ENCRYPTIONKEY2 and Local Data Encrypting Method in LTIjs

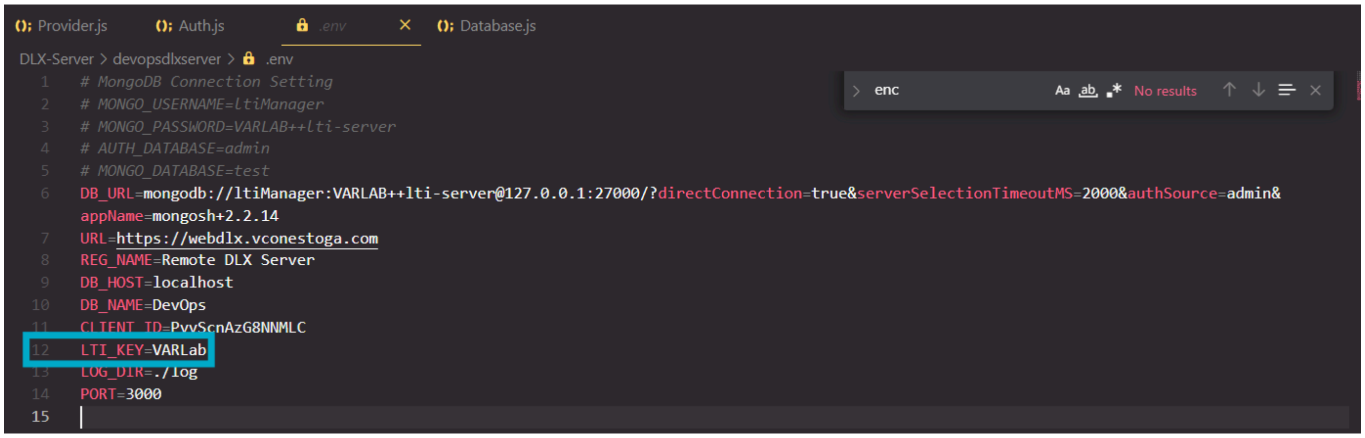
1. */_ENCRYPTIONKEY2*

: This encrypt key is used to encrypting a data when we insert or get data from the database. So, all the basic logic of encrypting data can be found in '**Database.js**' file

The "**ENCRYPTIONKEY2**" data will be set during the "**.setup()**" process.

For example,

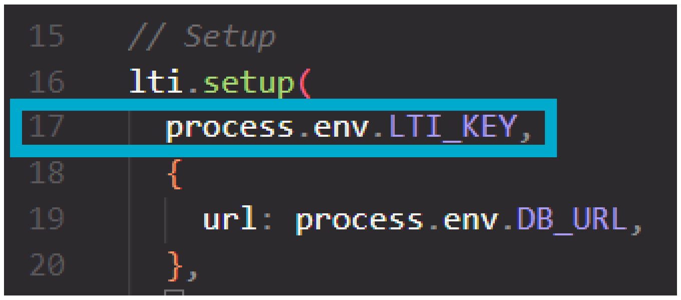
We set our ENCRYPTIONKEY2 as a "VARLab"



```
# MongoDB Connection Setting
# MONGO_USERNAME=ltiManager
# MONGO_PASSWORD=VARLAB++lti-server
# AUTH_DATABASE=admin
# MONGO_DATABASE=test
DB_URL=mongodb://ltiManager:VARLAB++lti-server@127.0.0.1:27000/?directConnection=true&serverSelectionTimeoutMS=2000&authSource=admin&
appName=mongosh+2.2.14
URL=https://webdlx.vconestoga.com
REG_NAME=Remote DLX Server
DB_HOST=localhost
DB_NAME=DevOps
CLIENT_ID=PvyScnAzG8NNMLC
LTI_KEY=VARLab
LOG_DIR=./log
PORT=3000
```

