MBSecurity 3.0 Documentation

MBSecurity is a provided C#.Net library to helping you Encrypt and Decrypt data.

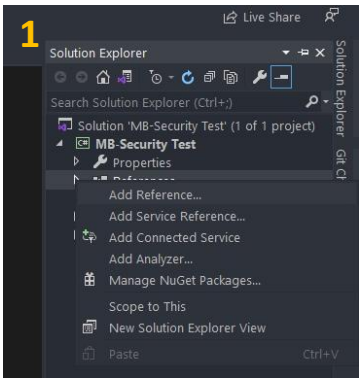
integrate in your project

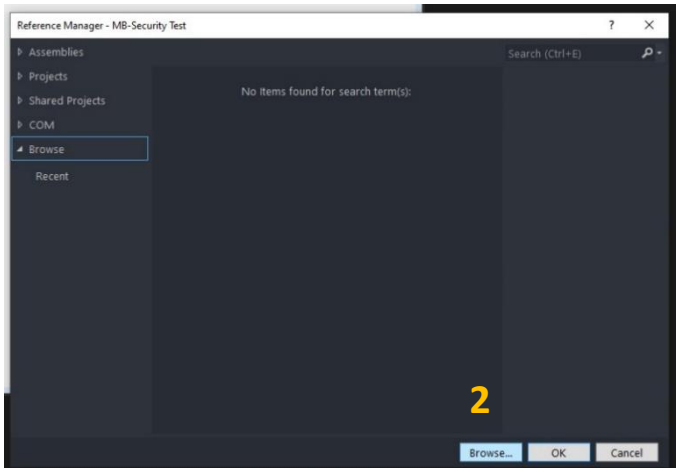
From DLLs

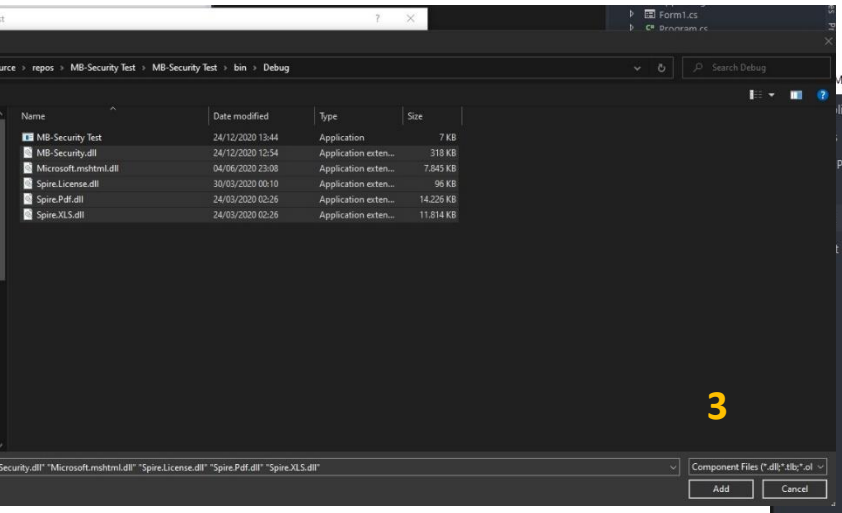
1. Download the DLLs from the GitHub repository Link
2. Click on MB-SECURITY 3.x.zip
3. Decompress the **.zip** and copy all files into your C# project bin/Debug folder
4. From **Solution explorer** in **visual studio,** right click on **References**
5. chose **Add reference**
6. Click on **Browse**
7. Navigate to your bin/Debug folder and select all DLLs and click on **Add**
8. Click on **Ok**

