

Forum:[Hack Forums](#)

Thread: [\[C#\]\[VB .NET\]Customizable AES Class - Now with Byte Encryption!](#)

Ethereal - January 28, 2011, 3:42 pm

On request, I converted the C# class that I use to VB .NET. It probably isn't the neatest or most efficient, but it works, and that is what matters.

The main reason to use this over other classes is customizability. In my Notepad projects, I allow the user to customize their encryption with different FileTypes for increased flexibility.

Anyway, without further ado, here it is.

The closest source to the original that I found was [this](#), so current credits go to James Craig of GutGames.com. However, I don't think that his post was the original. I didn't remove any credits from my source.

Update 29/1/11: Small indentation, and Using added.

Update 30/1/11: VB .NET Byte encryption added, I'll port it to C# when I can be bothered :) If you get any errors, please notify me.

Update 11/03/11: C# updated on request. I never got round to it before.

Update 12/03/11: Simple bug fix that I missed. C# is now working.

C#

```
Code using System.IO;
using System.Text;
using System.Security.Cryptography;
using System;
```

```
public class AESEncryption
{
    //Translated by Ethereal of HackForums.net
    //Leave these credits - they aren't hurting anybody.

    #region "Static Functions"

    int ByteCount = 0;
    public static string Encrypt(string PlainText, string Password, string Salt, string HashAlgorithm, int PasswordIterations, string InitialVector, int KeySize)
    {
        try
        {
            if (string.IsNullOrEmpty(PlainText))
            {
                return "";
            }
            return Convert.ToBase64String(Encrypt(System.Text.UTF8Encoding.UTF8.GetBytes(PlainText), Password, Salt, HashAlgorithm, PasswordIterations, InitialVector, KeySize));
        }
        catch (Exception ex)
```

```

    {
        throw;
    }

    }

    public static byte[] Encrypt(byte[] Data, string Password, string Salt, string HashAlgorithm, int
PasswordIterations, string InitialVector, int KeySize)
    {
        try
        {
            byte[] InitialVectorBytes = System.Text.ASCIIEncoding.ASCII.GetBytes(InitialVector);
            byte[] SaltValueBytes = System.Text.ASCIIEncoding.ASCII.GetBytes(Salt);
            byte[] PlainTextBytes = Data;
            System.Security.Cryptography.PasswordDeriveBytes DerivedPassword = new
System.Security.Cryptography.PasswordDeriveBytes(Password, SaltValueBytes, HashAlgorithm,
PasswordIterations);
            byte[] KeyBytes = DerivedPassword.GetBytes(KeySize / 8);
            System.Security.Cryptography.RijndaelManaged SymmetricKey = new
System.Security.Cryptography.RijndaelManaged();
            SymmetricKey.Mode = System.Security.Cryptography.CipherMode.CBC;
            byte[] CipherTextBytes = null;
            using (System.Security.Cryptography.ICryptoTransform Encryptor =
SymmetricKey.CreateEncryptor(KeyBytes, InitialVectorBytes))
            {
                using (System.IO.MemoryStream MemStream = new System.IO.MemoryStream())
                {
                    using (System.Security.Cryptography.CryptoStream CStream = new CryptoStream(MemStream,
Encryptor, System.Security.Cryptography.CryptoStreamMode.Write))
                    {
                        CStream.Write(PlainTextBytes, 0, PlainTextBytes.Length);
                        CStream.FlushFinalBlock();
                        CipherTextBytes = MemStream.ToArray();
                        MemStream.Close();
                        CStream.Close();
                    }
                }
            }
            SymmetricKey.Clear();
            return CipherTextBytes;
        }
        catch (Exception ex)
        {
            throw;
        }
    }

    public static object Decrypt(string CipherText, string Password, string Salt, string HashAlgorithm,
int PasswordIterations, string InitialVector, int KeySize)
    {
        try
        {
            if (string.IsNullOrEmpty(CipherText))
            {