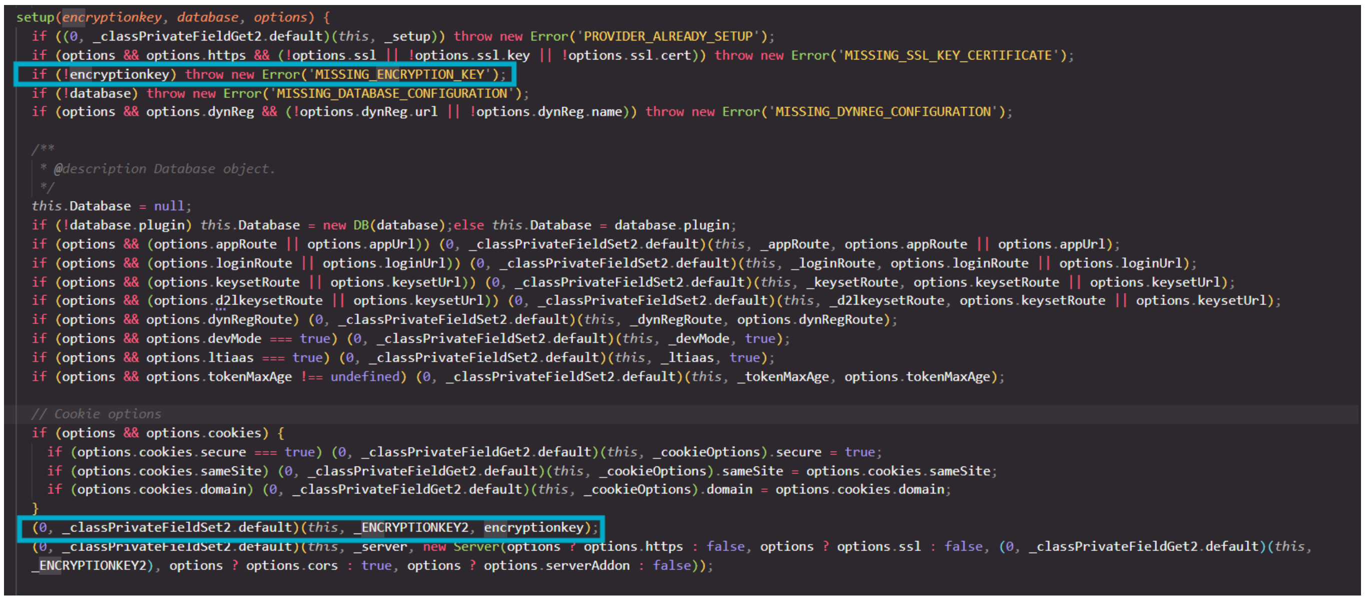
And pass set it when during the setup() process

[.setup() function call in the devpopsdlxserver/index.js]



```
// Setup
lti.setup(
  process.env.LTI_KEY,
  {
    url: process.env.DB_URL,
  },
```

