OCSP APIs for node.js [![Build Status](https://secure.travis-ci.org/indutny/ocsp.svg)](http://travis-ci.org/indutny/ocsp)

Various [OCSP][0]-related APIs to be used with node.js/io.js.

Installing

```bash
$ npm install ocsp
```

Parts

1. Agent
2. Cache
3. Server
4. `check()`/`verify()`
5. `request.generate()`
6. `getOCSPURI()`

Agent

Usage:

```javascript
var agent = new ocsp.Agent();

https.request({
  method: ...,
  host: ...,
  port: ...,
  path: ...,
  // Other options

  agent: agent
}, function(res) {
  // ...
});
```

The following code snippet will perform request to the specified server, and
will verify that the certificate of the server using OCSP (either stapling or
response from the CA).

NOTE: You may pass `options` object to `new ocsp.Agent(options)`, it may have
following properties:

* `CACacheSize`: number of CA certificates to keep in the cache. (Default: 1024)

Cache

Usage:

```javascript
var cache = new ocsp.Cache();

var server = https.createServer({
  cert: cert,
  key: key
}, function(req, res) {
  res.end('hello world');
});

server.on('OCSPRequest', function(cert, issuer, cb) {
  ocsp.getOCSPURI(cert, function(err, uri) {
    if (err) return cb(err);
    if (uri === null) return cb();

    var req = ocsp.request.generate(cert, issuer);
```

```
    cache.probe(req.id, function(err, cached) {
      if (err) return cb(err);
      if (cached !== false) return cb(null, cached.response);

      var options = {
        url: uri,
        ocsp: req.data
      };

      cache.request(req.id, options, cb);
    });
  });
});
```

Cache should be used to provide [OCSP Stapling][1] responses to the client.

NOTE: Constructor accepts `options` object with following properties:

* `probe`: override `.probe()` method
* `store`: override `.store()` method
* `filter`: `filter(url, callback)` to white list CA urls to do requests

Server

Usage:

```javascript
var server = ocsp.Server.create({
  cert: cert,
  key: key
});

server.addCert(43, 'good');
server.addCert(44, 'revoked', {
  revocationTime: new Date(),
  revocationReason: 'CACompromise'
});

server.listen(8000);
```

OCSP Server, i.o.w. HTTP server providing OCSP responses for supplied OCSP
requests.

Has following methods:

* `.addCert(serialNumber, status, info)`, where:
  * `serialNumber` could be either plain number, or instance of `bn.js`
  * `status` is one of `good`, `revoked`
  * `info` should be empty for `good` and should contain object for `revoked`
    (see example above, `revocationReason` is one of: `unspecified`,
     `keyCompromise`, `CACompromise`, `affiliationChanged`, `superseded`,
     `cessationOfOperation`, `certificateHold`, `removeFromCRL`,
     `privelegeWithdrawn`, `AACompromise`)
* All of `http.Server` methods!

.check()

Usage:

```javascript
ocsp.check({
  cert: cert,
  issuer: issuerCert
}, function(err, res) {
  if (err)
    throw err;

  console.log(res);
```

});
```

Send an OCSP request to the CA and ask if the cert is still valid. `res`
constains the info.

## .verify()

Usage:

```javascript
ocsp.verify({
  request: request,
  // Optional, `issuer: issuerCert,`
  response: response
}, function(err, res) {
});
```

Verify that `response` matches the `request` and is signed by the CA.

request.generate()

Usage:

```javascript
var req = ocsp.request.generate(cert, issuerCert);
```

Generate OCSP request for `.verify()` or for sending it manually to OCSP server.

getOCSPURI()

Usage:

```javascript
ocsp.getOCSPURI(cert, function(err, uri) {
});
```

Get URI of OCSP server.

LICENSE

[0]: http://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol
[1]: http://en.wikipedia.org/wiki/OCSP_stapling
```