```

```

        return "";
    }
    return System.Text.Encoding.UTF8.GetString(Decrypt(Convert.FromBase64String(CipherText),
Password, Salt, HashAlgorithm, PasswordIterations, InitialVector, KeySize));
}
catch (Exception ex)
{
    throw;
}
}

    public static byte[] Decrypt(byte[] Data, string Password, string Salt, string HashAlgorithm, int
PasswordIterations, string InitialVector, int KeySize)
    {
        try
        {
            byte[] InitialVectorBytes = System.Text.Encoding.ASCII.GetBytes(InitialVector);
            byte[] SaltValueBytes = System.Text.Encoding.ASCII.GetBytes(Salt);
            byte[] CipherTextBytes = Data;
            System.Security.Cryptography.PasswordDeriveBytes DerivedPassword = new
System.Security.Cryptography.PasswordDeriveBytes(Password, SaltValueBytes, HashAlgorithm,
PasswordIterations);
            byte[] KeyBytes = DerivedPassword.GetBytes(KeySize / 8);
            System.Security.Cryptography.RijndaelManaged SymmetricKey = new
System.Security.Cryptography.RijndaelManaged();
            SymmetricKey.Mode = System.Security.Cryptography.CipherMode.CBC;
            byte[] PlainTextBytes = new byte[CipherTextBytes.Length + 1];
            using (System.Security.Cryptography.ICryptoTransform Decryptor =
SymmetricKey.CreateDecryptor(KeyBytes, InitialVectorBytes))
            {
                using (System.IO.MemoryStream MemStream = new
System.IO.MemoryStream(CipherTextBytes))
                {
                    using (System.Security.Cryptography.CryptoStream CStream = new CryptoStream(MemStream,
Decryptor, System.Security.Cryptography.CryptoStreamMode.Read))
                    {
                        MemStream.Close();
                        CStream.Close();
                    }
                }
                SymmetricKey.Clear();
                return PlainTextBytes;
            }
        }
        catch (Exception ex)
        {
            throw;
        }
    }
}
#endregion
}

```

Usage:

Code AESEncryption.Encrypt(txtBox.Text, "This.Is.Your.Pass", "This_Is_Your_Salt", "SHA1", 2,

```
"ThisIs16chars123", 256);  
Convert.ToString(AESEncryption.Decrypt(encryptedtext,"This.Is.Your.Pass", "This_Is_Your_Salt",  
"SHA1", 2, "ThisIs16chars123", 256));
```

VB .NET

Code Public Class AESEncryption

'Translated by Ethereum of HackForums.net

'Leave these credits - they aren't hurting anybody.

#Region "Static Functions"

Shared ByteCount As Integer = 0

Public Shared Function Encrypt(ByVal PlainText As String, ByVal Password As String, ByVal
Salt As String, ByVal HashAlgorithm As String, ByVal PasswordIterations As Integer, ByVal
InitialVector As String, ByVal KeySize As Integer) As String

Try

If String.IsNullOrEmpty(PlainText) Then

Return ""

End If

Return

Convert.ToBase64String(Encrypt(System.Text.UTF8Encoding.UTF8.GetBytes(PlainText),
Password, Salt, HashAlgorithm, PasswordIterations, InitialVector, KeySize))

Catch ex As Exception

Throw

End Try

End Function

Public Shared Function Encrypt(ByVal Data As Byte(), ByVal Password As String, ByVal Salt As
String, ByVal HashAlgorithm As String, ByVal PasswordIterations As Integer, ByVal InitialVector As
String, ByVal KeySize As Integer) As Byte()

Try

Dim InitialVectorBytes As Byte() = System.Text.ASCIIEncoding.ASCII.GetBytes(InitialVector)

Dim SaltValueBytes As Byte() = System.Text.ASCIIEncoding.ASCII.GetBytes(Salt)

Dim PlainTextBytes As Byte() = Data

Dim DerivedPassword As New System.Security.Cryptography.PasswordDeriveBytes(Password,
SaltValueBytes, HashAlgorithm, PasswordIterations)

Dim KeyBytes As Byte() = DerivedPassword.GetBytes(KeySize / 8)

Dim SymmetricKey As New System.Security.Cryptography.RijndaelManaged()

SymmetricKey.Mode = System.Security.Cryptography.CipherMode.CBC

Dim CipherTextBytes As Byte() = Nothing

Using Encryptor As System.Security.Cryptography.ICryptoTransform =
SymmetricKey.CreateEncryptor(KeyBytes, InitialVectorBytes)

Using MemStream As System.IO.MemoryStream = New System.IO.MemoryStream()

Using CStream As System.Security.Cryptography.CryptoStream = New

CryptoStream(MemStream, Encryptor, System.Security.Cryptography.CryptoStreamMode.Write)

CStream.Write(PlainTextBytes, 0, PlainTextBytes.Length)

CStream.FlushFinalBlock()

CipherTextBytes = MemStream.ToArray()

MemStream.Close()

CStream.Close()