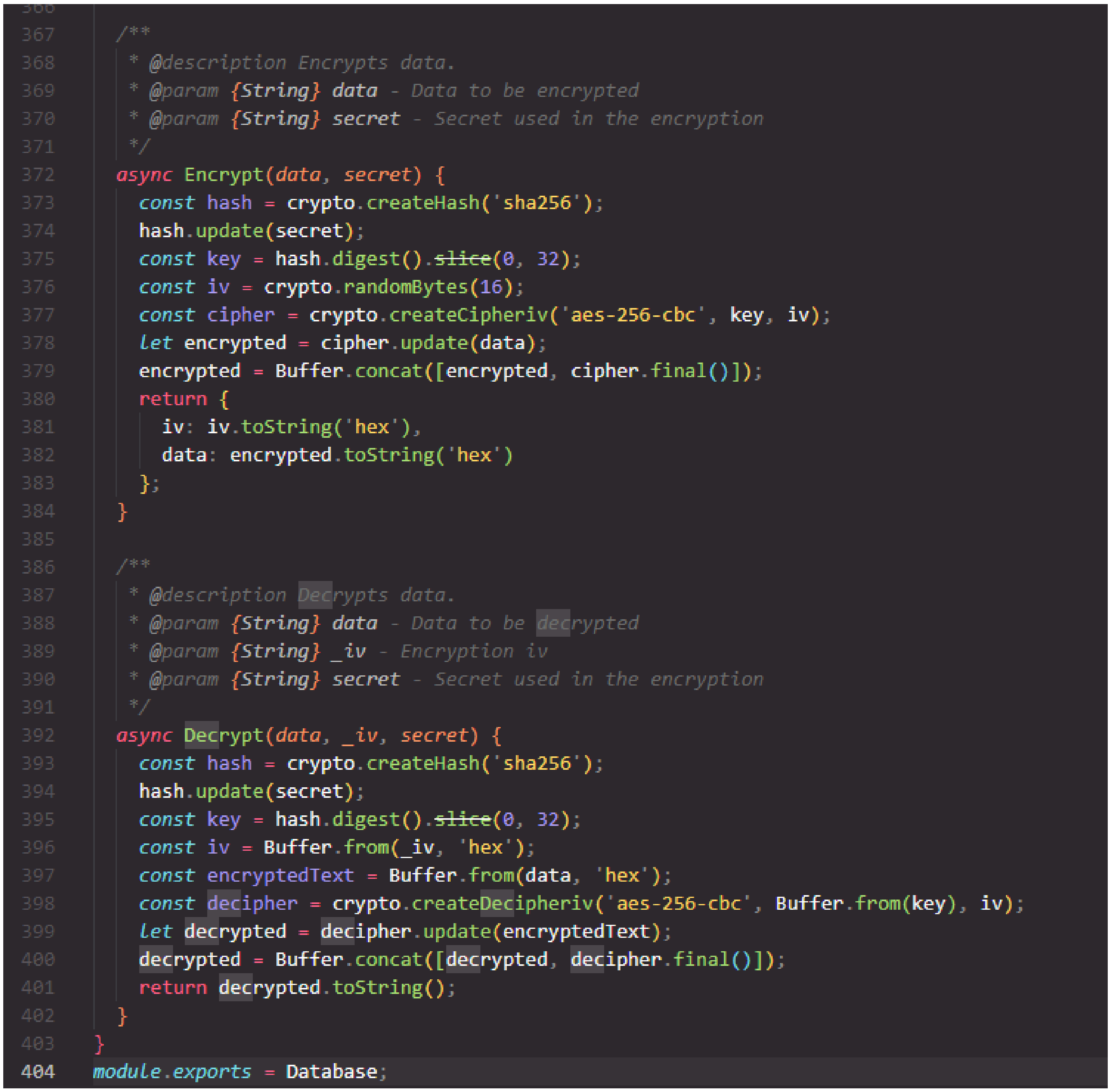
[.setup() function definition in the devpopsdlxserver/node_modules/ltijs/dis/Provider/Provider.js]



```
setup(encryptionkey, database, options) {
  if ((0, _classPrivateFieldGet2.default)(this, _setup)) throw new Error('PROVIDER_ALREADY_SETUP');
  if (options && options.https && (!options.ssl || !options.ssl.key || !options.ssl.cert)) throw new Error('MISSING_SSL_KEY_CERTIFICATE');
  if (!encryptionkey) throw new Error('MISSING_ENCRYPTION_KEY');
  if (!database) throw new Error('MISSING_DATABASE_CONFIGURATION');
  if (options && options.dynReg && (!options.dynReg.url || !options.dynReg.name)) throw new Error('MISSING_DYNREG_CONFIGURATION');

  /**
   * @description Database object.
   */
  this.Database = null;
  if (!database.plugin) this.Database = new DB(database);else this.Database = database.plugin;
  if (options && (options.appRoute || options.appUrl)) (0, _classPrivateFieldSet2.default)(this, _appRoute, options.appRoute || options.appUrl);
  if (options && (options.loginRoute || options.loginUrl)) (0, _classPrivateFieldSet2.default)(this, _loginRoute, options.loginRoute || options.loginUrl);
  if (options && (options.keysetRoute || options.keysetUrl)) (0, _classPrivateFieldSet2.default)(this, _keysetRoute, options.keysetRoute || options.keysetUrl);
  if (options && (options.d2lkeysetRoute || options.keysetUrl)) (0, _classPrivateFieldSet2.default)(this, _d2lkeysetRoute, options.keysetRoute || options.keysetUrl);
  if (options && options.dynRegRoute) (0, _classPrivateFieldSet2.default)(this, _dynRegRoute, options.dynRegRoute);
  if (options && options.devMode === true) (0, _classPrivateFieldSet2.default)(this, _devMode, true);
  if (options && options.ltiaas === true) (0, _classPrivateFieldSet2.default)(this, _ltiaas, true);
  if (options && options.tokenMaxAge !== undefined) (0, _classPrivateFieldSet2.default)(this, _tokenMaxAge, options.tokenMaxAge);

  // Cookie options
  if (options && options.cookies) {
    if (options.cookies.secure === true) (0, _classPrivateFieldGet2.default)(this, _cookieOptions).secure = true;
    if (options.cookies.sameSite) (0, _classPrivateFieldGet2.default)(this, _cookieOptions).sameSite = options.cookies.sameSite;
    if (options.cookies.domain) (0, _classPrivateFieldGet2.default)(this, _cookieOptions).domain = options.cookies.domain;
  }
  (0, _classPrivateFieldSet2.default)(this, _ENCRYPTIONKEY2, encryptionkey);
  (0, _classPrivateFieldSet2.default)(this, _server, new Server(options ? options.https : false, options ? options.ssl : false, (0, _classPrivateFieldGet2.default)(this,
  _ENCRYPTIONKEY2), options ? options.cors : true, options ? options.serverAddon : false));
```

