BankMobile

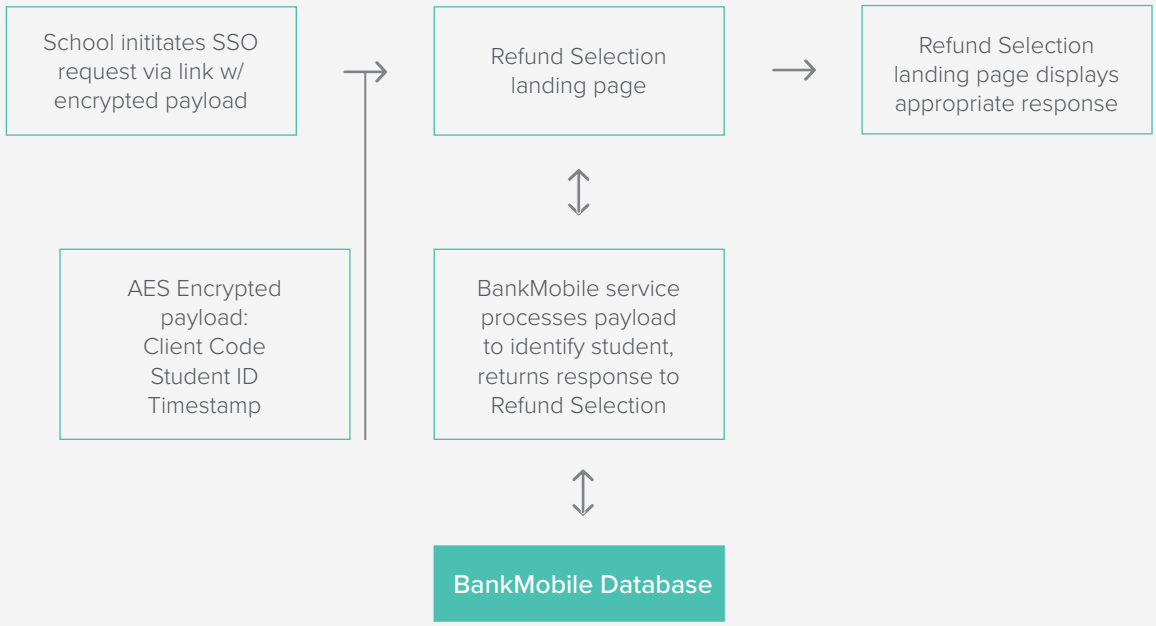
Refund Selection Single Sign On

OVERVIEW

The BankMobile Refund Selection Single Sign-On (SSO) solution offers an integrated user authentication system for students to make their initial refund selection preferences. The intention of this service is to streamline the student's experience and authenticate the student via their student portal credentials.

This service eliminates the need for the "Personal Code" and allows for quicker access to the student refund selection process. The school acts as the identity provider and BankMobile uses the data provided to deploy the solution as documented below.

The overall approach for our SSO solution is illustrated in the following diagram:



©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

BankMobile

As illustrated above, the school generates an SSO request link with an encrypted payload string. Students who click this link will be directed to the BankMobile Refund Selection application, where a dedicated service processes the payload string and uses the included values to identify the student. Once having identified the student, and assuming no other errors or the expiration of the token, the Refund Selection application will display an appropriate response and "sign the user on" by directing them to the existing profile page. This allows students to circumvent the usual Refund Selection welcome page on which they are prompted to enter a Personal Code in order to identify them.

SCHOOL IMPLEMENTATION

The BankMobile SSO solution presumes the existence of a shared key/phrase that will be used to encrypt the request payload string at the school and decrypt said string within the Refund Selection application. It is critically important that schools implementing SSO make every effort to ensure the security of the production shared key. We will be utilizing AES encryption for our initial implementation, with a key size of 128 bits.

The request payload string will have the following format: clientCode&studentid&timestamp

The constituent parts of the payload string are defined as follows:

clientCode: School identifier provided by BankMobile

studentid: School's identifier for the student, which is also the ID1 value that is configured in the Customer Import process. This ID is unique per student at the school.

timestamp: Capture the current timestamp and convert to UTC timezone, formatted as MM/dd/yyyy HH:mm:ss.

