SecretEngine

Painless file encryption library for .NET

About SecretEngine

SecretEngine is reliable, yet easy to use, file encryption library for .NET. It takes care of the nasty details so you can provide secure solutions for your customers with ease.

SecretEngine is cross-platform and supports Windows, MacOS as well as Linux. SecretEngine supports both 32 bit and 64 bit. It's compiled for .NET Standard 2 which means is compatible for .NET Framework 4.6.2 (at least 4.7.2 is recommended) or later as well as for .NET Core 2.0 or later.

Technical details

Under the hood SecretEngine uses Advanced Encryption Standard (AES) in CBC mode with 256bit keys. 128 bit cryptographically random initialization vector (IV) is used.

For key derivation PBKDF2 function is used. Key derivation is configured to use 200000 iterations. Also, 256 bit cryptographically random salt is applied.

For authentication HMAC with SHA256 is used. SecretEngine uses encrypt-then-HMAC which is the preferred way for data authentication. This means that data is first encrypted and then HMACSHA256 is computed for the encrypted data. Verification of the HMAC is done in timing safe manner before attempting to decrypt the data.

Code examples

SecretEngine is very easy to use and is designed to prevent misuse. Just with couple of lines of code you can successfully encrypt a file or even files in any directory (given that you have sufficient permissions).

This code example shows how to encrypt individual files as well as directories.

```
//30-day trial license.
//For purchased license use: new License(License.LicenseType.FULL, "licensecode");
License license = new License(License.LicenseType.TRIAL, null);
Crypto cryptoEngine = new Crypto(license);
CryptoKey key = new CryptoKey("your-secure-password");
bool deleteOriginal = true;
string error;

//Encrypting a single file
if(!cryptoEngine.EncryptFile("/path/to/some/file", key, deleteOriginal))
        error = cryptoEngine.LastErrorMessage;
//Decrypting a single file
if(!cryptoEngine.DecryptFile("/path/to/some/file.secret", key, deleteOriginal))
        error = cryptoEngine.LastErrorMessage;

//Encrypting directory recursively
bool recursive = true;
//In this tuple item1 (bool) indicates if the file encryption was OK.
//Second item (item2) is the filename(full path) to the processed file
List<Tuple<bool, string>> files;
files = cryptoEngine.EncryptDirectory("/path/to/some/dir", key, recursive,
deleteOriginal);

foreach (var t in files)
{
        if (t.Item1 == false)
        {
            Console.WriteLine("Failed to encrypt file " + t.Item2 + ".");
        }
}
//DecryptDirectory() works exactly the same way.
```

Support

Above example shows how simple it is to use SecretEngine. Create a license, create the engine itself and create a CryptoKey from your password.

If you need support or even custom functionality for SecretEngine, do not hesitate to contact us. Fastest way to reach us is to send an email to our lead developer niko@byteptr.com