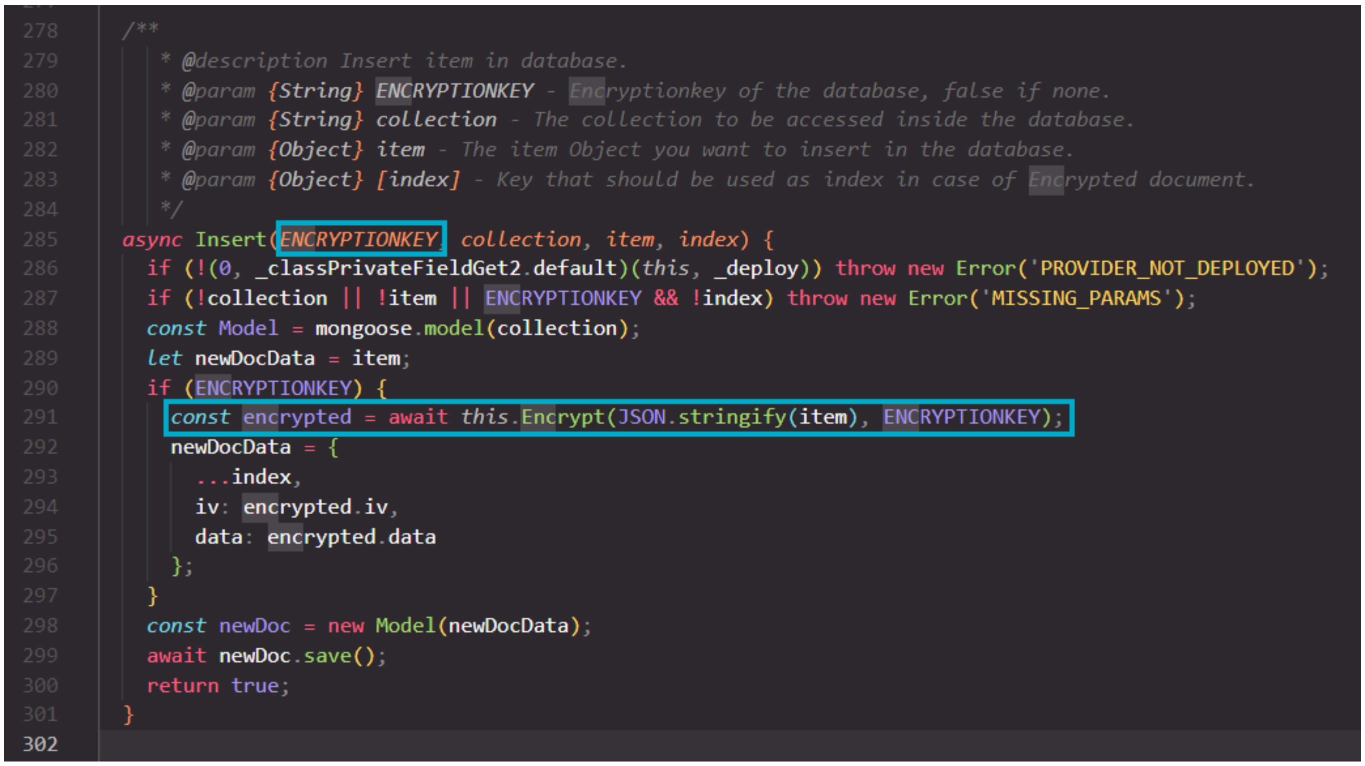
After ENCRYPTIONKEY2 has been setup, it wil be for encrypting or decrypting data from Local MongoDB

[Encrypt() and Decrpyt() Methods from the Database.js]



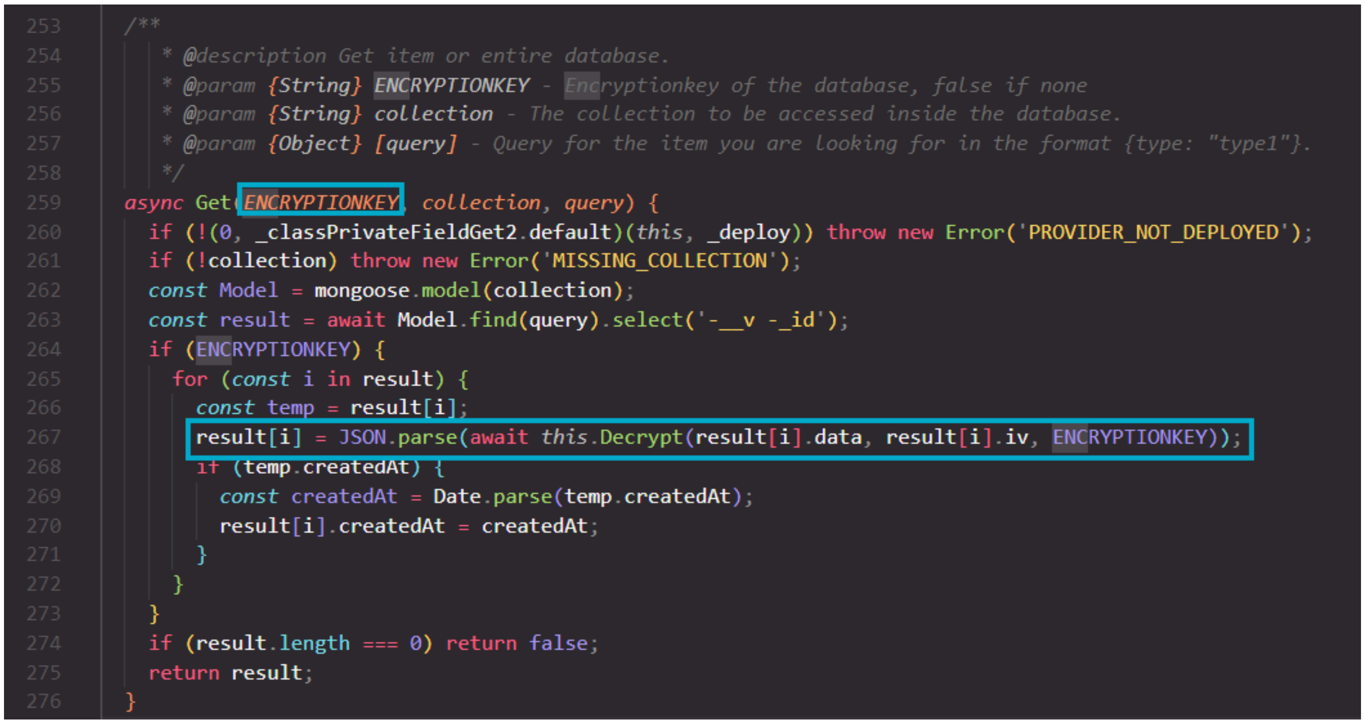
```
    /**
     * @description Encrypts data.
     * @param {String} data - Data to be encrypted
     * @param {String} secret - Secret used in the encryption
     */
    async Encrypt(data, secret) {
      const hash = crypto.createHash('sha256');
      hash.update(secret);
      const key = hash.digest().~~slice~~(0, 32);
      const iv = crypto.randomBytes(16);
      const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);
      let encrypted = cipher.update(data);
      encrypted = Buffer.concat([encrypted, cipher.final()]);
      return {
        iv: iv.toString('hex'),
        data: encrypted.toString('hex')
      };
    }

    /**
     * @description Decrypts data.
     * @param {String} data - Data to be decrypted
     * @param {String} _iv - Encryption iv
     * @param {String} secret - Secret used in the encryption
     */
    async Decrypt(data, _iv, secret) {
      const hash = crypto.createHash('sha256');
      hash.update(secret);
      const key = hash.digest().~~slice~~(0, 32);
      const iv = Buffer.from(_iv, 'hex');
      const encryptedText = Buffer.from(data, 'hex');
      const decipher = crypto.createDecipheriv('aes-256-cbc', Buffer.from(key), iv);
      let decrypted = decipher.update(encryptedText);
      decrypted = Buffer.concat([decrypted, decipher.final()]);
      return decrypted.toString();
    }
  }
  module.exports = Database;
```