The above request payload shall be AES encrypted, then used as a parameter to the incoming HTTPS request URL to the Refund Selection SSO landing page. The parameter should be appended to the SSO landing base URL with a parameter name of "token". Additionally, a parameter named "clientcode" should be appended to the end of the URL following the token parameter. The clientcode parameter value is the same as that used in the payload string prior to encryption. Note this is an HTTP GET request,

www.refundselection.com/#/landing?token=ENCRYPTED PAYLOAD&clientcode=CLIENT_CODE

ENCRYPTION SCHEME

Prior to implementation, BankMobile shall provide a 32-character random hexadecimal string to be used as the shared key between systems for encryption and decryption of the token payload. Further technical details of the AES encryption scheme are as follows:

©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

Encryption Algorithm: AES

Key Size: 128 bit

Mode: ECB

Padding: PKCS5

PAYLOAD STRING & REQUEST URL EXAMPLE

Given the following values:

clientCode: some_university
studentid: 12345678
shared key: 0123456789ABCDEF0123456789ABCDEF

Capture the timestamp:

Determine current timestamp: 01/09/2017 12:14:15
Convert to UTC timezone formatted as “MM/dd/yyyy HH:mm:ss”: 01/09/2017 17:14:15

Concatenate clientCode, studentid and timestamp, using ampersand symbol as separator:

some_university&12345678&01/09/2017 17:14:15

AES encrypt the resulting string using the shared key, then represent it as a string of hexadecimal digits for use as the token parameter value:

www.refundselection.com/#landing?token=298C3A4DE0FA95D02CAA07F24F7F234BC78EB18F98B DD2 CBC20AC18FB9F39AF84B0CE8A37773DEBEDC9D74AB1C2D8E5D&clientcode=some_university

Redirect the user to the SSO destination passing the result of the step above in the query string using the query string parameter “token” and appending the “clientcode” parameter with the appropriate value:

www.refundselection.com/#/landing?token=298C3A4DE0FA95D02CAA07F24F7F234BC78EB-18F98BDD2CBC20AC18FB9F39AF84B0CE8A37773DEBEDC9D74AB1C2D8E5D&clientcode=some_university

©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

REFUND SELECTION ERROR HANDLING

The following is a list of use cases we will handle for this implementation:

1. Inability to decrypt the payload, or missing value in payload: Prompt the user about configuration error and ask them to reach out to a school administrator to learn about the status of their refund.

2. clientCode does not match a configured client school: Prompt the user about configuration error and ask them to reach out to a school administrator to learn about the status of their refund.

3. studentid does not match a student at client school: Prompt the user about configuration error and ask them to reach out to a school administrator to learn about the status of their refund.

4. expiration is past: Prompt the user about session timeout and ask them to log back in to school's portal and try again

5. Student is identified but in an improper state: Prompt user to call Care

6. Student is identified but already in an active state: Prompt user to log in

7. Core banking services are not available: Prompt the user about system being unavailable and ask them to try again later.

AES TOKEN ENCRYPTION JAVA SAMPLE CODE

Test Class

```
package com.bankmobile.service.sso;

import org.junit.Test;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.TimeZone;

import static org.junit.Assert.assertEquals;

/**
 * <pre>
 * Test class to exercise the SSOAESExample class.
 * </pre>
*/
public class TestSSOAESExample {
   private static final String SHARED_KEY = "0123456789ABCDEF0123456789ABCDEF";

  @Test
   public void testSSOAESEncryptThenDecrypt( ) {
            String clientcode = "schoolcode";
            String studentid = "12345678";
            Calendar timestampCal = new GregorianCalendar(2017, 0, 9, 17, 29, 15);
            TimeZone utc = TimeZone.getTimeZone("UTC");
            SimpleDateFormat smp = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
            smp.setTimeZone(utc);
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
            String token = SSOAESExample.buildTokenString(clientcode, studentid, smp.format
(timestampCal.getTime()));
            String encrypted = SSOAESExample.aesEncrypt(SHARED_KEY, token);
            String decrypted = SSOAESExample.aesDecrypt(SHARED_KEY, encrypted);
            assertEquals("Token string prior to encryption should equal decrypted token string", token, decrypted);
            System.out.println("Original token string: " + token);
            System.out.println("Encrypted token string: " + encrypted);
            System.out.println("Decrypted token string: " + decrypted);
    }
}
```