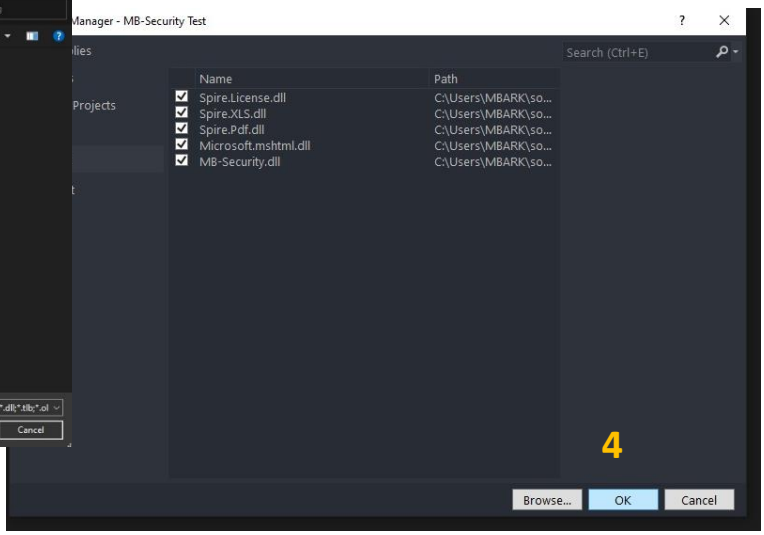
Steps in pics

1

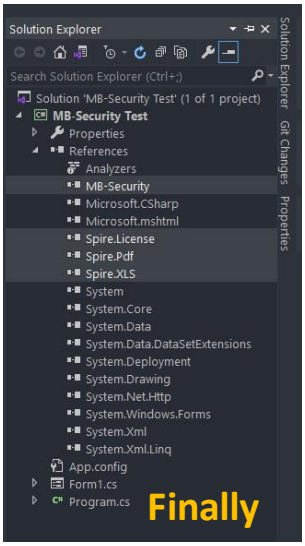








9. Finally, everything is ready



From Nuget Package

1. From visual studio go Tools
2. Nuget package manager
3. Manage NuGet packages for solution...
4. Click on Browse tab
5. In the search box type MBSecurity
6. Click on MBSecurity.By.MBARK.T3STO
7. In the right panel, under Version check project
8. Click on Install
9. Click on ok

Using

Now to use the library call it first in using section

```
Using MB_Security;
```

Possible Issue if you installed MBSecurity using Nuget package manager:

After typing `Using MB_Security;` and visual studio showed an error :

```
Error   CS0246  The type or namespace name 'MB_Security' could not be found (are you missing a using directive or an assembly reference?)
```

To fix this issue

1. Go and right click onto **references**
2. Chose **Add Reference…**
3. Click on **Browse…**
4. Navigate to your **project** folder
5. Go to **Packages** folder
6. Go to **MBSecurity.By.MBARK.T3STO.3.x.x** folder
7. Select all files and click on **Add**
8. Click on **Ok**.

Till now everything is ready to go

Encrypt Strings

- To **encrypt** a `string` value, use the `MBSecurity.MBString` static class.
- Two methods to encrypt a string :
    1. `Encrypt(Plain_Data,Key,IV)`
    2. `Encrypt(Plain_Data,Key)`
    3. `Encrypt(Plain_Data)`
    4. `EncryptWithoutKey(Plain_Data)`

Encrypt 3 overloads

Encrypt a string using a custom Key and/or IV

Parameter	Case	Type	Rules
Plain_Data	Required	String	-
Key	Optional	String	Should be 8 characters
IV	Optional	Byte[]	-

EncryptWithoutKey

Encrypt a string without Key and IV

Parameter	Case	Type	Rules
Plain_Data	Required	String	-

Example 1 (Encrypt a string value)

```
string S = "Welcome to MB Security";
//Encrypt string
var EncryptedData = new MBSecurity.StrEncryptionModel();
EncryptedData = MBSecurity.MBString.Encrypt( S , "ImTheKey" );

//Get the encrypted string and key
string Encrypted_String = EncryptedData.EncryptedText;
string Encrypted_Key    = EncryptedData.Key;

Console.WriteLine( $"Encrypted text : {Encrypted_String}" );
Console.WriteLine( $"Encrypted key : { Encrypted_Key}" );
```

Result

```
Encrypted text : yRLd6pLwO0XfEf5/+OrDdicXXTVDB9DR
Encrypted key : TdP2EenEgodLgNCowYOq7w==
```

Example 2 (Encrypt a string value with an auto generated Key)

```
string S = "Welcome to MB Security";
//Encrypt string
var EncryptedData = new MBSecurity.StrEncryptionModel();
EncryptedData = MBSecurity.MBString.Encrypt(S);

//Get the encrypted string and key
string Encrypted_String               = EncryptedData.EncryptedText;
string Encrypted_Auto_Generated__Key = EncryptedData.Key;

Console.WriteLine( $"Encrypted text : {Encrypted_String}" );
Console.WriteLine( $"Encrypted key : {Encrypted_Auto_Generated__Key}" );
```

Result

Example 3 (Encrypt without Key)

```
Encrypted text : elNvaJh6uXEzCx3VTDKA7EewXCuiXi5b
Encrypted Auto Generated key : 2b/GNnsTiiQGB1+1Fwh1dg==
```

To encrypt a **string** value without using **Key** use the **MBSecurity.MBString.EncryptWithoutKey** static method.