[Insert() Method from Database.js]



```
/**
  * @description Insert item in database.
  * @param {String} ENCRYPTIONKEY - Encryptionkey of the database, false if none.
  * @param {String} collection - The collection to be accessed inside the database.
  * @param {Object} item - The item Object you want to insert in the database.
  * @param {Object} [index] - Key that should be used as index in case of Encrypted document.
  */
async Insert(ENCRYPTIONKEY, collection, item, index) {
  if (!(0, _classPrivateFieldGet2.default)(this, _deploy)) throw new Error('PROVIDER_NOT_DEPLOYED');
  if (!collection || !item || ENCRYPTIONKEY && !index) throw new Error('MISSING_PARAMS');
  const Model = mongoose.model(collection);
  let newDocData = item;
  if (ENCRYPTIONKEY) {
    const encrypted = await this.Encrypt(JSON.stringify(item), ENCRYPTIONKEY);
    newDocData = {
      ...index,
      iv: encrypted.iv,
      data: encrypted.data
    };
  }
  const newDoc = new Model(newDocData);
  await newDoc.save();
  return true;
}
```

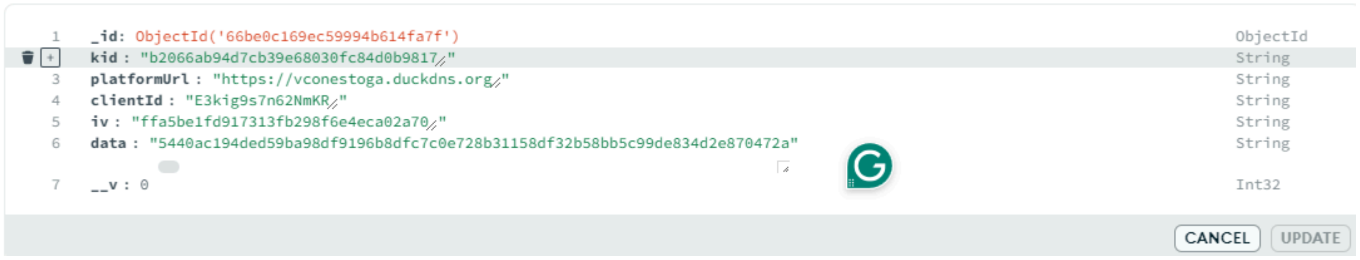
[Get() Method from Database.js]



```
/**
  * @description Get item or entire database.
  * @param {String} ENCRYPTIONKEY - Encryptionkey of the database, false if none
  * @param {String} collection - The collection to be accessed inside the database.
  * @param {Object} [query] - Query for the item you are looking for in the format {type: "type1"}.
  */
async Get(ENCRYPTIONKEY, collection, query) {
  if (!(0, _classPrivateFieldGet2.default)(this, _deploy)) throw new Error('PROVIDER_NOT_DEPLOYED');
  if (!collection) throw new Error('MISSING_COLLECTION');
  const Model = mongoose.model(collection);
  const result = await Model.find(query).select('-__v -_id');
  if (ENCRYPTIONKEY) {
    for (const i in result) {
      const temp = result[i];
      result[i] = JSON.parse(await this.Decrypt(result[i].data, result[i].iv, ENCRYPTIONKEY));
      if (temp.createdAt) {
        const createdAt = Date.parse(temp.createdAt);
        result[i].createdAt = createdAt;
      }
    }
  }
  if (result.length === 0) return false;
  return result;
}
```

This `Encrypt()` or `Decrpyt()` method is called every database interaction.

