1M3YTiE8I3KnkfrKcxqxD4ShcZbeZZjS/SN+fJ
FYjmaQUcT9kmwUWonbpZa7X4qiBvA+fF7Qn3gokL+q4AO9pxpBg+vDvl4uBUohr4
osnmwiYuOy76paIX+GfdZdLaDcgMGf86OEG9i/0mmY9DEUryGY0CAwEAAaCBvzCB
vAYJKoZIhvcNAQkOMYGuMIGrMIGoBgNVHREEgaAwgZ2HBKwaA2SHBKwaA2eCB250
aW1lMDGCEm50aW1lMDEuYWxiZWRvLmJpeoIHbnRpbWUwMoISbnRpbWUwMi5hbGJl
ZG8uYml6ggdudGltZTAzghJudGltZTAzLmFsYmVkby5iaXqCB250aW1lMDSCEm50
aW11MDQuYWxiZWRvLmJpeoIHbnRpbWUwNYISbnRpbWUwNS5hbGJlZG8uYml6MA0G
CSqGSIb3DQEBCwUAA4IBAQBHeMarhHuZc/fsSl0JOg3qRTfjXbniI+l6YvcMG7fY
AhYAGvvL97pTFTDqAqBdD1fhp5s1Z1iWOxrerOOsGdDoOCgCWLu5bu2kJG/VfJWc
x2VU6/z0RkJgrCJtIFCifvjd/aD6JJhUjLjBtOfYwx6sN6hGUpeZytl3P5vTBMjj
lrYqczc3mMytyKj5DY+rOSESgoUBlHY5p0BrMTlJC4voDAIQkNeJrvv0ZCuw01S8
A6sECa2kXg1j1vnJgManK7qXxukV8pFtJVkXnWPbBl479cPAVumTHi1X8iaAPYDv
8ACZYoJS35tZGS29TdjfTmmKvdX/XmLpJPCNqj0821qC
-----END CERTIFICATE REQUEST-----
```

6. The encoding used in the CSR does not reveal its structure when it is displayed in text format. Use the following command to display detailed information about the CSR just generated. It is important that you verify that Net.Time domain names and/or IP addresses have been properly included in the file.

```
user@host:~$ openssl req -noout -text -in ntime-mkt-01.csr
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = ES, ST = CATALONIA, L = BARCELONA, O = ALBEDO Telecom SL, OU = Marketing, CN =
ntime.albedo.biz, emailAddress = info@albedotelecom.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
```

```
                Modulus:
                    00:ee:d1:93:5a:bd:ed:b3:a6:0a:0a:13:08:d5:bd:
                    be:f6:83:78:f5:07:7c:47:56:16:e1:32:01:f2:d1:
                    24:bc:b8:bd:b8:8f:6d:4f:90:19:1c:86:61:6c:52:
                    09:29:3c:d0:46:d8:0c:89:63:75:37:c0:db:2f:cf:
                    e4:2b:d1:bf:f1:f2:3e:16:e8:e9:5a:7e:cc:38:7c:
                    74:c0:4a:70:ce:e0:46:ca:01:19:22:96:54:94:6a:
                    8c:6b:e6:8a:20:97:33:b7:15:c1:6e:ce:69:7d:8f:
                    b5:6b:9d:c5:b1:5e:f2:6c:12:bd:5f:a7:94:42:46:
                    64:25:9e:f6:fd:24:3d:a9:7f:4c:50:7b:2a:65:ce:
                    3b:bc:cf:f4:87:ed:3c:0d:4c:dd:84:e2:13:c2:37:
                    2a:79:1f:ac:a7:31:ab:10:f8:4a:17:19:6d:e6:59:
                    8d:2f:d2:37:e7:c9:15:88:e6:69:05:1c:4f:d9:26:
                    c1:45:a8:9d:ba:59:6b:b5:f8:aa:20:6f:03:e7:c5:
                    ed:09:f7:82:89:0b:fa:ae:00:3b:da:71:a4:18:3e:
                    bc:3b:e5:e2:e0:54:a2:1a:f8:a2:c9:e6:c2:26:2e:
                    3b:2e:fa:a5:a2:17:f8:67:dd:65:d2:da:0d:c8:0c:
                    19:ff:3a:38:41:bd:8b:fd:26:99:8f:43:11:4a:f2:
                    19:8d
                Exponent: 65537 (0x10001)
            Attributes:
            Requested Extensions:
                X509v3 Subject Alternative Name:
                    DNS:ntime01, DNS:ntime01.albedo.biz, DNS:ntime02, DNS:ntime02.albedo.biz
        Signature Algorithm: sha256WithRSAEncryption
             47:78:c6:ab:84:7b:99:73:f7:ec:4a:5d:09:3a:0d:ea:45:37:
             e3:5d:b9:e2:23:e9:7a:62:f7:0c:1b:b7:d8:02:16:00:1a:fb:
             cb:f7:ba:53:15:30:ea:02:a0:5d:0f:57:e1:a7:9b:35:67:58:
             96:3b:1a:de:ac:e3:ac:19:d0:e8:38:28:02:58:bb:b9:6e:ed:
             a4:24:6f:d5:7c:95:9c:c7:65:54:eb:fc:f4:46:42:60:ac:22:
             6d:20:50:a2:7e:f8:dd:fd:a0:fa:24:98:54:8c:b8:c1:b4:e7:
             d8:c3:1e:ac:37:a8:46:52:97:99:ca:d9:77:3f:9b:d3:04:c8:
             e3:96:b6:2a:73:37:37:98:cc:ad:c8:a8:f9:0d:8f:ab:39:21:
             12:82:85:01:94:76:39:a7:40:6b:31:39:49:0b:8b:e8:0c:02:
             10:90:d7:89:ae:fb:f4:64:2b:b0:d3:54:bc:03:ab:04:09:ad:
             a4:5e:0d:63:d6:f9:c9:80:c6:a7:2b:ba:97:c6:e9:15:f2:91:
             6d:25:59:17:9d:63:db:06:5e:3b:f5:c3:c0:56:e9:93:1e:2d:
             57:f2:26:80:3d:80:ef:f0:00:99:62:82:52:df:9b:59:19:2d:
             bd:4d:d8:df:4e:69:8a:bd:d5:ff:5e:62:e9:24:f0:8d:aa:3d:
             3c:db:5a:82
```

The CSR enables you to request a CRT for your Net.Time servers from a CA by a procedure that depends on the specific CA that is issuing the file, but something that is common to all them is that in some way or another they will need to set the authenticity of any IP address, host name or domain name to be certified by them.

3. Enabling the Certificate in Net.Time

An important point is that it is not enough to upload the CRT file to Net.Time to enable proper operation of HTTPS. It is true that the certificate provided by the CRT enables any remote host to authenticate a Net.Time server. The public key included in the certificate also allows remote hosts to encrypt data transmitted to Net.Time. However, in order to decrypt this data, Net.Time must have access to the private key as well. Asymmetric key encryption is used in HTTPS sessions only in the initial stage. This approach allows the endpoints to exchange symmetric keys required for exchanging data. However, for the initial stage, the private key is required. The procedure to upload the certificate and the private key to Net.Time is described below.

1. Log into the Net.Time unit with an administrator account.