This method doesn't need a key to encrypt data.

```
string S = "Welcome to MB Security";

//Encrypt string
var EncryptedData = MBSecurity.MBString.EncryptWithoutKey(S);

//Get the encrypted string
string Encrypted_String = EncryptedData.EncryptedText;

Console.WriteLine( $"Encrypted text : {Encrypted_String}" );
```

Important Note

If you encrypt data using **EncryptWithoutKey** method, you should use **DecryptWithoutKey** method for decryption.

Decrypt Strings

- To **decrypt** an encrypted **string** value, use the MBSecurity.**MBString** static class.
- Two methods to decrypt an encrypted string :
    1. **Decrypt**(EncryptedData,Key,IV)
    2. **Decrypt**(EncryptedData,Key)
    3. **DecryptWithoutKey**(EncryptedData)

Decrypt 2overloads

Decrypt an encrypted string using a custom Key and/or IV used in encryption operation

Parameter	Case	Type	Rules
EncryptedData	Required	String	A string already encrypted using **MBSecurity**
Key	Required	String	The encrypted Key used in encryption
IV	Optional	Byte[]	The same IV used in encryption

DecryptWithoutKey

Decrypt an encrypted string using **EncryptWithoutKey** method

Parameter	Case	Type	Rules
EncryptedData	Required	String	A string already encrypted using **MBSecurity**

Example 1 (Decrypt using the Key and IV)

```
byte[] MyIV = new Byte[] {12, 6, 3, 2, 89, 3, 1, 23, 43};

string S = "Welcome to MB Security";

//Encrypt string
var EncryptedData = MBSecurity.MBString.Encrypt(S, "MyKey123", MyIV);

//Decrypt the encrypted string
var DecryptedData = MBSecurity.MBString.Decrypt(EncryptedData.EncryptedText,
EncryptedData.Key, MyIV);


Console.WriteLine( $"Encrypted text : {EncryptedData.EncryptedText}" );
Console.WriteLine( $"Decrypted the encrypted text : {DecryptedData.DecryptedText}" );
```

Result

```
Encrypted text : WUp7oRb82rTKdOT03JTKEtwwsVuho/Jq
Decrypted the encrypted text : Welcome to MB Security
```

Example 2 (Decrypt using the Key)

```
string S = "Welcome to MB Security";

//Encrypt string
var EncryptedData = MBSecurity.MBString.Encrypt(S, "MyKey123");

//Decrypt the encrypted string
var DecryptedData = MBSecurity.MBString.Decrypt(EncryptedData.EncryptedText,
EncryptedData.Key);


Console.WriteLine( $"Encrypted text : {EncryptedData.EncryptedText}" );
Console.WriteLine( $"Decrypted the encrypted text : {DecryptedData.DecryptedText}" );
```

Result

```
Encrypted text : Q5qfjmnOHpooZTTwhk/qEp+Nt2rQ3Gt0
Decrypted the encrypted text : Welcome to MB Security
```

Example 3 (Decrypt without Key)

To decrypt an encrypted **string** value without using **Key** use the **MBSecurity.MBString.DecryptWithoutKey** static method.

This method doesn't need a key to decrypt data.

Important Note

If you decrypt data using **DecryptWithoutKey** method, you should use **EncryptWithoutKey** method for encryption.

```
string S = "Welcome to MB Security";

//Encrypt string
var EncryptedData = MBSecurity.MBString.EncryptWithoutKey(S);

//Decrypt the encrypted string
var DecryptedData =
MBSecurity.MBString.DecryptWithoutKey(EncryptedData.EncryptedText);


Console.WriteLine( $"Encrypted text : {EncryptedData.EncryptedText}" );
Console.WriteLine( $"Decrypted the encrypted text : {DecryptedData.DecryptedText}" );
```

Result

```
Encrypted text : Wqk5lqa/2voYpjGZNJVHL31NVY9XKIRP
Decrypted the encrypted text : Welcome to MB Security
```

Encrypt & Decrypt Files

- To **encrypt** a **file**, use the MBSecurity.MBFile static class.
- Two methods to encrypt files :

1. Encrypt(FileBytes,Key,IV)
2. Encrypt(FileBytes,Key)
3. Encrypt(FileBytes)

4. Encrypt(FilePath,Key,IV)
5. Encrypt(FilePath,Key)
6. Encrypt(FilePath)

7. EncryptWithoutKey(FileBytes)
8. EncryptWithoutKey(FilePath)

Encrypt 3 overloads

Encrypt a file using a custom Key and/or IV

Parameter	Case	Type	Rules
FileBytes	Required	Byte[]	-
Key	Optional	String	Should be 8 characters
IV	Optional	Byte[]	-

Encrypt 3 overloads

Encrypt a file using a custom Key and/or IV

Parameter	Case	Type	Rules
FilePath	Required	String	-
Key	Optional	String	Should be 8 characters
IV	Optional	Byte[]	-

