

---

**ownca**

***Release 0.4.0***

**Kairo de Araujo**

**Mar 06, 2023**



# CONTENTS

- 1 Usage 3**
  - 1.1 Creating a Certificate Authority . . . . . 3
  - 1.2 Loading a existent Certificate Authority . . . . . 5
  - 1.3 Multiple Certificate Authorities . . . . . 6
  - 1.4 Issuing certificate . . . . . 6
  - 1.5 Loading host/client certificate . . . . . 7
- 2 The motivation 9**
  - 2.1 ownca package . . . . . 9
  - 2.2 ownca.crypto package . . . . . 20
- 3 Indices and tables 23**
- Python Module Index 25**
- Index 27**



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)>, ...)>
```



## 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 `intermediate=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-----\n
↳ nMIICi jCCAXICAQAwEjEQMA4GA1UEAwHQ29ycCBDQTCCASIwDQYJKoZIhvcNAQEB\n
BQADggEPADCCAQoCggEBANervwkteBXe0PybgWT7Su3Bduig/73Y75kEOzz+Ph4G\n
↳ nz3a4GEG6Gowgb5TXBpPMp6JVqo7uiSqP0V9f8SJW21CWCGu518Sit5BRFJ4wFf3P\n
↳ nzEtffbi7fMr9H2JqjXVyQnVdrIAicWLJo3uF1P5RI5fm8tk5Cq1jRk/2CdfU3nP\n
↳ n6UANjoE9FAVT1tA2F84TVuGlKBXvsF80JcCU+HoQhy9suMiTJikaK5Qeti+JBvrZ\n
↳ nfbijLk8L4u1cUYVVCazFH+xtwg3TGeH020mlybJKkm63cre4ixdSNm6AS+o456Mb\n
↳ nIKn8ksja7orH9lYyocxaitUax0b3iHNPsRFF/M0Q8XsCAwEAAaAzMDEGCSqGSib3\n
↳ nDQEJDjEkMCIEgYDVR0RBASwCYIHQ29ycCBDQTAMBgNVHRMEBTADAQH/MA0GCSqG\n
↳ nSib3DQEBcwUAA4IBAQAu90YSeZMrJZFxrBLqdv60STmyRx+s2/7cq9khOMdayItu\n
↳ kUAW0EIEoB3+uCRm4tvRrZeK2rgDKp4InyJ3cCPMcU02H8400Hen1V3H9WWUEBP\n
↳ nuxkecQiFpGLzj/\n
↳ gisFjqG0uV/PzeuB/VhfiCJm7tG0PVK9n/JzZ1WBVL9u3GxDHY\n37328J7GniD4XDidevMY/3Gq+lZI9X/\n
↳ OHMSIMh2Q12FG/0l8mBVDksp4gDbNs98D\nctzfHrmGBTF/f94JX/\n
↳ p94xerjp3NvcAIkzrm9Tfa05BDfpq8RsGgvPAZo4S8Hphz\n
↳ nKHokUqabq5IC76VBMDFTb6GU3Vv80nBYTN+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-----\
↳ nMIIC2TCCAcGgAwIBAgIUxn4msF60NA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
↳ nBQAwEjEQA4GA1UEAwHQ29ycCBDQTAeFw0yMDA0Mjc0ODA0MjBaFw0yMjA4MDEx\
↳ nODA0MjBaMBIxEDAOBgNVBAMMB0NvcnAgQ00EwggEiMA0GCSqGSIb3DQEBAQUAA4IB\
↳ nDwAwggEKAoIBAQC8JqeBHWnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
↳ norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+U1LU7XV3+TghiaHVsd4lVz7\
↳ nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kx1HidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/\
↳ sOndEjVquYip00XIZwRl4ZXZ29K4UdoW+9iMCvhtVPChz4FEl\nnPBFn2vuqRg13EcZ6X3/\
↳ 83VJa05TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfxc\
↳ nEPkffYHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAMC\
4nB0NvcnAgQ00EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAZyMd\
↳ n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFx1zKI/8NG1QrAEwG1Rai2yTU/\