```
user@host:~$ ssh admin@ntime01
admin@ntime01's password:
Last login: Mon Jul  7 08:14:11 2025 from 172.26.4.90
```

```
Net.Time - Control console

You are running with administrator privileges.


admin@ntime01#
```

2. Type the following commands in Net.Time from an administrator account to enable HTTPS.

```
admin@ntime04# set netmanager protocol http disable
admin@ntime04# set netmanager protocol https disable
```

3. Net.Time is now running HTTPS with a default certificate. There is some information about this certificate that can be displayed with the following command.

```
admin@ntime01# file certificate list internal

Name                            Type        Enabled                    Validity
-------------------------------------------------------------------------------
default                         X.509       Yes             2042-01-07 17:39:12

1 certificate(s) listed.
```

4. It is unlikely that the default certificate could be used in any realistic scenario. For this reason the default certificate must be replaced by a CRT file provided by a CA. To upload a CRT to Net.Time, type the following command:

```
admin@ntime01# file certificate import name ntime-mkt-01 interactive

Type below the X.509 certificate. It should begin with
"-----BEGIN CERTIFICATE-----" and end with
"-----END CERTIFICATE-----" (Press Ctrl-D to finish):

>
```

5. Now, copy and paste the content of the CRT file you got from the CA. Finish with CTRL-D

```
> -----BEGIN CERTIFICATE-----
> MIIEajCCA1KgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTAkVT
> MRIwEAYDVQQIDAlDYXRhbG9uaWExEjAQBgNVBAcMCUJhcmNlbG9uYTEaMBgGA1UE
> CgwRQUxCRURPIFRlbGVjb20gU0wxEjAQBgNVBAsMCU1hcmtldGluZzEcMBoGA1UE
> AwwTQUxCRURPIFRlbGVjb20gVGVzdDElMCMGCSqGSIb3DQEJARYWaW5mb0BhbGJl
> ZG90ZWxlY29tLmNvbTAeFw0yNTA3MDcwODU1MDdaFw0yNjA3MDcwODU1MDdaMIGT
> MRkwFwYDVQQDDBBudGltZS5hbGJlZG8uYml6MRIwEAYDVQQIDAlDQVRBTE9OSUEx
> CzAJBgNVBAYTAkVTMRowGAYDVQQKDBFBTEJFRE8gVGVsZWNvbSBTTDESMBAGA1UE
> CwwJTWFya2V0aW5nMSUwIwYJKoZIhvcNAQkBFhZpbmZvQGFsYmVkb3RlbGVjb20u
> Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7tGTWr3ts6YKChMI
> 1b2+9oN49Qd8R1YW4TIB8tEkvLi9uI9tT5AZHIZhbFIJKTzQRtgMiWN1N8DbL8/k
> K9G/8fI+FujpWn7MOHx0wEpwzuBGygEZIpZUlGqMa+aKIJcztxXBbs5pfY+1a53F
> sV7ybBK9X6eUQkZkJZ72/SQ9qX9MUHsqZc47vM/0h+08DUzdhOITwjcqeR+spzGr
> EPhKFxlt5lmNL9I358kViOZpBRxP2SbBRaidullrtfiqIG8D58XtCfeCiQv6rgA7
> 2nGkGD68O+Xi4FSiGviiyebCJi47Lvqlohf4Z91l0toNyAwZ/zo4Qb2L/SaZj0MR
> SvIZjQIDAQABo4GuMIGrMIGoBgNVHREEgaAwgZ2HBKwaA2SHBKwaA2eCB250aW1l
> MDGCEm50aW1lMDEuYWxiZWRvLmJpeoIHbnRpbWUwMoISbnRpbWUwMi5hbGJlZG8u
> Yml6ggdudGltZTAzghJudGltZTAzLmFsYmVkby5iaXqCB250aW1lMDSCEm50aW1l
> MDQuYWxiZWRvLmJpeoIHbnRpbWUwNYISbnRpbWUwNS5hbGJlZG8uYml6MA0GCSqG
> SIb3DQEBCwUAA4IBAQB5VGoEXTKsEhb9ztXu0AbhdE0TFaVPiQkriQmrUcahaNZl
> gcCR1kGmkdEvTrAIajr6yZJlRTHieqxFAxvmGs3wbld4ZGZrY48U1gPAIcqH6WpF
> eNt8QcuKhELN1zuByIjEj1rzpw1Ioh5VH4dWnReg+w5H5sFGIO90VM+hxljtO9Zz
> 9OqjYobW+lociEjRaehg5Xce8fL5u23M5QtY3YYea3JdaDcwmo7/5jkVRM4ltYlc
> RmIjvujy1XZpn1T0ZhtALJJc6mHJF6W9WZCTpLkND0D6epW/1nSduwuNIVyV2ddD
> IDH0IFfI4MzuOBgNlKtv0wo/oeTloxESaDaHz027
> -----END CERTIFICATE-----
>
```

6. The unit now requests the administrator to enter the private key. Copy and paste the content from the private key as you did for the CRT. Finish with CTRL-D.

