Client certificate

Client certificate authentication can be configured with the `Client`, the required options are passed along through the `connect` option.

The client certificates must be signed by a trusted CA. The Node.js default is to trust the well-known CAs curated by Mozilla.

Setting the server option `requestCert: true` tells the server to request the client certificate.

The server option `rejectUnauthorized: false` allows us to handle any invalid certificate errors in client code. The `authorized` property on the socket of the incoming request will show if the client certificate was valid. The `authorizationError` property will give the reason if the certificate was not valid.

Client Certificate Authentication

```js
const { readFileSync } = require('fs')
const { join } = require('path')
const { createServer } = require('https')
const { Client } = require('undici')

const serverOptions = {
  ca: [
    readFileSync(join(__dirname, 'client-ca-crt.pem'), 'utf8')
  ],
  key: readFileSync(join(__dirname, 'server-key.pem'), 'utf8'),
  cert: readFileSync(join(__dirname, 'server-crt.pem'), 'utf8'),
  requestCert: true,
  rejectUnauthorized: false
}

const server = createServer(serverOptions, (req, res) => {
  // true if client cert is valid
  if(req.client.authorized === true) {
    console.log('valid')
  } else {
    console.error(req.client.authorizationError)
  }
  res.end()
})

server.listen(0, function () {
  const tls = {
    ca: [
      readFileSync(join(__dirname, 'server-ca-crt.pem'), 'utf8')
    ],
    key: readFileSync(join(__dirname, 'client-key.pem'), 'utf8'),
    cert: readFileSync(join(__dirname, 'client-crt.pem'), 'utf8'),
    rejectUnauthorized: false,
    servername: 'agent1'
  }
  const client = new Client(`https://localhost:${server.address().port}`, {
    connect: tls
  })

  client.request({
    path: '/',
    method: 'GET'
  }, (err, { body }) => {
    body.on('data', (buf) => {})
    body.on('end', () => {
      client.close()
      server.close()
    })
  })
})
```