End Using

End Using

End Using

```

        SymmetricKey.Clear()
        Return CipherTextBytes
    Catch ex As Exception
        Throw
    End Try
End Function

Public Shared Function Decrypt(ByVal CipherText As String, ByVal Password As String, ByVal
Salt As String, ByVal HashAlgorithm As String, ByVal PasswordIterations As Integer, ByVal
InitialVector As String, ByVal KeySize As Integer)
    Try
        If String.IsNullOrEmpty(CipherText) Then
            Return ""
        End If
        Return System.Text.Encoding.UTF8.GetString(Decrypt(Convert.FromBase64String(CipherText),
Password, Salt, HashAlgorithm, PasswordIterations, InitialVector, KeySize), 0, ByteCount)
    Catch ex As Exception
        Throw
    End Try
End Function

Public Shared Function Decrypt(ByVal Data As Byte(), ByVal Password As String, ByVal Salt As
String, ByVal HashAlgorithm As String, ByVal PasswordIterations As Integer, ByVal InitialVector As
String, ByVal KeySize As Integer) As Byte()
    Try
        Dim InitialVectorBytes As Byte() = System.Text.Encoding.ASCII.GetBytes(InitialVector)
        Dim SaltValueBytes As Byte() = System.Text.Encoding.ASCII.GetBytes(Salt)
        Dim CipherTextBytes = Data
        Dim DerivedPassword As New System.Security.Cryptography.PasswordDeriveBytes(Password,
SaltValueBytes, HashAlgorithm, PasswordIterations)
        Dim KeyBytes As Byte() = DerivedPassword.GetBytes(KeySize / 8)
        Dim SymmetricKey As New System.Security.Cryptography.RijndaelManaged()
        SymmetricKey.Mode = System.Security.Cryptography.CipherMode.CBC
        Dim PlainTextBytes(CipherTextBytes.Length) As Byte
        Using Decryptor As System.Security.Cryptography.ICryptoTransform =
SymmetricKey.CreateDecryptor(KeyBytes, InitialVectorBytes)
            Using MemStream As System.IO.MemoryStream = New
System.IO.MemoryStream(CipherTextBytes)
                Using CStream As System.Security.Cryptography.CryptoStream = New
CryptoStream(MemStream, Decryptor, System.Security.Cryptography.CryptoStreamMode.Read)
                    ByteCount = CStream.Read(PlainTextBytes, 0, PlainTextBytes.Length)
                    MemStream.Close()
                    CStream.Close()
                End Using
            End Using
            SymmetricKey.Clear()
            Return PlainTextBytes
        Catch ex As Exception
            Throw
        End Try
    End Function
#End Region
End Class

```

Usage:

```
Code AEEncryption.Encrypt(txtBox.Text, "This.Is.Your.Pass", "This_Is_Your_Salt", "SHA1", 2, "ThisIs16chars123", 256);
```

```
AEEncryption.Decrypt(encryptedtext,"This.Is.Your.Pass", "This_Is_Your_Salt", "SHA1", 2, "ThisIs16chars123", 256);
```

I will be updating this, and I hope that it is useful to you.

Oreex - January 28, 2011, 3:43 pm

Respect for this man

Very nice coded! thanks

Zer0Fact0r - January 28, 2011, 3:56 pm

You should credit the original creator of that code.

Ethereal - January 28, 2011, 3:57 pm

(Jan 28, 2011 04:56 PM)Zer0Fact0r Wrote: [\[CITE\]](#) You should credit the original creator of that code. If I can find the source, I will.

DragonHunter - January 28, 2011, 4:01 pm

(Jan 28, 2011 04:57 PM)Ethereal Wrote: [\[CITE\]](#) If I can find the source, I will.[original creator...](#) James Craig

Zer0Fact0r - January 28, 2011, 4:05 pm

(Jan 28, 2011 05:01 PM)DragonHunter Wrote: [\[CITE\]](#) [original creator...](#) James Craig

Whatever, it seems like fairly generic code. Although I think it originated from this:

<http://www.obviex.com/samples/Code.aspx?Source=EncryptionCS&Title=Symmetric%20Key%20Encryption&Lang=C%23>

Ethereal - January 28, 2011, 11:55 pm

(Jan 28, 2011 05:01 PM)DragonHunter Wrote: [\[CITE\]](#) [original creator...](#) James Craig

I already updated the OP.

However, the source I originally obtained the class from did not have the credits at the top, nor the Namespace or Imports. The original source was purely the class.

(Jan 28, 2011 05:05 PM)Zer0Fact0r Wrote: [\[CITE\]](#) Whatever, it seems like fairly generic code. Although I think it originated from this:

<http://www.obviex.com/samples/Code.aspx?Source=EncryptionCS&Title=Symmetric%20Key%20Encryption&Lang=C%23>

That looks likely.