```
Type below the private key. It should begin with
"-----BEGIN PRIVATE KEY-----" and end with
"-----END PRIVATE KEY-----" (Press Ctrl-D to finish):

> -----BEGIN RSA PRIVATE KEY-----
> Proc-Type: 4,ENCRYPTED
> DEK-Info: AES-128-CBC,AB2E759D3D290EF072589CF27AFE7B4B
>
> hL9VjFdI2PYO/EudI3wkpJq4LRWfe+dJmUc452sHSTEMdGes0RPAEMdgBzTyUNDW
> +32/6AYqqLlKavBY3pfLJN8VR/LLRRmIXCqEFuAiwpR290WoDhNJ2J7Pxl5gNKgW
> zDAGaKECWTHnhYos4HXsNGkoHTPnsISFoDDFwtIgbwGTSXgBA4vCMAH5TljKrUrc
> uhXXXTO3Empic59bEt2vaDd3EyY/D5ndqpvxLKk4KgkDqpmyjk302uW+Ngkj1G4c
> 2aHU+SqC/9h1bk6R+H9nhxvd2maAf2yFroF0OMQyfODdIYMRmpoNyXtfNKUckJ7P
> DcoZeE9S1ADKu119ZhAAcnQjhde6zGuQJOBnZsc+MVXVbDXylGXJi/tp8cNSvNRq
> 4v//aKrMvqkeaW3LrL/lRfxFnmg0dy8aoX/SGpzS0FGOtRoVDchnKkdfDA+pl7Rk
> /FIhyn4BBuNCHFNbSsKqNRfLEQqByDkHvLufAfOMWlnslk1wLXh6Gzq4lyLzVEnh
> 0eUEi701YQzAN0W+O3g/M+9WMlfBoCC6kcM71m64vMnCGZlMkeQA3GVIxqvWpNyB
> o6zmdi/PRiBP21nJxSncjxVshfvWVIQhEqfxRoDgBRqjWhxkgMpp6yigARmTk+xm
> i510J0VmKWmsBXkMiNOpJqoz2DHAFGdhoL4+LT0no2koHdVfnnBVWVlVbL2M5yuI
> VjxMhD/Uig22XbJ3j/pxaKoJEPFSpIwjIHqDPEVVReYlg39CLytbKittyI1FevPZ
> vpQ5WyNW1AKwwNPUWXMErHeJqjAcSI0TKMP7CWehVJsfNNYllWFvu3tZUsg9lK1v
> wJ/XZK7xxymQzCQN+XUiXl+GAlCKP69fSdJduW6LF2X9IWlUfoceuE/+3Ny/WFrd
> ejqJFmvXrg44G8rIJqRLZuEyN6CR1Gf2arhp9pqQZz1CmtlxQkWZ7QDMGG6RKlHe
> Io08iB+zgPwP9f8meYs6gQS7F3knBAzaMXtvE5uNEFUp6W2ZHWEFaWH0z+m75Dht
> Uqhf0OU3cUiO+YylCXasCwYc3v4tgrfC/lddN3h0L7K4V0Fx5r+KFpIQPWgcmf4O
> fSVYt+It72QfBFT3zFlPfwLW9A+TDqY9ff0uMDuNdXt8nqhzFrCQj17q3MyNymtb
> Rk5TtKl8koP4SkzizvuQKiRG8ghuOcY3O0ZO3ah4fhzNjvk/dGf/BnjURCp8mu3E
> 7Ja4rglYm4eTM7gJ7Nda5zRD6aQQU25E7d9SJVgMpImMv9D2IUmnNLMmN0Z2RKs+
> 3aBS5K/8SL1CcK0p2XvvDxXghtwK6I6K10i+HF1RLKFzjr1My2krOZrK0w8gDnqs
> AWVx6wUT5jA5mkSWy7qTJiSz0SpCGiya3qubcTcA7Xr5kkEm52b+i2ZpljOUybxl
> m8gb+85Q/efgLJVKxEGmo1TfF1kqU75YXAv+SYyaC8ta5QxijSIpcixit8aPAaGy
> lO6FGp3WTbzudwWP/WCwCvsMf7B4MCcMJDGitljfY2Oa7xFcBeqVR+6vk7h6O3O7
> FupxipnLdzAAWI9tRihvq1CablMHL7wgDcj2ier9TJpvxnNefzYISYzi96s9eW9x
> -----END RSA PRIVATE KEY-----
>
Enter private key's encryption password:
```

7. If the certificate is accepted by Net.Time, then you can list it with the following command.

