SECRET/SecretEngine

Command line file encryption tool and development library

About SECRET

SECRET is a command line encryption tool for Windows. It comes as a standalone binary file which does not need installation. Copy the file to some path and run it from there. Note that you will need .NET 5 installed for SECRET to work.

SECRET is very easy to use and can handle encryption of very large files. It supports all file types. SECRET also allows you to encrypt directories with ease, even recursively.

Examples:

Note that SECRET has built-in help function which you can call by using SECRET.exe –help.

File encryption:

```
SECRET.exe encrypt -f  ”/path/to/some/file.txt” –delete
```

This example would encrypt file.txt, rename it to file.txt.secret and after the encryption delete the original file (file.txt)

File decryption:

```
SECRET.exe decrypt -f  ”/path/to/some/file.txt.secret” –delete
```

This example would decrypt file.txt.secret, rename it to file.txt and after the encryption delete the original encrypted file (file.txt.secret)

Directory encryption:

```
SECRET.exe encrypt -d  ”/path/to/some/test-dir” --recursive
```

Here we’re encrypting directory test-dir with –recursive flag. So all directories under test-dir will be encrypted too. How SECRET works is that it goes through all the files in the directory and encrypting them one by one. Without –delete flag SECRET will leave original files intact, useful for testing and for cases where you do want to keep the original files too.

Decrypting directory would work the same way but calling decrypt instead of encrypt subcommand.

Getting the password:

By default, SECRET will ask for the password via a password prompt in the command line. However, this is not very useful if you want to run SECRET as a part of some script. For example, a backup script where SECRET would encrypt your backups.

By setting an environment variable called SECRET_MASTER_KEY, if found, SECRET will use value that variable instead of prompting for it.

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 supports both 32 bit and 64 bit. It's compiled for .NET Standard 2.1 which means is compatible for .NET 5 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.

```
Crypto cryptoEngine = new Crypto();
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.
```

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