Question
Menno Voerman · Nov 4, 2022

Difference of $SYSTEM.Encryption.AESCBCEncrypt with C# implementation

Hi All,

Hopefully someone can help me with this case. I need to encrypt a text(querystring) with an AES265 encryption. An other vendor is decrypting this information. I have a working class in C#. I've tried to build the same in Objectscript for the encrypt part but there's a missing link somewhere. **What's the difference between the C# and Objectscript implementation?**

Objectscript code (until now):

```
Class TEST.ENCRYPT
{

// Symmetric Keys sample to encrypt

ClassMethod DoAESCBCEncrypt() As %Status
{
 set key="pZR8qfrz7t47G+dboyJCH4NnJRrF+dJbvxq37y/cLUo="
 set iv=##class(%PopulateUtils).StringMin(16,16)
 Write "Key="_key,!
 Write "IV="_iv,!
 Set ivBase64 = $SYSTEM.Encryption.Base64Encode(iv)
 Write "IVBase64="_ivBase64,!

 set plaintext=
"This is just an encryption test with AES256, blocksize 128, padding PKCS7, mode, CBC
 with an IV of 16 bytes"
 Set text=$ZCONVERT(plaintext,"O","UTF8")
 Write "Text UTF-8: "_text,!

 Set encrypted=$SYSTEM.Encryption.AESCBCEncrypt(text,key,iv)
 Set EncryptedBase64=$SYSTEM.Encryption.Base64Encode(encrypted)
 Write "EncryptedBase64: "_EncryptedBase64,!

 Set encryptedComplete = ivBase64_EncryptedBase64
 Write "EncryptedBase64WithIV: "_encryptedComplete,!

 Set ciphertext = $$$URLENCODE(encryptedComplete)

 write "URL Encoded:"_ciphertext,!
 return $$$OK
}

}
```

This give this as example output:

```
Debugger executing '##class(TEST.ENCRYPT).DoAESCBCEncrypt()'
Executing ##class(TEST.ENCRYPT).DoAESCBCEncrypt()
Key=pZR8qfrz7t47G+dboyJCH4NnJRrF+dJbvxq37y/cLUo=
IV=fTwrLaVlAerJf2KB
IVBase64=ZlR3ckxhVmxBZXJKZlpLQg==
Text UTF-8: This is just an encryption test with AES256, blocksize 128, padding PKCS7, mode, CBC with an IV of 16 bytes
EncryptedBase64: 8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r+KZ+ThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb
D8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb+N4eMY6jg05HkGejXlIoEo8Q==
EncryptedBase64WithIV: ZlR3ckxhVmxBZXJKZlpLQg==8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r+KZ+ThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb
D8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb+N4eMY6jg05HkGejXlIoEo8Q==
URL Encoded:ZlR3ckxhVmxBZXJKZlpLQg%3D%3D8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r%2BKZ%2BThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb%0D%0AD8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb%2BN4eMY6jg05HkGejXlIoEo8Q%3D%3D
```

URL Encoded: ZlR3ckxhVmxBZXJKZlpLQg%3D%3D8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r%2BKZ%2BThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb%0D%0AD8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb%2BN4eMY6jg05HkGejXlIoEo8Q%3D%3D

When decrypting with the C# code, I get the following eror.

```
ICryptoTransform cryptoTransformDecrypt = DecryptProvider.CreateDecryptor();
byte[] numArray = Convert.FromBase64String(toDecryptWithoutIV);
string decrypted = Encoding.UTF8.GetString(cryptoTransformDecrypt.TransformFinalBlock(numArray, 0, (int)numArray.Length));

Console.WriteLine("Decrypted: " + decrypted);
Console.WriteLine("######## DECRYPTION END ########");

'erences
tic AesCryptoServiceProvider CreateCryptoProvider(byte[] HashedKey, byte[] IV)

return new AesCryptoServiceProvider()
{
```

Exception Unhandled

System.Security.Cryptography.CryptographicException: 'Padding is invalid and cannot be removed.'

View Details | Copy Details | Start Live Share session...

Exception Settings

I see a few things. When running the decrypting code the IV looks fine, also the "To Decrypt without IV" decrypt looks fine (it matches the output of the Objectscript implementation). So I think there's a difference in implementation of the encryption itself but how to find the difference?

```
######## DECRYPTION START ########
To Decrypt: ZlR3ckxhVmxBZXJKZlpLQg%3D%3D8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r%2BKZ%2BThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb
%0D%0AD8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb%2BN4eMY6jg05HkGejXlIoEo8Q%3D%3D
De-Escaped: ZlR3ckxhVmxBZXJKZlpLQg==8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r+KZ+ThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb
D8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb+N4eMY6jg05HkGejXlIoEo8Q==
IV String: ZlR3ckxhVmxBZXJKZlpLQg==
To Decrypt without IV: 8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r+KZ+ThSzVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb
D8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG74lhAjtvb+N4eMY6jg05HkGejXlIoEo8Q==
######## DECRYPTION END ########
```

See complete enrypt/decrypt C# code below.