For example encrypted `publickey` data looks like this



```
_id: ObjectId('66be0c169ec59994b614fa7f')          ObjectId
kid : "b2066ab94d7cb39e68030fc84d0b9817"           String
platformUrl : "https://vconestoga.duckdns.org"     String
clientId : "E3kig9s7n62NmKR"                       String
iv : "ffa5be1fd917313fb298f6e4eca02a70"            String
data : "5440ac194ded59ba98df9196b8dfc7c0e728b31158df32b58bb5c99de834d2e870472a"   String
__v : 0                                            Int32
```

Here is the data after decoding

[Sample Encrypted Public Key Data]

```
5440ac194ded59ba98df9196b8dfc7c0e728b31158df32b58bb5c99de834d2e870472af926
8ac793f660f3688bf7483437a0cb665551c9bda07973bfd979d9d88f7877ec902369287f67
94e0b7983f4a5c1ffcaa4ccd755a6fa23f5231a45a07d1ab975fd970d4aa6e3a8a3acbe1e8
9ddedf20e7b0a7e3553506a7329b471054de6ac4759f81e79bd8459540a7e006b02a4b7f97
3f9269ee9b6e489b2fe4bdda973d49a714bc12249f5c04ddbf1507127a23bfb0dcdd04683c
45448259f8521b43806b93232e994112e431cf69fe012cda14f0b4238d31c50e8a403d3193
a852dc5de7735a616dbbe47dc2415876a1438292e6724880314b1cae8db546cd59d024feda
b0e20572044787edcbcce25943ed371d80b8d248bbf4e6715b944fda26198598db75f95f4d
6aac4af294db9e413e7e8084da6d97d1d3d802e474a8d05e2997e9b8748d4cb2b7b7d62b93
3544de33b35a54f7267ebad0ff81b1e28cd77990b4611a9080274260c6551f28ecbbc71782
fefe0832d5d6c617020a7366402355ff7562ceb213dcaebe77dc8633ab56f761a9d9fb1245
f3966c9804a3dcecf55f4a6b8fff663229654a3021e1dd6b4a2b07a7328761259b5bf1179a
dafc04a4e65aea17520e96290cc2ab17f7a890c8d925b909c18ff36e590a6c863eb425f033
77152a0efb77ef775c6dc2ab753204cc4eaa25826f84383f7718895d968e05f2916167645a
2b12e39389e053e8ae1d72965a980870068e40ac8927de678741d3385686ebda0b7ccc0784
e4be83e21af19d8b23e2b6c0d32e5efa5435fe1e8b7a3cc2714ac4cb1ac583de9e0b4a26c9
494e4bc77f90b736c1030bc39585c952ce17f72e329b3274811c244d068eff8af471e97a5b
ddab66b15626ffc0010b5f4578e4834119fe9dd4f4e36b672d2556659a09eec34752561d56
ba952a004567d137bc55bbe0f582aafbe0ac5276fb89acbb751a4d6bc8dfaefe96a3d710e0
0cedeeef088c07f539efcc420d38717d1398a6f5147b0ba0864ca618ce952ce94f1185853a
176200380552c39203da59d86915fe7c5fe679753c9e257f92e10e5753cc150e31bfb91daa
c55357d85d010cd7a93314024abcb8ca43f3eba479ec09dc90de8323950a618eed23332ecf
f87bb693eb27f709fb3f29443a30e103d0960ca80274e468bf3de1836a59160ec00edef5c3
1a5188f915429a1e79cfb8176bc729920d964a6097fc60ee116603a787c0fcf35c2782bdab
c75253dcf3a1bd5df96ee51400b54a2839277c9d8ecf9f8a32b21a35e949ca98adeb906c70
c4db35a4f911bae7c48c0695385681937dbc8838bc944d44eee45088755b9715a14c7b1646
58a88c9e2f48a8647f3191c215f7797f6268b53ab9ef7acc9dc845e569be7fb3c92b2c99f5
b20cffbf3e89c7cad831fed3cdbbca077746fa7cfbaa44d9359ad27a85008cc30bea72e6f4
```

```
6c9c75e1edd7fa390c3f65d99476f7dc07cef95442836518a97d5012dfae2f9327cefa9998
ea26ed23bf3b6bfdc66f3981a8c72175e71eeb3d56de82d877a8bce3ebe648a2dba5fe5ee0
cd2f3ed895d703eb0d1b0531db0e0e7a5b29eb12bccd81968a482400ba61da4de046ee9d96
3106efc118d898ec7eda587405e96361b7d18724d4c2558adb88f53d556f5c03376031ca06
616cc03c2e6c83f9c7c2653fb1853d9fff35e600517dd5e3e9baaa9488433a2f78ab74756d
a0979e91de720a466550a16685b5c0610909d7ec726d9f88c06cbef6667a81940468b432b0
2cfa9f9b2bc4c3ea54966cc182af2e8d2dc5abe64755b281cb5f84144aaa1c2916561ec52c
0f1d6ae23c1cf53301ae3c2aff40a49f4774c2bb7da013b2e493de5daa1dddd3f56745c6b1
27dba52a10ea2b7d27908a8292b8815e9201467cc2a1a659c8acd8097a5b296a2853328612
8406b6319ef9dcddb0ba7412f1ca4fbd9a557ac177cbd3c64d1e5274c4b45a292bfd668044
95bbe7c0031473c87f6a0a9515a67c9d0f8c411d4be8bc87fcbfe82f132ee89875120f948a
d08b3fe8f2f86117367d219cabd7b8989a969b06c4a825c58251259f1290fa04e2a33583e9
4481acbe8cc5a1c6c2d652d3828ca730983255dc2389eed4c78aba34b27d8aa5db24662623
b8438c69dc4b3ab78377a03b7537108d220706806bbdc3e3a97e641171e9b6d691717b4992
