ownca

Release 0.4.0

Kairo de Araujo

Jan 09, 2023

CONTENTS

OwnCA makes easy handle Certificate Authority (CA) and manage certificates for hosts, servers or clients.

An example of high level usage:

```
>>> from ownca import CertificateAuthority
>>> ca = CertificateAuthority(ca_storage='/opt/CA', common_name='MyCorp CA')
>>> example_com = ca.issue_certificate('www.example.com', dns_names=['www.example.com',
↪'w3.example.com')
```

Basically in this three lines steps:

1. Imported the ownca Certificate Authority library
2. Created a new CA named as *Corp CA* that uses `/opt/CA` as CA storage for certificates, keys etc.
3. Create a signed certificates by *Corp CA* server *www.mycorp.com*, the files are also stored in `/opt/CA/certs//www.example.com`.

   ```
   >>> example_com.cert
   <Certificate(subject=<Name(CN=www.example.com)>, ...)>
   ```

CHAPTER

ONE

USAGE

1.1 Creating a Certificate Authority

The creation of a Certificate Authority (CA) is done by `class` CertificateAuthority.

Code example:

```
>>> from ownca import CertificateAuthority
>>> ca_corp = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA')
```

It will create the CA files in in `/opt/CA`.

1.1.1 Creating an Intermediate Certificate Authority

If a Certificate Authority (CA) needs to be Intermediate, it means the certificate needs to be signed by another CA, you can create that using the option `intermendiate=True`.

This action will generate only the Certificate Signing Request (CSR). Given the csr to the Root CA to be signed and having the certificate file, it needs to be added to the `ca_storage` folder as `ca.crt` and after that can be used.

Code example:

```
>>> from ownca import CertificateAuthority
>>> ica_corp = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA',␣
↪intermediate=True)
>>> ica_corp.csr_bytes
b'-----BEGIN CERTIFICATE REQUEST-----\
↪nMIICijCCAXICAQAwEjEQMA4GA1UEAwwHQ29ycCBDQTCCASIwDQYJKoZIhvcNAQEB\n
BQADggEPADCCAQoCggEBANErvwkteBXe0PybgWT7Su3Bduig/73Y75kEOzz+Ph4G\
↪nz3a4GEG6Gowgb5TXBpPMp6JVqo7uiSqpOV9f8SJW21CWCGu518Sit5BRFJ4wFf3P\
↪nzEtffb1i7fMr9H2JqjXVyQnVdrIAicWLJo3uF1P5RI5fm8tk5Cq1jRk/2CdfU3nP\
↪n6UANjoE9FAVT1tA2F84TVuGlKBXvsF8OJcCU+HoQhy9suMiTJikaK5Qeti+JBvrZ\
↪nfbijLk8L4u1cUYVVCAzFH+xtwg3TGeH02OmlybJKkm63cre4ixdSNm6AS+o456Mb\
↪nIKn8ksja7orH9lYyocxaitUax0b3iHNPsRFF/M0Q8XsCAwEAAaAzMDEGCSqGSIb3\
↪nDQEJDjEkMCIwEgYDVR0RBAswCYIHQ29ycCBDQTAMBgNVHRMEBTADAQH/MA0GCSqG\
↪nSIb3DQEBCwUAA4IBAQAu9OYSeZMrJZFXrBLqdv60STmyRx+s2/7cq9khOMdayItu\n/
↪kUAw0EIEoB3+uCRm4tvRrZeK2rgDKp4InyJ3cCPMcU02H84OOHen1V3H9WWUEBP\nuxkecQiFpGLzj/
↪gisFjqGOuV/PzeuB/VhfiCJm7tG0PVK9n/JzZ1WBVL9u3GxDHY\n37328J7GniD4XDidevMY/3Gq+lZI9X/
↪OHMSIMh2Q12FG/Ol8mBVdksp4gDbNs98D\nctzfHrmGBTF/f94JX/
↪p94xerjp3NvcAIkzrm9Tfa05BDfpq8RsGgvPAZo4S8Hphz\
↪nKHokUqabqsIC76VBMDFTb6GU3Vv80nBYTN+LrXmr\n-----END CERTIFICATE REQUEST-----\n'
```

Note: Note that this Intermediate CA is not ready to be used, certificate file is missing.

```
>>> ica_corp.issue_certificate('qa.dev.ownca.org')
Traceback (most recent call last):
...
ownca.exceptions.OwnCAIntermediate: Intermediate Certificate Authority has not a signed␣
↪certificate file in CA Storage
```

Is necessary get the certificate signed from the CA to have this Intermediate CA ready. Add the certificate to `ca_storage` folder as `ca.crt`.