However, this thread was not to discuss the origins of the C# implementation; It was to share a VB .NET version.

skxprive - January 29, 2011, 7:48 am

It doesn't work for my crypter,

an error about the string/byte, but encryption is nice thanks

hopefully you can fix it

Ethereal - January 29, 2011, 7:51 am

(Jan 29, 2011 08:48 AM)skxprive Wrote: [\[CITE\]](#) It doesn't work for my crypter, an error about the string/byte, but encryption is nice thanks hopefully you can fix it

Because it takes a string input, and returns one.

I will be adding a byte only overload. Whoever one-starred this thread is a douche.

skxprive - January 29, 2011, 7:54 am

(Jan 29, 2011 08:51 AM)Ethereal Wrote: [\[CITE\]](#) Because it takes a string input, and returns one.

I will be adding a byte only overload. Whoever one-starred this thread is a douche.
Yes that will be it

What do you mean? I will be glad if you could code it to make byte to byte and not only string
You didnt only translate it, but also improved it

Ethereal - January 29, 2011, 8:02 am

(Jan 29, 2011 08:54 AM)skxprive Wrote: [\[CITE\]](#) Yes that will be it

What do you mean? I will be glad if you could code it to make byte to byte and not only string
You didnt only translate it, but also improved it
I am adding an overload(which means you can enter both a string, or a byte[]) which will solve this problem.

skxprive - January 29, 2011, 8:06 am

Oh okay Thanks I now I know what it is :D

thats awesome

Oreex - January 29, 2011, 12:21 pm

ive been trying to make it. no success :) that failed miserably

Ethereal - January 30, 2011, 4:30 am

VB .NET code updated for Byte encryption. Please notify me of any errors you may encounter.

C# will be updated when I can be bothered :)

Oreex - January 30, 2011, 10:57 am

Really nice job, excellent

Rvaveo - January 31, 2011, 11:02 am

I really need this for my crypter.

Ethereal - January 31, 2011, 11:45 am

(Jan 31, 2011 12:02 PM)Ultimate Coder Wrote: [\[CITE\]](#) I really need this for my crypter.
Try it.

I think I know where there may be a bug, and so will test it again.

zxcv - March 11, 2011, 1:53 pm

Hi Ethereal,

Can you also do make it for Byte+string for the C# version? thanks
Btw this needs system.cryptography right? This will make a crypter stub more detected? Is AES possible without cryptography, so just like some XOR sources out here?

Ethereal - March 11, 2011, 1:55 pm

(Mar 11, 2011 02:53 PM)zx cv Wrote: [\[CITE\]](#) Hi Ethereum,

Can you also do make it for Byte+string for the C# version? thanks

Btw this needs system.cryptography right? This will make a crypter stub more detected? Is AES possible without cryptography, so just like some XOR sources out here?

I'll convert it in a sec.

Onemin.

zx cv - March 11, 2011, 1:59 pm

Thanks very much for the fast reaction.

Is the thing what I said true about cryptography import?

zx cv - March 11, 2011, 1:59 pm

Thanks very much for the fast reaction.

Is the thing what I said true about cryptography import?

Ethereal - March 11, 2011, 2:02 pm

Done.

You could write an algorithm without System.Security.Cryptography, but it's much faster, easier and of higher quality to use the official prebuilt class.

zx cv - March 11, 2011, 2:05 pm

Thanks for your input.

I have one last question(maybe basic): Do you recommend using this AES in a crypter? or will it become slow compared to Xor and RC4

zx cv - March 11, 2011, 2:05 pm

Thanks for your input.

I have one last question(maybe basic): Do you recommend using this AES in a crypter? or will it become slow compared to Xor and RC4

Ethereal - March 11, 2011, 2:13 pm

Yes, it's not that slow, and XOR/RC4 are detected(detect the encrypted offsets).

besimbicer - March 11, 2011, 4:23 pm

have a solutions please share

zx cv - March 11, 2011, 5:01 pm

I still get an error with the decryption, the error is:

Error 2 Cannot implicitly convert type 'object' to 'string'. An explicit conversion exists (are you missing a cast?)

So its only the decryption, I tried to fix it by adding Convert.ToString but it failed.
Help please :) ty

zxcv - March 11, 2011, 5:01 pm

I still get an error with the decryption, the error is:

Error 2 Cannot implicitly convert type 'object' to 'string'. An explicit conversion exists (are you missing a cast?)