```
admin@ntime01# file certificate list internal

Name                            Type        Enabled                    Validity
-------------------------------------------------------------------------------
default                         X.509       Yes              2042-01-07 17:39:12
ntime-mkt-01                    X.509       No               2026-07-06 08:55:07


2 certificate(s) listed.
```

8. You can also display details about the certificate with the next CLI command.

```
admin@ntime01# file certificate show name ntime-mkt-01 internal

Metadata
-------------------------------------------------
Name:                                ntime-mkt-01
Source:                                  internal
Enabled:                                       no


Subject
-------------------------------------------------
Organization:                    ALBEDO Telecom SL
Common name:                      ntime.albedo.biz
```

```
Issuer
-------------------------------------------------
Organization:                    ALBEDO Telecom SL
Common name:                   ALBEDO Telecom Test


Validity
-------------------------------------------------
Not before:                    2025-07-06 08:55:07
Not after:                     2026-07-06 08:55:07
```

9. Once imported, the certificate must be enabled before it can be used. The required command is as follows:

```
admin@ntime01# file certificate set https ntime-mkt-01
```

The HTTPS web server starts using the new certificate as soon as it is enabled. If the process has been correctly done, any client trusting the CA used to sign the certificate will be able to authenticate web pages generated by the server.

4. Creating your Own CA

You can create a CA that will enable you to sign CSRs generated in your organization. Unless your CA is not trusted by a higher rank CA you will not be able to use certificates signed by your CA universally, but you can still use them within your organization. The steps to follow to create your CA are described below.

1. Generate a private key for your CA. The following example generates an *atsl-ca.key* private key. As an option, the private key can be encrypted. The example encrypts the private key with the *Data Encryption Standard* (DES) protocol.

```
user@host:~$ openssl genrsa -des -out atsl-ca.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................+++++
.....................................................................................................++
+++
e is 65537 (0x010001)
Enter pass phrase for atsl-ca.key:
Verifying - Enter pass phrase for atsl-ca.key:
```

2. The previous command generates a base-64 encoded text message that you can display with the following command.

```
user@host:~$ cat atsl-ca.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-CBC,3CA61B9BA9FD9103

JK8gMXm9maPcW5x76xT9aX8keLnjtroZDvhvzqR9IfW1zjdjwvchuST0ep0w/JH/
5npQx8cAL8ji/xlF1RFVoylcJok0S5ozZCeLTcTCpwWVJ5DOZz3ja+S+2u14S4P6
UiWjOyLZQTm2azO1nB504oovDbmOdeHYNSssWPc8EWnWgEG7Ne+uT6/k2cwLLSSq
pcuFrO5vM/cLmlLNIbxeYeIrDsyZrEhBc2TuijkP5Ep3x26+ERedIXTwB66hTEAm
vtYfzuf5K6zryFghj301lBTaVG/KUWkoB1/CjguSrWwxwb3laZQDJAEDsY+QK2IL
NqFLwz+gHHr5nzpK7RxwUy4mLptlR+AfTzAQdHm1it4nQ6nIMTxaZq/cMC4HNWrW
rhyOiExzDV03Dea+XBfAmB5tgGiiwUlysPzvQr9qfy35Vab2GuFsq62Zte3YbzGV
ukzEEIWhRWrhOfhiNTlDc1sy3HKqs2MjsWym5b6p8EdMoUnSoNVwMLJsHDYr6IZn
3lWYi1OU0JQeEBexzyfN7y1PxCPy1+8e+nhet2++XUn5oAtNfHEX+XNPvMFFuUFH
hYsCtcEIajE0d9dsnNRlkZ37p+uWzgzbS3va44EY9r/vYYQ6yB2Kot2fPBc7kQe8
1FivkNoo8eW4lvnCOXb8fFenewfwXBnlWq1A+FCDOQWYkzcPQDrqZ19xOtGTXyt6
aLZLEFv19C1yanURP0tYoTzLxegWs/A8ENLC7afGgxcixkT8un4OlOaaswV3XB+9
```

```
WYVx6iDmlqrrJNlzlCpraRMo5ixH3QtF9N3qlTpjWIixOHpzk5GzqQzS7ai2mZu/
b+bU9fzA8QaU0UL6EoCmkRXZch7LG+akITF1wmIMjMnms3JWPZuIpx0jFx0wKBQZ
atFAp2OMrGiqEEHE6gTzWxLf0mTXS63pMTIbKTF99yeUDR/xsQubXMIv4XFJnsP2
63+ZwU6YddxBrGkPOGy0J/Em2iSA89+OasxFnclmZPqkmCp9+doHJXIJ7Ya+R7QX
kqribuYeRqJsoYCTuN0Mnn3s+o2AUggTcNG2pkFfLU1fX1M7A9u6HKSl4eN/6CBv
9rDVpFTuqFvve2bdhMyZpFEr5AtTfC2FYf6aDi6v5+yIC7TKXNJ1Wrn1gxgcYjdn
1svGY3U0yh1ZZ/ENZlC1ZNTkWFcoXexHWFyK+sBAKnZY4QF0D113Z9P1n64aqqGB
YRrdsVrzidBibehMI5EGoAQRGk3jeo8nc4wM1KXSnxPU4ABW4AYF5pmai83pujwZ
aevG11LMltftvnBrXr1K78JrWLTq2/FGNQbODYmBwgnLiA1WYkQonbezO4g1LCu5
VzmJ2gVBaWVwWILq9zqWdt0628CICWd7p7AR/60xt8UKNeeUvcEN/kMh6M7p4bti
3WM2B4lfFfgMOCYJZxfOSC20+8GybEvjQVTRTcSavoZcC+xd6RWHhLZ7VqPi6n7L
eG6o9rZw/wF19gYppzcWvOz9hV+wq+AB3HxhcgomE1E5GLTVuZ5jvCc3VhXmtLYV
ijRekT6PQ/pQT6/5AaBnQ9UkvtJLJNIl9bqqUy4fUlfmJelC0mGggA==
-----END RSA PRIVATE KEY-----
```

3. You can now use this private key to generate a certificate for your CA. You will be prompted to write your private key password if the key is encrypted.