1.1.2 Available methods

The Certificate Authority has built in methods such as

- common_name
- cert
- cert_bytes
- certificates
- csr
- csr_bytes
- key
- key_bytes
- public_key
- public_key_bytes
- hash_name
- issue_certificate
- revoke_certificate
- status
- sign_csr

See CertificateAuthority for more details.

Code Example:

```
>>> ca_corp.cert
<Certificate(subject=<Name(CN=Corp CA)>, ...)>
>>> ca_corp.cert_bytes
b'-----BEGIN CERTIFICATE-----\
↪nMIIC2TCCAcGgAwIBAgIUXn4msF6ONA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
↪nBQAwEjEQMA4GA1UEAwwHQ29ycCBDQTAeFw0yMDA0MjcxODA0MjBaFw0yMjA4MDEx\
↪nODA0MjBaMBIxEDAOBgNVBAMMB0NvcnAgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB\
↪nDwAwggEKAoIBAQC8JqeBHwVnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
↪norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+UlLU7XV3+TghiaHVsd4lVz7\
↪nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kxlHidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/
↪sOndEjVquYipOQXIZwRJ4ZXZ29K4UdoW+9iMCvhtVPCHz4FEl\nPBFn2vuqRg13EcZ6X3/
↪83VJaO5TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfxc\
↪nEPkffyHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAmC\
↪nB0NvcnAgQ0EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAZyMd\
↪n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFX1zKI/8NG1QrAEwG1Rai2yTU/\
↪n07s5XBRwGIcRuFC1tcT7oqAjHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
↪nlS5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4KwlBQISxYi5u9pSwCum+gIS\
```

(continues on next page)

(continued from previous page)

1.2 Loading a existent Certificate Authority

In the same way if the /opt/CA exists and the file is there, it will load and it does not overwrite the files.

Code example:

```
>>> from ownca import CertificateAuthority
>>> ca_corp = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA')
>>> ca_corp.cert
<Certificate(subject=<Name(CN=Corp CA)>, ...)>
>>> ca_corp.key_bytes
b'-----BEGIN CERTIFICATE-----\
↪nMIIC2TCCAcGgAwIBAgIUXn4msF6ONA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
↪nBQAwEjEQMA4GA1UEAwwHQ29ycCBDQTAeFw0yMDA0MjcxODA0MjBaFw0yMjA4MDEx\
↪nODA0MjBaMBIxEDAOBgNVBAMMB0NvcnAgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB\
↪nDwAwggEKAoIBAQC8JqeBHwVnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
↪norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+UlLU7XV3+TghiaHVsd4lVz7\
↪nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kxlHidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/
↪sOndEjVquYipOQXIZwRJ4ZXZ29K4UdoW+9iMCvhtVPCHz4FEl\nPBFn2vuqRg13EcZ6X3/
↪83VJaO5TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfxc\
↪nEPkffyHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAmC\
↪nB0NvcnAgQ0EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAZyMd\
↪n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFX1zKI/8NG1QrAEwG1Rai2yTU/\
↪n07s5XBRwGIcRuFC1tcT7oqAjHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
↪nlS5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4KwlBQISxYi5u9pSwCum+gIS\
↪nx2+Vc7jJGCUEP1iMLPuxpOHIns9FusfzPfRfApFQRqZfxBO2Hpewoj1pbb6HckAJ\
↪nVlOyV5KcAunC9UsUtliwN3eFef+U/tNakYtcZjzqn1R5hlLBfaENCwdG4pdvuFw7\na/
↪a5r9CF+SDw0tldZw==\n-----END CERTIFICATE-----\n'
>>>
>>> load_ca = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA')
>>> load_ca.cert
<Certificate(subject=<Name(CN=Corp CA)>, ...)>
>>> load_ca.key_bytes
b'-----BEGIN CERTIFICATE-----\
↪nMIIC2TCCAcGgAwIBAgIUXn4msF6ONA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
↪nBQAwEjEQMA4GA1UEAwwHQ29ycCBDQTAeFw0yMDA0MjcxODA0MjBaFw0yMjA4MDEx\
↪nODA0MjBaMBIxEDAOBgNVBAMMB0NvcnAgQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB\
↪nDwAwggEKAoIBAQC8JqeBHwVnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
↪norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+UlLU7XV3+TghiaHVsd4lVz7\
↪nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kxlHidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/
↪sOndEjVquYipOQXIZwRJ4ZXZ29K4UdoW+9iMCvhtVPCHz4FEl\nPBFn2vuqRg13EcZ6X3/
↪83VJaO5TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfxc\
↪nEPkffyHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAmC\
↪nB0NvcnAgQ0EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAZyMd\
↪n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFX1zKI/8NG1QrAEwG1Rai2yTU/\
↪n07s5XBRwGIcRuFC1tcT7oqAjHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
↪nlS5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4KwlBQISxYi5u9pSwCum+gIS\
↪nx2+Vc7jJGCUEP1iMLPuxpOHIns9FusfzPfRfApFQRqZfxBO2Hpewoj1pbb6HckAJ\
↪nVlOyV5KcAunC9UsUtliwN3eFef+U/tNakYtcZjzqn1R5hlLBfaENCwdG4pdvuFw7\na/
↪a5r9CF+SDw0tldZw==\n-----END CERTIFICATE-----\n'
```