↳ n07s5XBRwGICRuFC1tcT7oqAJHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
↳ n1S5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4Kw1BQISxYi5u9pSwCum+gIS\
↳ n23HcZi1GCUED1iMLBumQWJca0Fua6rBcD5AnE0Pa76rBQ2Hrcmci1abbGhshA1\
(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-----\
    ↪nMIIC2TCCAcGgAwIBAgIUxn4msF60NA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
    ↪nBQAwEjEQA4GA1UEAwHQ29ycCBDQTAeFw0yMDA0Mjc0ODA0MjBaFw0yMjA4MDEx\
    ↪nODA0MjBaMBIxEDA0BgNVBAMMB0NvcnAgQ0EwggEiMA0GCSqGSIB3DQEBAQUAA4IB\
    ↪nDwAwggEKAAoIBAQC8JqeBHwVnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
    ↪norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+U1LU7XV3+TghiaHVsd4lVz7\
    ↪nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kxlHidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/
    ↪sOndEjVquYip0QXIZwRJ4ZXZ29K4UdoW+9iMCvhtVPCHz4FE1\nPBFn2vuqRg13EcZ6X3/
    ↪83VJa05TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfx\
    ↪nEPkffYHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAMC\
    ↪nB0NvcnAgQ0EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCQAQEAZyMd\
    ↪n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFX1zKI/8NG1QrAEwG1Rai2yTU/\
    ↪n07s5XBRwGICRuFC1tcT7oqAjHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
    ↪n1S5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4KwLBQISxYi5u9pSwCum+gIS\
    ↪nx2+Vc7jJGCUeP1iMLPuxpOHIns9FusfzPfrfApFQRqZfxB02Hpewoj1pbb6HckAJ\
    ↪nV10yV5KcAunC9UsUtlwN3eFef+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-----\
    ↪nMIIC2TCCAcGgAwIBAgIUxn4msF60NA8lWcehVqd1xxdRvYkwDQYJKoZIhvcNAQEL\
    ↪nBQAwEjEQA4GA1UEAwHQ29ycCBDQTAeFw0yMDA0Mjc0ODA0MjBaFw0yMjA4MDEx\
    ↪nODA0MjBaMBIxEDA0BgNVBAMMB0NvcnAgQ0EwggEiMA0GCSqGSIB3DQEBAQUAA4IB\
    ↪nDwAwggEKAAoIBAQC8JqeBHwVnmJkeOKLwqMcil/nY4QBLDsAg4LKhhzFAB/SvJ16F\
    ↪norqip2jLuRhpxrPNUYa9p8+ZPZziAL7ir68csnJI+U1LU7XV3+TghiaHVsd4lVz7\
    ↪nHBRhMLQcFQvnEyC5sfm84fptetlL4HN8jJUda/M26kxlHidJRCL221R9g+/RI113\n73tBX7iZSAcBTv/
    ↪sOndEjVquYip0QXIZwRJ4ZXZ29K4UdoW+9iMCvhtVPCHz4FE1\nPBFn2vuqRg13EcZ6X3/
    ↪83VJa05TSh7Qzl87MVmfBtGBWvib5gXxPEY1zOnhojfx\
    ↪nEPkffYHauwyORFkpaE00LkrkNjxNEQ5qhCKHAgMBAAGjJzAlMBIGA1UdEQQLMAMC\
    ↪nB0NvcnAgQ0EwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCQAQEAZyMd\
    ↪n5eu76geBT8yobTyovhPUq63+9BWvmUViNhukZSFX1zKI/8NG1QrAEwG1Rai2yTU/\
    ↪n07s5XBRwGICRuFC1tcT7oqAjHYDQw+3RgYYd+isPUo3Mi7SSWQYpJWmk7ICmqYzy\
    ↪n1S5uk4iZatPWFVwL4XcH9ssgTVTK3kIdG9LKPPz/4KwLBQISxYi5u9pSwCum+gIS\
    ↪nx2+Vc7jJGCUeP1iMLPuxpOHIns9FusfzPfrfApFQRqZfxB02Hpewoj1pbb6HckAJ\
    ↪nV10yV5KcAunC9UsUtlwN3eFef+U/tNakYtcZjzqn1R5hlLBfaENCwdG4pdvuFw7\na/
    ↪a5r9CF+SDw0tldZw==\n-----END CERTIFICATE-----\n'
```

