Encrypt and decrypt content with Nodejs

WRITTEN BY CHRISTOPH HARTMANN

Nodejs offers great support for cryptography. Under the hood it uses openssl and ships with a Javascript api. Unfortunately the api is not always as intuitive as it should be, especially when you have to deal with error codes. To make you life easier, I collected various approaches for encryption with AES 256.

Update: All examples are available on Github node-crypto-examples, too.

Encryption mode

The first decision is the AES encryption mode. Currently I recommend the CTR mode. You may want to read Evaluation of Some Blockcipher Modes of Operation or On the Security of CTR + CBC-MAC. The next nodejs version comes with support for GCM to do authenticated encryption. Until then you have to use approaches like Encrypt-then-MAC and combine the encryption with the generation of SHA hashs.

Examples

Encrypt and decrypt text

```
1  // Part of https://github.com/chris-rock/node-crypto-examples
2
3  // Nodejs encryption with CTR
4  var crypto = require('crypto'),
5      algorithm = 'aes-256-ctr',
6      password = 'd6F3Efeq';
7
8  function encrypt(text){
9    var cipher = crypto.createCipher(algorithm,password)
```

```
  var crypted = cipher.update(text,'utf8','hex')
  crypted += cipher.final('hex');
  return crypted;
}

function decrypt(text){
  var decipher = crypto.createDecipher(algorithm,password)
  var dec = decipher.update(text,'hex','utf8')
  dec += decipher.final('utf8');
  return dec;
}

var hw = encrypt("hello world")
// outputs hello world
console.log(decrypt(hw));
```

crypto-ctr.js hosted with ♡ by GitHub view raw

Encrypt and decrypt buffers

```
// Part of https://github.com/chris-rock/node-crypto-examples

var crypto = require('crypto'),
    algorithm = 'aes-256-ctr',
    password = 'd6F3Efeq';

function encrypt(buffer){
  var cipher = crypto.createCipher(algorithm,password)
  var crypted = Buffer.concat([cipher.update(buffer),cipher.final()]);
  return crypted;
}

function decrypt(buffer){
  var decipher = crypto.createDecipher(algorithm,password)
  var dec = Buffer.concat([decipher.update(buffer) , decipher.final()]);
  return dec;
}

var hw = encrypt(new Buffer("hello world", "utf8"))
// outputs hello world
console.log(decrypt(hw).toString('utf8'));
```

crypto-buffer.js hosted with ♡ by GitHub view raw

Encrypt and decrypt streams

```
// Part of https://github.com/chris-rock/node-crypto-examples

// Nodejs encryption of buffers
var crypto = require('crypto'),
    algorithm = 'aes-256-ctr',
    password = 'd6F3Efeq';

var fs = require('fs');
var zlib = require('zlib');

// input file
var r = fs.createReadStream('file.txt');
// zip content
var zip = zlib.createGzip();
// encrypt content
var encrypt = crypto.createCipher(algorithm, password);
// decrypt content
var decrypt = crypto.createDecipher(algorithm, password)
// unzip content
var unzip = zlib.createGunzip();
// write file
var w = fs.createWriteStream('file.out.txt');

// start pipe
r.pipe(zip).pipe(encrypt).pipe(decrypt).pipe(unzip).pipe(w);
```

crypto-stream.js hosted with ♡ by GitHub view raw

Use GCM for authenticated encryption

If you replace `aes-256-ctr` with `aes-256-gcm` you may think everything works as expected. Unfortunately this will result with a confusing error message: `TypeError: error:00000000:lib(0):func(0):reason(0)`.

Authenticated encryption includes a hash of the encrypted content and helps you to identify manipulated encrypted content.

You need to set the authentication tag via `decrypt.setAuthTag()`, which is currently only available if you use `crypto.createCipheriv(algorithm, key, iv)` with an initialization vector. GCM's security is dependent on choosing a unique initialization vector for each encryption.

```
// Nodejs encryption with GCM
// Does not work with nodejs v0.10.31
// Part of https://github.com/chris-rock/node-crypto-examples

var crypto = require('crypto'),
  algorithm = 'aes-256-gcm',
  password = '3zTvzr3p67VC61jmV54rIYu1545x4TlY',
  // do not use a global iv for production,
  // generate a new one for each encryption
  iv = '60iP0h6vJoEa'

function encrypt(text) {
  var cipher = crypto.createCipheriv(algorithm, password, iv)
  var encrypted = cipher.update(text, 'utf8', 'hex')
  encrypted += cipher.final('hex');
  var tag = cipher.getAuthTag();
  return {
    content: encrypted,
    tag: tag
  };
}

function decrypt(encrypted) {
  var decipher = crypto.createDecipheriv(algorithm, password, iv)
  decipher.setAuthTag(encrypted.tag);
  var dec = decipher.update(encrypted.content, 'hex', 'utf8')
  dec += decipher.final('utf8');
  return dec;
}

var hw = encrypt("hello world")
  // outputs hello world
console.log(decrypt(hw));
```

crypto-gcm.js hosted with ♡ by GitHub view raw

The new GCM mode is available in nodejs 0.11. Try it with n via

```
npm install -g n
sudo n 0.11.13
n use 0.11.13 crypto-gcm.js
```

Also take a look at the nodejs tests for more tests with different setups.

Conclusion

I hope the samples help you to get started with nodejs encryption.

If you have any questions contact me via Twitter @chri_hartmann or Github

See also:

- SHA 512 Hashs with nodejs
- Simple file uploads with Express 4
- Applied Content Security Policy for Nginx and Nodejs
- Ready for ES6?

Discuss on Hacker News Tweet

« Full blog