(continues on next page)

(continued from previous page)

1.3 Multiple Certificate Authorities

Just use different `ca_storage` and you can have/manage multiple CAs

Code example:

```
>>> from ownca import CertificateAuthority
>>> ca_corp = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA')
>>> ca_edu = CertificateAuthority(ca_storage='/opt/edu_CA', common_name='Edu CA')
>>> ca_edu.cert
<Certificate(subject=<Name(CN=Edu CA)>, ...)>
>>> ca_corp.cert
<Certificate(subject=<Name(CN=Corp CA)>, ...)>
```

1.4 Issuing certificate

To issue a new certificate, you need use an existent instance of `class` CertificateAuthority and use the `function` issue_certificate().

Code example:

```
>>> from ownca import CertificateAuthority
>>> ca_corp = CertificateAuthority(ca_storage='/opt/corp_CA', common_name='Corp CA')
>>> example_com = ca_corp.issue_certificate("www.example.com", dns_names=["www.example.
↪com", "w3.example.com"], oids={"country_name": "BR", "locality_name": "Uba"})
```

1.4.1 Available methods

The Certificate Authority has built in methods such as

- common_name
- cert
- cert_bytes
- csr
- csr_bytes
- key
- key_bytes
- public_key
- public_key_bytes
- revoked

See HostCertificate for more details.

Checking the certificate

```
>>> example_com.cert
<Certificate(subject=<Name(C=BR,L=Uba,CN=www.example.com)>, ...)>
```

1.5 Loading host/client certificate

Same as the CA, if you use an existent certificate, it will be loaded and not overwrited.

Example:

```
>>> load_cert = ca_corp.load_certificate("www.example.com")
>>> load_cert.cert == example_com.cert
True
```

CHAPTER

TWO

THE MOTIVATION

The ownca was created in 2017 as a group of scripts to manage certificates, in 2018 it was moved to a very simple library (mostly hardcoded actions) and now 2019 was decide to open and be a library that could help others.

Basically, OwnCA uses the powerful library http://cryptography.io .

2.1 ownca package

2.1.1 Submodules

2.1.2 ownca.exceptions module

Copyright (c) 2018-2020 Kairo de Araujo

exception ownca.exceptions.**OwnCAFatalError**

Bases: Exception

No controlled Error, fatal error

exception ownca.exceptions.**OwnCAInconsistentData**

Bases: Exception

Certificate file is inconsistent.

exception ownca.exceptions.**OwnCAIntermediate**

Bases: Exception

CA is a Intermediate Certificate Authority missing certificate file

exception ownca.exceptions.**OwnCAInvalidCertificate**

Bases: Exception

The certificate is invalid or not found

exception ownca.exceptions.**OwnCAInvalidDataStructure**

Bases: Exception

Invalid Ownca Data Structure.

exception ownca.exceptions.**OwnCAInvalidFiles**

Bases: Exception

CA Files are inconsistent.

exception ownca.exceptions.**OwnCAInvalidOID**

Bases: Exception

Invalid OID

2.1.3 ownca.ownca module

Copyright (c) 2018-2022 Kairo de Araujo

class ownca.ownca.**CertificateAuthority**(*ca_storage=None*, *common_name=None*, *intermediate=False*, *maximum_days=825*, ***kwargs*)

Bases: object

The primary Python OWNCA class.

This class initializes the Certificate Authority (CA).