EncryptWithoutKey

Encrypt a file without Key and IV

Parameter	Case	Type	Rules
FileBytes	Required	Byte[]	-

EncryptWithoutKey

Encrypt a file without Key and IV

Parameter	Case	Type	Rules
FilePath	Required	String	-

- To **decrypt** a **file**, use the MBSecurity.**MBFile** static class.
- Two methods to decrypt files :

1. **Decrypt**(EncryptedFile,Key,IV)
2. **Decrypt**(EncryptedFile,Key)
3. **DecryptWithoutKey**(EncryptedFile)

Decrypt 3overloads **Decrypt** an **encrypted** file using a custom Key and/or IV			
Parameter	**Case**	**Type**	**Rules**
EncryptedFile	Required	Byte[]	A file **bytes** already encrypted using **MBSecurity**
Key	Optional	Byte[]	The encrypted Key used in encryption
IV	Optional	Byte[]	The same IV used in encryption

DecryptWithoutKey Decrypt an **encrypted file** without Key and IV			
Parameter	**Case**	**Type**	**Rules**
EncryptedFile	Required	Byte[]	A file **bytes** already encrypted using **MBSecurity**

Example 1

```
string MyFilePath = "C:\\Users\\MBARK\\Desktop\\MyWorkBook.xlsx";

byte[] fileBytes = File.ReadAllBytes(MyFilePath);

//Encrypt file
var EncryptedData = MBSecurity.MBFile.Encrypt(fileBytes);

//Decrypt the encrypted file
var DecryptedData = MBSecurity.MBFile.Decrypt(EncryptedData.EncryptedBytes,
EncryptedData.Key.ToBytes());
```

Example 2 (encrypt your file without key)

To encrypt a file without using **Key,** use the MBSecurity.MBFile.EncryptWithoutKey static method.

This method doesn’t need a key to encrypt data.

Important Note

If you encrypt file using **EncryptWithoutKey** method, you should use **DecryptWithoutKey** method for decryption.

```
string MyFilePath = "C:\\Users\\MBARK\\Desktop\\MyWorkBook.xlsx";

byte[] fileBytes = File.ReadAllBytes(MyFilePath);

//Encrypt file
var EncryptedData = MBSecurity.MBFile.EncryptWithoutKey(fileBytes);

//Decrypt the encrypted file
var DecryptedData = MBSecurity.MBFile.DecryptWithoutKey(EncryptedData.EncryptedBytes);
```

Example 3 (you can encrypt your file by sending the path)

Whether **Encrypt** or **EncryptWithoutKey** method, instead of sending the file as bytes you can send just the file path.

First example

```
string MyFilePath = "C:\\Users\\MBARK\\Desktop\\MyWorkBook.xlsx";

byte[] fileBytes = File.ReadAllBytes(MyFilePath);

//Encrypt file
var EncryptedData = MBSecurity.MBFile.EncryptWithoutKey(MyFilePath);

//Decrypt the encrypted file
var DecryptedData = MBSecurity.MBFile.DecryptWithoutKey(EncryptedData.EncryptedBytes);
```

Second example

```
string MyFilePath = "C:\\Users\\MBARK\\Desktop\\MyWorkBook.xlsx";

byte[] fileBytes = File.ReadAllBytes(MyFilePath);

//Encrypt file
var EncryptedData = MBSecurity.MBFile.Encrypt(MyFilePath);

//Decrypt the encrypted file
var DecryptedData = MBSecurity.MBFile.Decrypt(EncryptedData.EncryptedBytes,
EncryptedData.Key.ToBytes());
```

Important Notes

If you **encrypt data** using **EncryptWithoutKey** method, you should use **DecryptWithoutKey** method **for decryption.**

You can use **MBSecurity.MBXLSX** static class for a special encryption for **Excel files.**

You can use **MBSecurity.MBPDF** static class for a special encryption for **PDF files.**

You can use many extension methods provided by **MBSecurity**

Some Extension methods provided by MBSecurity

Helping you to do some routine operations (related with Encoding and Decoding) in short time

Ext Method	Description	Return data type
.ToByte	Encode a **String** or **Int** value to Bytes	Byte[]
.FromBase64ToBytes	Decode a String value (From Base64) to Bytes array Equivalent	Byte[]
.FromBytesToString	Decode an array of Bytes to String Equivalent	String
.ToBase64	Convert an array of Bytes to Base64	String
.ToBase64	Convert a string to Base64	String

More details or suggestions or for report an issue

MbarkTiesto@outlook.com

mbark@t3sto.com

Twitter

Linkedin