395bbf4aa9be672a08bef1570161891c7b57a61e79e73294f1b117e26533e630f6cbabdc2e
6e2bf41f330889f564518cd7885e0af64919d2d8671c6969c2bdf1a751c64a37f5e1fe59c7
767ec4588af64d42cc76a7992d41c06aa58611cc20947e8322fe71c09299eb465d1d4de5e0
a61f789379a785cde928c10eeefbeff1a47c17552d2ff146ec5449224e4ed153f7e793d367
cb5e5779a691aed3542bb8145d10bf5b5a8e61edac52a374ef8f15def1f6187bd55ad48bc0
c104ec936a1e417f1f63d8b5742e0594f64042841fe8f281f06514a52270237b661b0009f3
6ede4878283021d15480c21b7b406dd38eba171ff1354238181cb0314ee6c0cba49fcacc1e
784e320e9e2a1ca116976b72dce46cddaaf913718e7e4230e8223ae2b2df7b82df46c524c6
a29f43c4061f2758d0fffea7eb57a61dccc7c690d6e026cdf1757427c1cf854e1876eecc55
ab29cf7d35cbbf63114029b28a1aa7c4255dea7d63f747f067eff310d42a272ee79615afa7
6bc57b809fe87e682bd9933df9de17c27d8613f2afb370dc80c0b1d91248908f0fc14f666c
7dec2798b730a4882f8f96b10547cf69f0797bc26c1c9b83355d94ce4d954dfe292a9d2fb7
16e8bca9ef1eab5975fc73bf040ba734ca523aae0ec853ec4e6d33d7e3f015dab9128deb4c
1a7bbd72d18912f96e1dcdaca9cd260f8e66963114e86fa758b3b23ef064b7bd1cdb7e6b98
3d3c871f36b16d41495a8439c884a1840197a6afbcbd14b5ce0cc019a896f7d2fe3e3f7d8c
22ff719b7e5b2d3da46769c8e8ceca1ae3ed456f41d9efdb8d6276a688aafda803a3679b3c
2b89b0ac9ae25197b7b8395e1a1a59f7c3ced4c943e34bd2096a31152bdccb395f1d759090
0811aa05613e951fcd27d3ee1cb085bb0b79473e93b46932685fad4022835bb5ec7103f5d8
6de7c8fb8544bf24592c90d75e827c669dc17a3ce4cd94a3b117bd0826da83f3ddec7ea9b2
1286be9b65dcb36fb4cfc40dedb4b0f41a30344f3700484fafaba46899a90ba6cc1c363e4f
ee3a4af41a510062349184dfb54c092f5b32f61044e1d8ca6b77fd66681271a0aea24d5c44
980aa3d8b5fed30ffea9e49a8a7515ea0d9a57cb222cf72ac6ea2d6d4806f4df2bb1f4d499
be072cbc6a42154d985f646e0e77c5d2534de6b15b78ee72b5790e3dcabea646ac27dc7074
dfccd659978fe0710da12021ea295cbace02d36ca60b8b23dccd49f8ddf93e7f352fb666d7
```

```
ddc0d7b0dac66ee12591e8182ce7f5186e2737950d6a5dd34e5db3e76845887d18cc0d28cb
000e75e1937977215178d079d2cced0e26b7a86b066f71dd6117687adb4e161c610978424f
cea9bcf9ccafb6a2a7777b5a5673468b54cfcdc8130e4311809ed369e041bd736fcce8cb01
acc739b72d024b6a2a5ab4a0eded18542c8c9d55e1759dbdd252f6a86810d8e64abbea3d9f
e8d4b4b8132515dae277084124ba902d052d15f1922cfb94fde6b7037eaf2f759117d0c2a7
2bb5f690d5596cde76913baebc8b55ebbbd1aed0f89c0d4eec6ebc9554866811755f379db7
1ced2e69bd5a0db4828e90f434ab89ed9971df57efa33fa2b45c607bef0db0a6dd3f4873bc
cd728862deb997ebd86a4535b962c780062d4e2835c1d488f6bb36208d0b30a280372043fb
c8f9418c38504d561bf7126bc5509d2af13b4dc1110d65463eee0a5e7da1772979e8359561
84b0fde29ce9fa4b58ade0bb432b42b8bb41ca50597f9cde4bec308617a333bf2bc9172dcf
f837202b8ed8ef311a02a5369a733d32efd4be6fb0ec8130601ce726e7477e4c6bf7df97e1
42a3d4e70ee101500179c2431b24c5585cb5c8b2fef0630ddf62f5688a62cbf83725449978
60ea0e7fcd270e2e9de045e6698fea3d3bf7b013d5853b174cb705002f3406ac0741221e85
3562b69fde5328acdae9faea12062541c1c28cddde1dddde2258caac47a74f8a1cfe59c60e
79a7f09818e0bfdce6e3e5a2f77b091eecbca50693de4c2aec6a3f57b1341f694f2fb9f584
3ca25734cb58f16d965d30d334c83e72312292e2dcf98be2dee3634b5fb4de837521701b2b
601c48f244e5c82e44533de1acd75272833e5fdcbf0c57198f974b185d9cdbfe95c39b1cec
9e46744c927f1fe4822471dece1520c0bc08641039375d7c854906a86cbfbe17d1060d2339
e317b8d8ddb3d50ed5de12459e8c78006e9ddbc363e3953a32097e29e9708083413563377c
04617469c337723394350126ab5339eab8c6d45e156c24ebd1c56a51691c9e303bd27a9600
91359b8cb7a2a8c3c490ec2b69e371c0c135a3e721d2d1e3f37c9742c5390e0fed11c17929
7ff07afe25b0f6c157c22fd1ebef531e83ae4852cce5aa93f572ffd623191b5eb820ae96e3
71fc7647293470d786b14b58b3cd51cd6ba9fca2c1e175dee39609bc7f07f390e564b526a0
dff528660a9335fcc643c0ad91d29d834e737f9cb30df62267d19031ee29d71d644067a52f
e9ae429a2b398d24d552882a332f2b2287f3f1e85032c32c5c49d775e4e5
```