Parameters

- **ca_storage** (*str, required when there is no CA*) – path where CA files and hosts files are stored. Default is the current directory (os.getcwd())
- **common_name** (*str, required when there is no CA*) – Common Name for CA
- **dns_names** (*list of strings, optional*) – List of DNS names
- **intermediate** (*bool, default False*) – Intermediate Certificate Authority mode
- **oids** (*dict, optional, all keys are optional*) – CA Object Identifiers (OIDs). The are typically seen in X.509 names. Allowed keys/values: 'country_name': str (two letters), 'locality_name': str, 'state_or_province': str, 'street_address': str, 'organization_name': str, 'organization_unit_name': str, 'email_address': str,
- **public_exponent** (*int, default: 65537*) – Public Exponent
- **key_size** (*int, default: 2048*) – Key size

property cert

Get CA certificate

Returns
certificate class

Return type
class, cryptography.hazmat.backends.openssl.x509.Certificate

property cert_bytes

Get CA certificate in bytes

Returns
certificate

Return type
bytes,

property certificates

Get the CA list of issued/managed certificates

Returns
List of certificates (default is host/domain)

Return type
list

property common_name

Get CA common name

Returns
CA common name

Return type
str

property crl

Get CA certificate revocation list (crl)

Returns
certificate class

Return type
class, cryptography.hazmat.backends.openssl.x509._CertificateRevocationList

property crl_bytes

Get CA certificate revocation list (crl)

Returns
certificate class

Return type
bytes

property csr

Get CA Certificate Signing Request

Returns
certificate class

Return type
class, cryptography.hazmat.backends.openssl.x509._CertificateSigningRequest

property csr_bytes

Get CA Certificate Signing Request in bytes

Returns
certificate class

Return type
bytes

property hash_name

Get the CA hash name

Returns
CA hash name

Return type
str

initialize(*common_name=None*, *dns_names=None*, *intermediate=False*, *maximum_days=825*, *public_exponent=65537*, *key_size=2048*)

Initialize the Certificate Authority (CA)

Parameters

- **common_name** (*str, required*) – CA Common Name (CN)
- **dns_names** (*list of strings, optional*) – List of DNS names
- **maximum_days** (*int, default: 825*) – Certificate maximum days duration
- **public_exponent** (*int, default: 65537*) – Public Exponent
- **intermediate** (*bool, default False*) – Intermediate Certificate Authority mode
- **key_size** (*int, default: 2048*) – Key size

Returns
tuple with CA certificate, CA Key and CA Public key

Return type
tuple (`cryptography.x509.Certificate`, `cryptography.hazmat.backends.openssl.rsa`, string public key)

issue_certificate(*hostname*, *maximum_days=825*, *common_name=None*, *dns_names=None*, *oids=None*, *public_exponent=65537*, *key_size=2048*, *ca=True*)

Issues a new certificate signed by the CA

Parameters

- **hostname** (*str, required*) – Hostname
- **maximum_days** (*int, default: 825*) – Certificate maximum days duration
- **common_name** (*str, optional*) – Common Name (CN) when loading existent certificate
- **dns_names** (*list of strings, optional*) – List of DNS names
- **oids** (*dict, optional, all keys are optional*) – CA Object Identifiers (OIDs). The are typically seen in X.509 names. Allowed keys/values: `'country_name': str (two letters)`, `'locality_name': str`, `'state_or_province': str`, `'street_address': str`, `'organization_name': str`, `'organization_unit_name': str`, `'email_address': str`,
- **public_exponent** (*int, default: 65537*) – Public Exponent
- **key_size** (*int, default: 2048*) – Key size
- **ca** (*bool, default True.*) – Certificate is CA or not.

Returns
host object

Return type
`ownca.ownca.HostCertificate`

property key

Get CA RSA Private key

Returns
RSA Private Key class

Return type
class, `cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey`

property key_bytes

Get CA RSA Private key in bytes

Returns
RSA Private Key

Return type
bytes

`load_certificate`(*hostname*)

Loads an existent certificate.

Parameters
hostname (*str, required*) – Hostname (common name)

Returns
host object

Return type
ownca.ownca.HostCertificate

`property public_key`

Get CA RSA Public key

Returns
RSA Public Key class

Return type
class, cryptography.hazmat.backends.openssl.rsa._RSAPublicKey

`property public_key_bytes`

Get CA RSA Public key in bytes

Returns
RSA Public Key class

Return type
bytes

`revoke_certificate`(*hostname*, *common_name=None*)

Revokes an existent certificate owned by CA. It also updates the CA Certificate Revoked List.