So its only the decryption, I tried to fix it by adding Convert.ToString but it failed.
Help please :) ty

TheDubWubCub - March 11, 2011, 8:05 pm

i get a warning for VB, here's a screenshot, btw i also had an error with the cryptostream, but i fixed that (the warning was there before and after i changed it)

<http://img196.imageshack.us/img196/5124/311201180519pm.png>

Ethereal - March 11, 2011, 11:59 pm

(Mar 11, 2011 09:05 PM)21mods Wrote: [\[CITE\]](#) i get a warning for VB, here's a screenshot, btw i also had an error with the cryptostream, but i fixed that (the warning was there before and after i changed it)

<http://img196.imageshack.us/img196/5124/311201180519pm.png>

Ignore the warnings. It works just fine.

(Mar 11, 2011 06:01 PM)zxcv Wrote: [\[CITE\]](#) I still get an error with the decryption, the error is:

Error 2 Cannot implicitly convert type 'object' to 'string'. An explicit conversion exists (are you missing a cast?)

So its only the decryption, I tried to fix it by adding Convert.ToString but it failed.
Help please :) ty
One minute.

Lemme Debug it.

Ethereal - March 12, 2011, 12:36 am

(Mar 12, 2011 01:30 AM)((JUSTICE)) Wrote: [\[CITE\]](#) @ " Ethereal " GO learn SOem Programming Don't copy/paste stuffs from other Sites.I see many of your Codes are Copied from different places & u didn't bother to Give CRredits to them..SO gtfo...
Give examples, please.

Every other thread is code written by myself, and I wrote the overloads and debugged it.

Ethereal - March 12, 2011, 12:50 am

(Mar 12, 2011 01:45 AM)((JUSTICE)) Wrote: [\[CITE\]](#) I see your thread's,it is you who cant see / give credits .SO keep quite & agree ,no offense plz .
What do you mean, can't give credits?

It's all written by myself, not by someone else, so the credits go to me you fucking dick.

zxcv - March 12, 2011, 2:57 am

Justice fck off ok? He credited in OP and his other ones are made by himself so no credit is needed.
Simple as that

Btw this isn't original and he edited it.

@Ethereal Have you found the solution yet? (of the decryptin)

zxcv - March 12, 2011, 2:57 am

Justice fck off ok? He credited in OP and his other ones are made by himself so no credit is needed.
Simple as that
Btw this isn't original and he edited it.

@Ethereal Have you found the solution yet? (of the decryptin)

d3c0mpil3r - March 12, 2011, 3:13 am

I would use Rfc2898DeriveBytes Class instead of PasswordDeriveBytes.

Ethereal - March 12, 2011, 3:54 am

Fixed, I just needed to explicitly declare a byte[] instead of inferring it.

I also added the using's to save time ^_^

EDIT: Or not. Some annoying object casting problem.

zxcv - March 12, 2011, 5:19 am

Is it working properly? Thanks

Edit: its got exactly the same erro
Hopefully somebody knows the fix

bobolobo - March 12, 2011, 5:40 am

very good share :) .. was looking for this :)

zxcv - March 12, 2011, 8:59 am

Guess im using xor or something until this one is working.
(little Bump :))

zxcv - March 12, 2011, 8:59 am

Guess im using xor or something until this one is working.
(little Bump :))

Ethereal - March 13, 2011, 12:10 am

Ok, temporary fix.

For some reason, the decrypt method is returning an object, even though it should be converting it to a string in the source, so all you need is "Convert.ToString(AESEncryption.Decrypt());"

I think that the Byte[] seems to be working fine, but I'll have to look over it.

spike2147 - March 13, 2011, 7:11 am

at the vb.net useage is a fail, find it urself.

Also very nice class:D will definably use:D

Ethereal - March 13, 2011, 7:13 am

(Mar 13, 2011 07:11 AM)spike2147 Wrote: [\[CITE\]](#) at the vb.net useage is a fail, find it urself.

Also very nice class:D will definably use:D
Please elaborate.

I updated it.

Stenography - *March 30, 2011, 11:30 am*
awsome man thanks alot !!!

c6p0 - *March 30, 2011, 7:08 pm*
thanks you Ethereal i try this =)

H4ckThePlanet - *April 4, 2011, 10:07 pm*
Look's good very useful thanks for the share!

felicixus - *May 22, 2011, 6:10 am*
Thanks man! =) Your are great! But..
I have a question... For example we say we have this code
output = AESEncryption.Decrypt(file,blabla,blabla...)
I need to convert "output" to byte array, for using a runpe.
I've tried EVERYTHING! But nothing as worked, because the file get corrupted.
Please help me! =)

Ethereal - *May 22, 2011, 6:20 am*
There's a few issues with it that I need to sort out, however, if you want a byte[] encrypted and decrypted, use the byte[] overload.

However, the class has a little error, and I need to rewrite it.