```
user@host:~$ openssl req -x509 -new -key atsl-ca.key -days 1826 -subj "/C=ES/ST=Catalonia/L=Barcelo-
na/O=ALBEDO Telecom SL/OU=Marketing/CN=ALBEDO Telecom Test/emailAddress=info@albedotelecom.com" -out at-
sl-ca.pem

Enter pass phrase for atsl-ca.key:
```

4. The output is again a text file you can display by typing this command.

```
user@host:~$ cat atsl-ca.pem
-----BEGIN CERTIFICATE-----
MIIENzCCAx+gAwIBAgIUMiIadTV//qn/9hRRhysk8O+R/2MwDQYJKoZIhvcNAQEL
BQAwgaoxCzAJBgNVBAYTAkVTMRIwEAYDVQQIDAlDYXRhbG9uaWExEjAQBgNVBAcM
CUJhcmNlbG9uYTEaMBgGA1UECgwRQUxCRURPIFRlbGVjb20gU0wxEjAQBgNVBAsM
CU1hcmtldGluZzEcMBoGA1UEAwwTQUxCRURPIFRlbGVjb20gVGVzdDElMCMGCSqG
SIb3DQEJARYWaW5mb0BhbGJlZG90ZWxlY29tLmNvbTAeFw0yNTA1MjAwOTA1MDla
Fw0zMDA1MjAwOTA1MDlaMIGqMQswCQYDVQQGEwJFUzESMBAGA1UECAwJQ2F0YWxv
bmlhMRIwEAYDVQQHDAlCYXJjZWxvbmExGjAYBgNVBAoMEUFMQkVETyBUZWxlY29t
IFNMMRIwEAYDVQQLDAlNYXJrZXRpbmcxHDAaBgNVBAMME0FMQkVETyBUZWxlY29t
IFRlc3QxJTAjBgkqhkiG9w0BCQEWFmluZm9AYWxiZWRvdGVsZWNvbS5jb20wggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC1D6j3ujQ7PklZRPxICJYu3zSD
ijvd/0k7+1sFQ6bnfPX/jcMy5Ubu72hQwCrDoXyXppPqhMS/H3thJM7iM1Ctw3sJ
+ptr6a4eHCBU4aQVsYPuObYYN9zOyP12joyHGlTUQS8G7A/Vjysx2XH9LQIu+aCp
3M2aTcrNfypc0SQP9Uc3rWYIpMgMxhWY9c+4JuaAuXgefGYVM0hgAySlxjK3K9eA
tUtFnQoI85QJQmre1//8Z+C/9DfYG2h+G+zptetU/8uRmg90FlCTXVWmJpfpxCv2
ccj6ffR9eBCy9DbT8D7JJYb0R0AJmos32t10/IzfbF67WRThP/nE2pGQ9yzDAgMB
AAGjUzBRMB0GA1UdDgQWBBRQ43EiJj7HwM5ZpzqM0hYzFV9SizAfBgNVHSMEGDAW
gBRQ43EiJj7HwM5ZpzqM0hYzFV9SizAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3
DQEBCwUAA4IBAQBZ97x82McRvyxPOsq6qw3xLnLsiJ6u5TNm9EwxOrMPAj7hUMo8
1ZOQwUbutPRoP5fH0hmthMyh76NIKY1nNwcJ4OkPDhJ7DGQ24ruXB2/LKxEP6yFt
XRdx+PoVYJBCc/LW5gKJazbpEe5Ib4c0ni84c9qTM2hxIyqFRyxlDtpiudlPrHD/
cRaS2oV2YxY8gqgsJ0yVJbnZ/o5WdR/EMD/vIzOwVDhTiRMBhd6gAWCa17qlFTRc
hFVcR6aS/qHe9mvvs9ysTFYkieq0DLwYGbxyOMRP0pK2qw2gnT59xTBBrdBJiNtD
KypQe4qW98Tby1Ss0dQRoorXQ8l9Ol9uOs8i
-----END CERTIFICATE-----
```

5. The previous command does not say much about the CA certificate structure. But you can get more details with a more specialized command.

```
user@host:~$ openssl x509 -noout -text -in atsl-ca.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            32:22:1a:75:35:7f:fe:a9:ff:f6:14:51:87:2b:24:f0:ef:91:ff:63
        Signature Algorithm: sha256WithRSAEncryption
       Issuer: C = ES, ST = Catalonia, L = Barcelona, O = ALBEDO Telecom SL, OU = Marketing, CN = ALBEDO
Telecom Test, emailAddress = info@albedotelecom.com
        Validity
            Not Before: May 20 09:05:09 2025 GMT
```