Parameters

- **hostname** (*str, required*) – Hostname
- **common_name** (*str, optional*) – Common Name (CN) when loading existent certificate

Returns
CA object

Return type
ownca.ownca.CertificateAuthority

`sign_csr`(*csr*, *csr_public_key*, *maximum_days=825*)

Signs an Certificate Sigining Request and generates the certificates.

Parameters

- **hostname** (*str, required*) – Hostname
- **csr** – Certificate Signing Request Object
- **csr** – class, cryptography.hazmat.backends.openssl.x509._CertificateSigningRequest
- **maximum_days** (*int, default: 825*) – Certificate maximum days duration

Return type
class, cryptography.hazmat.backends.openssl.rsa._RSAPublicKey

Returns
host object

Return type
ownca.ownca.CertificateAuthority

property status

This method give the CA storage status

Returns
dict ownca.utils.ownca_directory

```
{
    "type": "Certificate Authority" or
        "Intermediate Certificate Authority",
    "certificate": bool,
    "crl": bool,
    "csr": bool,
    "key": bool,
    "public_key": bool,
    "ca_home": None or str,
}
```

property type

This method give the Certificate Authority type 'Certificate Authority' or 'Intermediate Certificate Authority'

Returns
str

class ownca.ownca.**HostCertificate**(*common_name*, *files*, *cert_data*)

Bases: object

This class provide the host certificate methods.

Parameters

- **common_name** (*str, required*) – Host CN (Common Name), FQDN standard is required.
- **files** (*dict, required*) – files path (certificate, key and public key) from host

```
{
    "certificate": str,
    "key": str,
    "public_key": str,
}
```

- **cert_data** (*object, required*) – certificate data ownca.OwncaCertData

property cert

Get certificate

Returns
certificate object

Return type
object, cryptography.hazmat.backends.openssl.x509.Certificate

property cert_bytes

Get certificate in bytes

Returns
certificate

Return type
bytes,

property common_name

Get common name

Returns
common name

Return type
str

property csr

Get Certificate Signing Request

Returns
certificate class

Return type
class, cryptography.hazmat.backends.openssl.x509._CertificateSigningRequest

property csr_bytes

Get Certificate Signing Request in bytes

Returns
certificate class

Return type
bytes

property key

Get RSA Private key

Returns
RSA Private Key class

Return type
object, cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey

property key_bytes

Get RSA Private key in bytes

Returns
RSA Private Key

Return type
bytes

property public_key

Get RSA Public key

Returns
RSA Public Key class

Return type
object, cryptography.hazmat.backends.openssl.rsa._RSAPublicKey

property public_key_bytes

Get RSA Public key in bytes

Returns
RSA Public Key class

Return type
bytes

property revoked

Get revoked state

Returns
True when revoked and False when valid.

Return type
str

class ownca.ownca.**OwncaCertData**(*data*)

Bases: object

Generates Ownca Certificate Data Structure

Parameters
data (*dict*) – Certificate Data

```
{
   "cert": cryptography.x509.Certificate,
   "cert_bytes": bytes,
   "csr": ``cryptography.x509._CertificateSigningRequest``
   "csr_bytes: bytes,
   "key": cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey,
   "key_bytes": bytes,
   "public_key":
    cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey,
   "public_key_bytes": bytes,
   "crl":
    cryptography.hazmat.backends.openssl.rsa._RSAPublicKey,
    "crl_bytes": bytes
}
```

Returns
OwncaCertData

Return type
ownca.ownca.OwncaCertData

Raises
exceptions.OwnCAInvalidDataStructure

property cert

Method to get the certificate

Returns
certificate

Return type
cryptography.x509.Certificate

property cert_bytes

Method to get the certificate in bytes

Returns
certificate

Return type
bytes

property crl

Method to get the certificate revocation list (crl)

Returns
certificate revocation list (crl)

Return type

cryptography.hazmat.backends.openssl.x509. _CertificateRevocationList

property crl_bytes

Method to get the certificate revocation list (crl)

Returns
certificate revocation list (crl)

Return type
bytes

property csr

Method to get the certificate signing request if an Intermediate CA

Returns
csr

Return type
cryptography.x509._CertificateSigningRequest

property csr_bytes

Method to get the certificate signing request in bytes

Returns
csr

Return type
bytes

property key

Method to get the key

Returns
key

Return type
cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey

property key_bytes

Method to get the key in bytes

Returns
key

Return type
bytes

property public_key

Method to get the public key

Returns
key

Return type
cryptography.hazmat.backends.openssl.rsa._RSAPublicKey

property public_key_bytes

Method to get the public key in bytes

Returns
public key

Return type
bytes

ownca.ownca.format_oids(*oids_parameters*)