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

class Program
{
    static void Main(string[] args)
    {
  string toEncrypt =
"This is just an encryption test with AES256, blocksize 128, padding PKCS7, mode, CBC
 with an IV of 16 bytes";
  string hashedKey = "pZR8qfrz7t47G+dboyJCH4NnJRrF+dJbvxq37y/cLUo=";
  Console.WriteLine("SECRET KEY / HASHED KEY: "+hashedKey);
  Console.WriteLine("");

  Console.WriteLine("What to encrypt/decrypt: " + toEncrypt);
  Console.WriteLine("");

  //Ecnryption Parts
  Console.WriteLine("######## ENCRYPTION START ########");

  Console.WriteLine("To Encrypt: " + toEncrypt);

  //create IV
```

```
  byte[] IV = new byte[16];
  (new Random()).NextBytes(IV);

  AesCryptoServiceProvider cryptoProvider = CreateCryptoProvider(Convert.FromBase64St
ring(hashedKey),IV);

  ICryptoTransform cryptoTransform = cryptoProvider.CreateEncryptor();
  byte[] bytes = Encoding.UTF8.GetBytes(toEncrypt);
  string encryptedUnformatted = Convert.ToBase64String(cryptoTransform.TransformFinal
Block(bytes, 0, (int)bytes.Length));
  Console.WriteLine("encryptedUnformatted: " + encryptedUnformatted);

  //add Random IV
  string randomIVBase64 = Convert.ToBase64String(IV);
  Console.WriteLine("randomIVBase64: " + randomIVBase64);

  encryptedUnformatted = string.Concat(randomIVBase64, encryptedUnformatted);
  Console.WriteLine("With Random IV: " + encryptedUnformatted);

  //escape string
  string encryptedFormatted = Uri.EscapeDataString(encryptedUnformatted);
  Console.WriteLine("encryptedFormatted: " + encryptedFormatted);

  Console.WriteLine("######## ENCRYPTION END ########");
  Console.WriteLine("");
  Console.WriteLine("############################################################");
  Console.WriteLine("");
  Console.WriteLine("######## DECRYPTION START ########");

  //Decryption Part
  string toDecrypt = encryptedFormatted;

//toDecrypt = "ZlR3ckxhVmxBZXJKZlpLQg%3D%3D8TvyqZNAFsv7Nb0HYkGIfrPmUOkeQ2r%2BKZ%2BThS
zVd1tqoyziSpRJsjQP/pEnAzRY5HzybCdFBDtb%0D%0AD8wNTNv1OFcXsHqk/GOhce1xdPlyJFg85llLUfBIG
74lhAjtvb%2BN4eMY6jg05HkGejXlIoEo8Q%3D%3D";
  Console.WriteLine("To Decrypt: " + toDecrypt);

  //de-escape string
  string deEscape=Uri.UnescapeDataString(toDecrypt);
  Console.WriteLine("De-Escaped: " + deEscape);

  //get IV
  string ivString = deEscape.Substring(0, 24);
  Console.WriteLine("IV String: " + ivString);

  byte[] IVDecrypt= Convert.FromBase64String(ivString);

  string toDecryptWithoutIV = deEscape.Substring(24);
  Console.WriteLine("To Decrypt without IV: " + toDecryptWithoutIV);

  AesCryptoServiceProvider DecryptProvider = CreateCryptoProvider(Convert.FromBase64S
tring(hashedKey), IVDecrypt);

  ICryptoTransform cryptoTransformDecrypt = DecryptProvider.CreateDecryptor();
  byte[] numArray = Convert.FromBase64String(toDecryptWithoutIV);
  string decrypted = Encoding.UTF8.GetString(cryptoTransformDecrypt.TransformFinalBlo
ck(numArray, 0, (int)numArray.Length));

  Console.WriteLine("Decrypted: " + decrypted);
```

```
  Console.WriteLine("######## DECRYPTION END ########");
 }
 static AesCryptoServiceProvider CreateCryptoProvider(byte[] HashedKey, byte[] IV)
 {
  return new AesCryptoServiceProvider()
  {
   KeySize = 256,
   BlockSize = 128,
   Key = HashedKey,
   IV = IV,
   Padding = PaddingMode.PKCS7,
   Mode = CipherMode.CBC
  };
 }
}
```

#Encryption #ObjectScript #Caché
Product version: IRIS 2021.2
$ZV: IRIS for Windows (x86-64) 2021.2 (Build 649U) Thu Jan 20 2022 08:46:32 EST [HealthConnect:3.4.0] [HealthConnect:3.4.0]

Source URL:https://community.intersystems.com/post/difference-systemencryptionaescbcencrypt-c-implementation