```
            Not After : May 20 09:05:09 2030 GMT
        Subject: C = ES, ST = Catalonia, L = Barcelona, O = ALBEDO Telecom SL, OU = Marketing, CN = ALBEDO
Telecom Test, emailAddress = info@albedotelecom.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b5:0f:a8:f7:ba:34:3b:3e:49:59:44:fc:48:08:
                    96:2e:df:34:83:8a:3b:dd:ff:49:3b:fb:5b:05:43:
                    a6:e7:7c:f5:ff:8d:c3:32:e5:46:ee:ef:68:50:c0:
                    2a:c3:a1:7c:97:a6:93:ea:84:c4:bf:1f:7b:61:24:
                    ce:e2:33:50:ad:c3:7b:09:fa:9b:6b:e9:ae:1e:1c:
                    20:54:e1:a4:15:b1:83:ee:39:b6:18:37:dc:ce:c8:
                    fd:76:8e:8c:87:1a:54:d4:41:2f:06:ec:0f:d5:8f:
                    2b:31:d9:71:fd:2d:02:2e:f9:a0:a9:dc:cd:9a:4d:
                    ca:cd:7f:2a:5c:d1:24:0f:f5:47:37:ad:66:08:a4:
                    c8:0c:c6:15:98:f5:cf:b8:26:e6:80:b9:78:1e:7c:
                    66:15:33:48:60:03:24:a5:c6:32:b7:2b:d7:80:b5:
                    4b:45:9d:0a:08:f3:94:09:42:6a:de:d7:ff:fc:67:
                    e0:bf:f4:37:d8:1b:68:7e:1b:ec:e9:b5:eb:54:ff:
                    cb:91:9a:0f:74:16:50:93:5d:55:a6:26:97:e9:c4:
                    2b:f6:71:c8:fa:7d:f4:7d:78:10:b2:f4:36:d3:f0:
                    3e:c9:25:86:f4:47:40:09:9a:8b:37:da:dd:74:fc:
                    8c:df:6c:5e:bb:59:14:e1:3f:f9:c4:da:91:90:f7:
                    2c:c3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                50:E3:71:22:26:3E:C7:C0:CE:59:A7:3A:8C:D2:16:33:15:5F:52:8B
            X509v3 Authority Key Identifier:
                keyid:50:E3:71:22:26:3E:C7:C0:CE:59:A7:3A:8C:D2:16:33:15:5F:52:8B

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         59:f7:bc:7c:d8:c7:11:bf:2c:4f:3a:ca:ba:ab:0d:f1:2e:72:
         ec:88:9e:ae:e5:33:66:f4:4c:31:3a:b3:0f:02:3e:e1:50:ca:
         3c:d5:93:90:c1:46:ee:b4:f4:68:3f:97:c7:d2:19:ad:84:cc:
         a1:ef:a3:48:29:8d:67:37:07:09:e0:e9:0f:0e:12:7b:0c:64:
         36:e2:bb:97:07:6f:cb:2b:11:0f:eb:21:6d:5d:17:71:f8:fa:
         15:60:90:42:73:f2:d6:e6:02:89:6b:36:e9:11:ee:48:6f:87:
         34:9e:2f:38:73:da:93:33:68:71:23:2a:85:47:2c:65:0e:da:
         62:b9:d9:4f:ac:70:ff:71:16:92:da:85:76:63:16:3c:82:a8:
         2c:27:4c:95:25:b9:d9:fe:8e:56:75:1f:c4:30:3f:ef:23:33:
         b0:54:38:53:89:13:01:85:de:a0:01:60:9a:d7:ba:a5:15:34:
         5c:84:55:5c:47:a6:92:fe:a1:de:f6:6b:ef:b3:dc:ac:4c:56:
         24:89:ea:b4:0c:bc:18:19:bc:72:38:c4:4f:d2:92:b6:ab:0d:
         a0:9d:3e:7d:c5:30:41:ad:d0:49:88:db:43:2b:2a:50:7b:8a:
         96:f7:c4:db:cb:54:ac:d1:d4:11:a2:8a:d7:43:c9:7d:3a:5f:
         6e:3a:cf:22
```

6. CRT files carry a serial number that is generated when the certificate is created. We need to create a file to set the serial number for the first CRT file generated by the CA. You need to worry only about the initial serial number. The CA will decide which serial number to use in subsequent CRT files. This file is going to be *atsl-ca.srl* in this example. The initial serial number is set to *1000*.

```
user@host:~$ echo 1000 > atsl-ca.srl
```

7. You need to specify a database file for the CA. This file stores historic details about generated CRT files. The database file is initially empty. The database file is a text file named *index.txt* in this example.

```
user@host:~$ touch index.txt
```

8. Finally, you need to generate a configuration file for the CA with information about how CRT files must be generated. In this example the configuration file will be named *atsl-ca.cfg*. The content of this file is described below:

```
[ca]
default_ca = CA_default

[CA_default]
database        = ./index.txt
serial          = ./atsl-ca.srl
new_certs_dir   = .
certificate     = atsl-ca.pem
private_key     = atsl-ca.key
default_days    = 365
default_md      = sha256
policy          = policy_any
copy_extensions = copy