Format dictionary OIDs to cryptography.x509.oid.NameOID object list

Parameters
oids_parameters (*dict, required*) – CA Object Identifiers (OIDs). The are typically seen in X.509 names. Allowed keys/values: 'country_name': str (two letters), 'locality_name': str, 'state_or_province': str, 'street_address': str, 'organization_name': str, 'organization_unit_name': str, 'email_address': str,

Returns
cryptography.x509.oid.NameOID object list

Return type
object cryptography.x509.oid.NameOID object list

ownca.ownca.load_cert_files(*common_name*, *key_file*, *public_key_file*, *csr_file*, *certificate_file*, *crl_file*)

Loads the certificate, keys and revoked list files from storage

Parameters

- **common_name** (*str, required when there is no CA*) – Common Name for CA
- **key_file** (*str, required*) – key file full path
- **public_key_file** (*str, required*) – public key file full path
- **csr_file** (*str, required*) – certificate signing request file full path
- **certificate_file** (*str, required*) – certificate file full path
- **crl_file** – certificate revocation list file full path

Returns
OwncaCertData

Raises
OwnCAInconsistentData

2.1.4 ownca.utils module

Copyright (c) 2018-2022 Kairo de Araujo

class ownca.utils.**CAStatus**(*ca_type_intermediate: bool = False*, *ca_home: str = ''*, *certificate: bool = False*, *crl: bool = False*, *csr: bool = False*, *key: bool = False*, *public_key: bool = False*)

Bases: object

ca_home: str = ''

ca_type_intermediate: bool = False

certificate: bool = False

crl: bool = False

csr: bool = False

key: bool = False

public_key: bool = False

ownca.utils.**file_data_status**(*ca_status: CAStatus*) → Optional[bool]

Verify the CA status based in the existent files.

Parameters
ca_status (*CAStatus, required*) – current ca_status file dictionary: ownca.utils.ownca_directory

Returns
True, False or None

Return type
bool or None

ownca.utils.**ownca_directory**(*ca_storage: str*) → *CAStatus*

Validates and manage CA storage directory and subfolders structure files.

Parameters
ca_storage (*string, required*) – CA storage

Returns
dict with state of ownca storage files

Return type
CAStatus

ownca.utils.**store_file**(*file_data: bytes*, *file_path: str*, *force: bool*, *permission: Optional[int]*) → bool

Stores (write) files in the storage

Parameters

- **file_data** (*str, required*) – the file data
- **file_path** (*str, required*) – the file absolute path
- **permission** (*int, optional*) – operating-system mode bitfield

Returns
bool

Return type
boolean

ownca.utils.**validate_hostname**(*hostname: str*) → bool

Validates if the hostname follows the common Internet rules for FQDN

Parameters
hostname (*sting, required*) – string hostname

Returns
bool

Return type
bool

2.1.5 Module contents

Copyright (c) 2018, 2019, 2020 Kairo de Araujo

2.2 ownca.crypto package

2.2.1 Submodules

2.2.2 ownca.crypto.cert module

2.2.3 ownca.crypto.keys module

Copyright (c) 2020 Kairo de Araujo

class ownca.crypto.keys.**OwncaKeyData**(*key_data*)

Bases: object

Generates Ownca Key Data Structure

Parameters
key_data (*dict*) – Key Data

```
{
    "key": cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey,
    "key_bytes": bytes,
    "public_key":
        cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey,
    "public_key_bytes": bytes,
}
```

Returns
OwncaKeyData

Return type
ownca.crypto.keys.OwncaKeyData

Raises
OwnCAInvalidDataStructure

property key

Method to get the key

Returns
key

Return type
cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey

property key_bytes

Method to get the key in bytes

Returns
key

Return type
bytes

property public_key

Method to get the public key

Returns
key

Return type
cryptography.hazmat.backends.openssl.rsa._RSAPublicKey

property public_key_bytes

Method to get the public key in bytes

Returns
public key

Return type
bytes

ownca.crypto.keys.**generate**(*public_exponent=65537*, *key_size=2048*)

Generates Private and Public keys

Parameters

- **public_exponent** (*int, optional, Default: 65537*) – Public Exponent
- **key_size** (*int, optional, Default: 2048*) – Key size

Returns
Ownca Key Data Structure

Return type
ownca.crypto.keys.OwncaKeyData

2.2.4 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

P

R

S

T

V