[Sample Decrypted Public Key]

```
{ "key": "-----BEGIN PUBLIC KEY-----
\nMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAswhBPDpfMSYpbGjBeEG7\nolW3cY
Ir1mtJzxWqTkJmZfECcJsXDt9OUvn3H8ZyeKybxJorikG9tiDiOAfJaS/6\nRjBEFfJScJE7OV
KhNqAttdIqFvFsiWKcUb4iHVnw9BM1hsLi325t2Rmf36U+xSZe\neD3NbDrH3Sbebfu98ZUrA1
FxkF5mENjFbwcR6bQvB1lFPaXcA0O9hoZmcq+4a4S1\nHNvWEngKfuA8ka9iAzm7+aMjJKg6Sz
XdPdCSo8Gt2aKT8HbObN1KwE+H9H5o812P\nHSLm86vekHTBgLMAKYVDgxT7WNumfWcSokpoeH
F5uDYZx9kvoEy+CXdgTjJ253bX\nBl4+kKiCIEC2lO0d5UsZSss/hlvJdkx9gjusvkG42FdrY1
H4Jl/dKE0RpGy4RCgM\nYxuJUCVxe9kKBT1QfXaTGoxGWv1wl/rl847hUgSP4v5NjnZEDho2AO
7YsHlbTuBh\nPZYR2iMUm46HTIQqIbA8C2khVFBFfRs4iuUeM+K2wc6JEH640MjLAkSerNMFSF
z0\nBMkeAqeIrr7H7FKl+rpmXfLFtdmreXOVLvT2DabRQmivcdSFEnTNxTyvNaf+qlq/\nFxVQ
rFc3ietTnYgtozMJSWnyvs0zXq8hh4Ax/pECDXsLsXMcTgM7d0vtU90YX0Zj\nU/r+IwB2PKU2
RGrns/5rCH8CAwEAAQ==\n-----END PUBLIC KEY-----\n", "kid":
"50c15f8802751730660b807232ee0e4d" }
```

We need to know 'iv', 'encrypt algorithm', and 'ENCRYPTIONKEY2' to decode the data from the database

- **iv**: can be found in the database it self
- **encrypt algorithm**: currently we are using 'sha256'
- **ENCRYPTIONKEY2**: we set 'VARLab' as an encryption key