[policy_any]
commonName              = supplied
stateOrProvinceName     = optional
countryName             = optional
organizationName        = optional
organizationalUnitName  = optional
emailAddress            = optional
```

Your CA is now ready to sign CSR files. It is important that the private key is kept in a safe location. Anyone with access to this file could use it to sign certifications on behalf or the organization represented by the CA.

In order to make any CRT file signed by your CA acceptable, you need to make the CA certificate a *Trusted Root Certification Authority* in the OS where the CRT are going to be used. For web sites based on HTTPS, the CA certificate must be included in any client accessing to the server. If you are using Microsoft Windows, you can import the CA certificate using the *Manage user certificates* application.

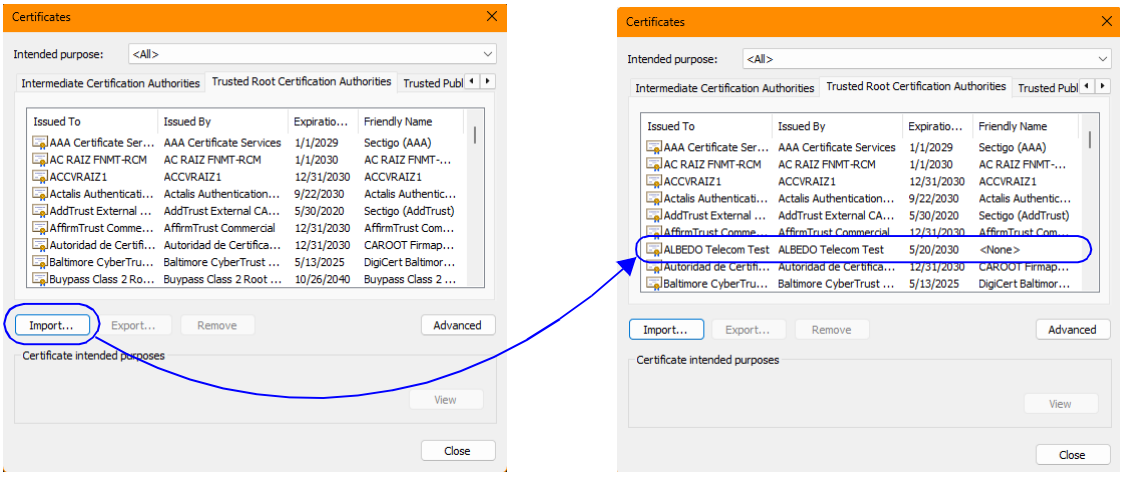

Figure 3 *Certificate import in Microsoft Windows.*

5. SIGNING A CERTIFICATE WITH YOUR CA

Once your CA has been set, you can use it to issue CRT files that will be valid within your organization or in any host that has installed the CA certificate as a Trusted Root Certification Authority. The correct procedure to generate a CRT file from a CSR is as follows

1. Generate the CRT from CSR file using the following command. Please, note that the CA key and the CA certificate, both needed to sign the CRT, are not specified in the command but they are described in the CA configuration file. If the CA private key is encrypted, the user will be prompted to enter the password for the key during the process. The same command will also ask for confirmation before signing the CSR and again before generating the CRT file.

```
user@host:~$ penssl ca -notext -config atsl-ca.cfg -in ntime-mkt-01.csr -out ntime-mkt-01.crt
Using configuration from atsl-ca.cfg
Enter pass phrase for atsl-ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :ASN.1 12:'CATALONIA'
localityName          :ASN.1 12:'BARCELONA'
organizationName      :ASN.1 12:'ALBEDO Telecom SL'
organizationalUnitName:ASN.1 12:'Marketing'
commonName            :ASN.1 12:'ntime.albedo.biz'
emailAddress          :IA5STRING:'info@albedotelecom.com'
Certificate is to be certified until Jul  7 08:55:07 2026 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

2. Like all other cryptographic objects generated by OpenSSL, the CRT is a text file you can display with the following command.

```
user@host:~$ cat ntime-mkt-01.crt
-----BEGIN CERTIFICATE-----
MIIEajCCA1KgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTAkVT
MRIwEAYDVQQIDAlDYXRhbG9uaWExEjAQBgNVBAcMCUJhcmNlbG9uYTEaMBgGA1UE
CgwRQUxCRURPIFRlbGVjb20gU0wxEjAQBgNVBAsMCU1hcmtldGluZzEcMBoGA1UE
AwwTQUxCRURPIFRlbGVjb20gVGVzdDElMCMGCSqGSIb3DQEJARYWaW5mb0BhbGJl
ZG90ZWxlY29tLmNvbTAeFw0yNTA3MDcwODU1MDdaFw0yNjA3MDcwODU1MDdaMIGT
MRkwFwYDVQQDDBBudGltZS5hbGJlZG8uYml6MRIwEAYDVQQIDAlDQVRBTE9OSUEx
CzAJBgNVBAYTAkVTMRowGAYDVQQKDBFBTEJFRE8gVGVsZWNvbSBTTDESMBAGA1UE
CwwJTWFya2V0aW5nMSUwIwYJKoZIhvcNAQkBFhZpbmZvQGFsYmVkb3RlbGVjb20u
Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7tGTWr3ts6YKChMI
1b2+9oN49Qd8R1YW4TIB8tEkvLi9uI9tT5AZHIZhbFIJKTzQRtgMiWN1N8DbL8/k
K9G/8fI+FujpWn7MOHx0wEpwzuBGygEZIpZUlGqMa+aKIJcztxXBbs5pfY+1a53F
sV7ybBK9X6eUQkZkJZ72/SQ9qX9MUHsqZc47vM/0h+08DUzdhOITwjcqeR+spzGr
EPhKFxlt5lmNL9I358kViOZpBRxP2SbBRaidullrtfiqIG8D58XtCfeCiQv6rgA7
2nGkGD68O+Xi4FSiGviiyebCJi47Lvqlohf4Z91l0toNyAwZ/zo4Qb2L/SaZj0MR
SvIZjQIDAQABo4GuMIGrMIGoBgNVHREEgaAwgZ2HBKwaA2SHBKwaA2eCB250aW1l
MDGCEm50aW1lMDEuYWxiZWRvLmJpeoIHbnRpbWUwMoISbnRpbWUwMi5hbGJlZG8u
Yml6ggdudGltZTAzghJudGltZTAzLmFsYmVkby5iaXqCB250aW1lMDSCEm50aW1l
MDQuYWxiZWRvLmJpeoIHbnRpbWUwNYISbnRpbWUwNS5hbGJlZG8uYml6MA0GCSqG
SIb3DQEBCwUAA4IBAQB5VGoEXTKsEhb9ztXu0AbhdE0TFaVPiQkriQmrUcahaNZl
gcCR1kGmkdEvTrAIajr6yZJlRTHieqxFAxvmGs3wbld4ZGZrY48U1gPAIcqH6WpF
eNt8QcuKhELN1zuByIjEj1rzpw1Ioh5VH4dWnReg+w5H5sFGIO90VM+hxljtO9Zz
9OqjYobW+lociEjRaehg5Xce8fL5u23M5QtY3YYea3JdaDcwmo7/5jkVRM4ltYlc
RmIjvujy1XZpn1T0ZhtALJJc6mHJF6W9WZCTpLkND0D6epW/1nSduwuNIVyV2ddD
IDH0IFfI4MzuOBgNlKtv0wo/oeTloxESaDaHz027
-----END CERTIFICATE-----
```