(continues on next 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 overwritten.

Example:

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



## 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 Signing 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** (*string, 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



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### O

- `ownca`, [20](#)
- `ownca.crypto`, [21](#)
- `ownca.crypto.keys`, [20](#)
- `ownca.exceptions`, [9](#)
- `ownca.ownca`, [10](#)
- `ownca.utils`, [19](#)



## C

ca\_home (*ownca.utils.CAStatus* attribute), 19  
 ca\_type\_intermediate (*ownca.utils.CAStatus* attribute), 19  
 CAStatus (*class* in *ownca.utils*), 19  
 cert (*ownca.ownca.CertificateAuthority* property), 10  
 cert (*ownca.ownca.HostCertificate* property), 14  
 cert (*ownca.ownca.OwncaCertData* property), 16  
 cert\_bytes (*ownca.ownca.CertificateAuthority* property), 10  
 cert\_bytes (*ownca.ownca.HostCertificate* property), 14  
 cert\_bytes (*ownca.ownca.OwncaCertData* property), 17  
 certificate (*ownca.utils.CAStatus* attribute), 19  
 CertificateAuthority (*class* in *ownca.ownca*), 10  
 certificates (*ownca.ownca.CertificateAuthority* property), 10  
 common\_name (*ownca.ownca.CertificateAuthority* property), 11  
 common\_name (*ownca.ownca.HostCertificate* property), 15  
 crl (*ownca.ownca.CertificateAuthority* property), 11  
 crl (*ownca.ownca.OwncaCertData* property), 17  
 crl (*ownca.utils.CAStatus* attribute), 19  
 crl\_bytes (*ownca.ownca.CertificateAuthority* property), 11  
 crl\_bytes (*ownca.ownca.OwncaCertData* property), 17  
 csr (*ownca.ownca.CertificateAuthority* property), 11  
 csr (*ownca.ownca.HostCertificate* property), 15  
 csr (*ownca.ownca.OwncaCertData* property), 17  
 csr (*ownca.utils.CAStatus* attribute), 19  
 csr\_bytes (*ownca.ownca.CertificateAuthority* property), 11  
 csr\_bytes (*ownca.ownca.HostCertificate* property), 15  
 csr\_bytes (*ownca.ownca.OwncaCertData* property), 17

## F

file\_data\_status() (*in module ownca.utils*), 19  
 format\_oids() (*in module ownca.ownca*), 18

## G

generate() (*in module ownca.crypto.keys*), 21

## H

hash\_name (*ownca.ownca.CertificateAuthority* property), 11  
 HostCertificate (*class* in *ownca.ownca*), 14

## I

initialize() (*ownca.ownca.CertificateAuthority* method), 11  
 issue\_certificate() (*ownca.ownca.CertificateAuthority* method), 12

## K

key (*ownca.crypto.keys.OwncaKeyData* property), 20  
 key (*ownca.ownca.CertificateAuthority* property), 12  
 key (*ownca.ownca.HostCertificate* property), 15  
 key (*ownca.ownca.OwncaCertData* property), 17  
 key (*ownca.utils.CAStatus* attribute), 19  
 key\_bytes (*ownca.crypto.keys.OwncaKeyData* property), 21  
 key\_bytes (*ownca.ownca.CertificateAuthority* property), 12  
 key\_bytes (*ownca.ownca.HostCertificate* property), 15  
 key\_bytes (*ownca.ownca.OwncaCertData* property), 17

## L

load\_cert\_files() (*in module ownca.ownca*), 18  
 load\_certificate() (*ownca.ownca.CertificateAuthority* method), 13

## M

module  
   ownca, 20  
   ownca.crypto, 21  
   ownca.crypto.keys, 20  
   ownca.exceptions, 9  
   ownca.ownca, 10  
   ownca.utils, 19

## O

ownca

- module, 20
- ownca.crypto
  - module, 21
- ownca.crypto.keys
  - module, 20
- ownca.exceptions
  - module, 9
- ownca.ownca
  - module, 10
- ownca.utils
  - module, 19
- ownca\_directory() (in module ownca.utils), 19
- OwncaCertData (class in ownca.ownca), 16
- OwnCAFatalError, 9
- OwnCAInconsistentData, 9
- OwnCAIntermediate, 9
- OwnCAInvalidCertificate, 9
- OwnCAInvalidDataStructure, 9
- OwnCAInvalidFiles, 9
- OwnCAInvalidOID, 9
- OwncaKeyData (class in ownca.crypto.keys), 20

## P

- public\_key (ownca.crypto.keys.OwncaKeyData property), 21
- public\_key (ownca.ownca.CertificateAuthority property), 13
- public\_key (ownca.ownca.HostCertificate property), 15
- public\_key (ownca.ownca.OwncaCertData property), 18
- public\_key (ownca.utils.CAStatus attribute), 19
- public\_key\_bytes (ownca.crypto.keys.OwncaKeyData property), 21
- public\_key\_bytes (ownca.ownca.CertificateAuthority property), 13
- public\_key\_bytes (ownca.ownca.HostCertificate property), 16
- public\_key\_bytes (ownca.ownca.OwncaCertData property), 18

## R

- revoke\_certificate()
  - (ownca.ownca.CertificateAuthority method), 13
- revoked (ownca.ownca.HostCertificate property), 16

## S

- sign\_csr()
  - (ownca.ownca.CertificateAuthority method), 13
- status (ownca.ownca.CertificateAuthority property), 14
- store\_file() (in module ownca.utils), 19

## T

- type (ownca.ownca.CertificateAuthority property), 14

## V

- validate\_hostname() (in module ownca.utils), 19