3. The structure of your certificate can be displayed with the following command:

```
user@host:~$ openssl x509 -noout -text -in ntime-mkt-01.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
```

```
       Issuer: C = ES, ST = Catalonia, L = Barcelona, O = ALBEDO Telecom SL, OU = Marketing, CN = ALBEDO
Telecom Test, emailAddress = info@albedotelecom.com
        Validity
            Not Before: Jul  7 08:55:07 2025 GMT
            Not After : Jul  7 08:55:07 2026 GMT
        Subject: CN = ntime.albedo.biz, ST = CATALONIA, C = ES, O = ALBEDO Telecom SL, OU = Marketing,
emailAddress = info@albedotelecom.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:ee:d1:93:5a:bd:ed:b3:a6:0a:0a:13:08:d5:bd:
                    be:f6:83:78:f5:07:7c:47:56:16:e1:32:01:f2:d1:
                    24:bc:b8:bd:b8:8f:6d:4f:90:19:1c:86:61:6c:52:
                    09:29:3c:d0:46:d8:0c:89:63:75:37:c0:db:2f:cf:
                    e4:2b:d1:bf:f1:f2:3e:16:e8:e9:5a:7e:cc:38:7c:
                    74:c0:4a:70:ce:e0:46:ca:01:19:22:96:54:94:6a:
                    8c:6b:e6:8a:20:97:33:b7:15:c1:6e:ce:69:7d:8f:
                    b5:6b:9d:c5:b1:5e:f2:6c:12:bd:5f:a7:94:42:46:
                    64:25:9e:f6:fd:24:3d:a9:7f:4c:50:7b:2a:65:ce:
                    3b:bc:cf:f4:87:ed:3c:0d:4c:dd:84:e2:13:c2:37:
                    2a:79:1f:ac:a7:31:ab:10:f8:4a:17:19:6d:e6:59:
                    8d:2f:d2:37:e7:c9:15:88:e6:69:05:1c:4f:d9:26:
                    c1:45:a8:9d:ba:59:6b:b5:f8:aa:20:6f:03:e7:c5:
                    ed:09:f7:82:89:0b:fa:ae:00:3b:da:71:a4:18:3e:
                    bc:3b:e5:e2:e0:54:a2:1a:f8:a2:c9:e6:c2:26:2e:
                    3b:2e:fa:a5:a2:17:f8:67:dd:65:d2:da:0d:c8:0c:
                    19:ff:3a:38:41:bd:8b:fd:26:99:8f:43:11:4a:f2:
                    19:8d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:ntime01, DNS:ntime01.albedo.biz, DNS:ntime02, DNS:ntime02.albedo.biz
    Signature Algorithm: sha256WithRSAEncryption
         79:54:6a:04:5d:32:ac:12:16:fd:ce:d5:ee:d0:06:e1:74:4d:
         13:15:a5:4f:89:09:2b:89:09:ab:51:c6:a1:68:d6:65:81:c0:
         91:d6:41:a6:91:d1:2f:4e:b0:08:6a:3a:fa:c9:92:65:45:31:
         e2:7a:ac:45:03:1b:e6:1a:cd:f0:6e:57:78:64:66:6b:63:8f:
         14:d6:03:c0:21:ca:87:e9:6a:45:78:db:7c:41:cb:8a:84:42:
         cd:d7:3b:81:c8:88:c4:8f:5a:f3:a7:0d:48:a2:1e:55:1f:87:
         56:9d:17:a0:fb:0e:47:e6:c1:46:20:ef:74:54:cf:a1:c6:58:
         ed:3b:d6:73:f4:ea:a3:62:86:d6:fa:5a:1c:88:48:d1:69:e8:
         60:e5:77:1e:f1:f2:f9:bb:6d:cc:e5:0b:58:dd:86:1e:6b:72:
         5d:68:37:30:9a:8e:ff:e6:39:15:44:ce:25:b5:89:5c:46:62:
         23:be:e8:f2:d5:76:69:9f:54:f4:66:1b:40:2c:92:5c:ea:61:
         c9:17:a5:bd:59:90:93:a4:b9:0d:0f:40:fa:7a:95:bf:d6:74:
         9d:bb:0b:8d:21:5c:95:d9:d7:43:20:31:f4:20:57:c8:e0:cc:
         ee:38:18:0d:94:ab:6f:d3:0a:3f:a1:e4:e5:a3:11:12:68:36:
         87:cf:4d:bb
```

Once the certificate has been enabled, it can be uploaded to Net.Time following the same procedure that for any other certificate. If the certificate is enabled in Net.Time, then it will be suppled in HTTPS sessions enabling any client connected to the web server to authenticate the session.