IMPLEMENTATION

```
package com.bankmobile.service.sso;

mport javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

/**
 *  <pre>
 *  Utility functions for BankMobile Refund Selection SSO (SSO).
 *  </pre>
*/
public class SSOAESExample {
    private static final String CIPHER_TYPE_AES = "AES/ECB/PKCS5Padding";
    private static final String ENCODING_TYPE_AES = "AES";

    /**
     * AES encrypt message string using sharedKey.
     * @param sharedKey - This is the shared encryption key between systems
     * @param token - The token to encrypt
    */
    protected static String aesEncrypt(String sharedKey, String token) {
            String encryptedString = null;
            try {
                SecretKey key = new SecretKeySpec(asBinary(sharedKey), ENCODING_TYPE_AES);
                Cipher cipher = Cipher.getInstance(CIPHER_TYPE_AES);
                cipher.init(Cipher.ENCRYPT_MODE, key);
                byte[] encrypted = cipher.doFinal(token.getBytes( ));
                encryptedString =  asHex(encrypted).toUpperCase( );
            } catch

(NoSuchAlgorithmException|NoSuchPaddingException|InvalidKeyException|IllegalBlockSizeException|
BadPaddingException e) {
                e.printStackTrace( );
             }
             return encryptedString;
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
    }
    /**
     * AES decrypt message string using sharedKey.
     * @param sharedKey - This is the shared encryption key between systems
     * @param token - The token to deccrypt
     */
    protected static String aesDecrypt(String sharedKey, String token) {
        String decryptedString = null;
        try  {
                SecretKey key = new SecretKeySpec(asBinary(sharedKey), ENCODING_TYPE_AES);
                Cipher cipher = Cipher.getInstance(CIPHER_TYPE_AES);
                cipher.init(Cipher.DECRYPT_MODE, key);
                byte[] decrypted = cipher.doFinal(asBinary(token));
                decryptedString = new String(decrypted);
        } catch

(NoSuchAlgorithmException|NoSuchPaddingException|InvalidKeyException|IllegalBlockSizeException|BadPaddingException e) {
                e.printStackTrace( );
        }
        return decryptedString;
}
(NoSuchAlgorithmException|NoSuchPaddingException|InvalidKeyException|IllegalBlockSizeException|BadPaddingException e)
{e.printStackTrace( ); }

        return encryptedString;
    }

    /**

     * AES decrypt message string using sharedKey.
     * @param sharedKey - This is the shared encryption key between systems
     * @param token - The token to deccrypt

     */

    protected static String aesDecrypt(String sharedKey, String token) {
    String decryptedString = null;

        try {

            SecretKey key = new SecretKeySpec(asBinary(sharedKey), ENCODING_TYPE_AES);
            Cipher cipher = Cipher.getInstance(CIPHER_TYPE_AES);
            cipher.init(Cipher.DECRYPT_MODE, key);
            byte[ ] decrypted = cipher.doFinal(asBinary(token));

/**
     * Utility function to convert bytes in to hex strings

     * @param buf
     * @return - hex string
     */
    private static String asHex( byte buf[ ] )
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
    {
        StringBuffer strbuf = new StringBuffer(buf.length * 2);
        for (int i = 0; i < buf.length; i++)  {
            if (((int) buf[i] & 0xff) < 0x10) {
                strbuf.append("0");
            }
            strbuf.append(Long.toString((int) buf[i] & 0xff, 16));
        }
        return strbuf.toString( );
    }
    /**
     * Convert hex string into corresponding bytes.
     * @param hexChars
     * @return bytes
     */
    private static byte[] asBinary( String hexChars )
    {
        // Find the length in bytes to be made. Each hexChar is 4 bits.
        int numBytes = hexChars.length() / 2;
        byte[] binary = new byte[ numBytes ];
        for ( int x=0; x<numBytes; x++ )
        {
                String temp = "0x" + hexChars.charAt( x*2 ) + hexChars.charAt( x*2+1 );
                binary[ x ] = Integer.decode( temp ).byteValue( );
        }
        return binary;
    }

    protected static String buildTokenString(String clientCode, String studentID, String timestamp) {
        StringBuffer tokenString = new StringBuffer();
        tokenString.append(clientCode);
        tokenString.append("&");
        tokenString.append(studentID);
        tokenString.append("&");
        tokenString.append(timestamp);
        return tokenString.toString( );
    }
}
```

AES TOKEN ENCRYPTION C# SAMPLE CODE

Test Class

```
using System;

class Program
{
    static void Main()
    {
        string clientCode = "some_university";
        string sharedKey = "0123456789ABCDEF0123456789ABCDEF";
        string studentId = "12345678";

        SampleUsage(clientCode, sharedKey, studentId);
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
        TestEncryption(clientCode, sharedKey, studentId);
    }

    static void SampleUsage(string clientCode, string sharedKey, string studentId)
    {
        // Instantiate the SSOAESURLConstructor class with the the specified client code and shared key
        SSOAESURLConstructor constructor = new SSOAESURLConstructor(clientCode, sharedKey);

        // Generate a URL for the given student
        string url = constructor.URLForStudent(studentId);

        Console.WriteLine("The URL for student {0} is: \n{1}", studentId, url);
    }

    static void TestEncryption(string clientCode, string sharedKey, string studentId)
    {
        Console.WriteLine("==============================");
        Console.WriteLine("Testing SSOAESURLConstructor");
        Console.WriteLine("Client code: " + clientCode);
        Console.WriteLine("Shared key:  " + sharedKey);
        Console.WriteLine("Student ID:  " + studentId);
        Console.WriteLine("==============================");

        SSOAESURLConstructor constructor = new SSOAESURLConstructor(clientCode, sharedKey);

        string timestamp = DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");
        string token = constructor.ConstructToken(clientCode, studentId, timestamp);

        string encrypted = SSOAESEncryption.Encrypt(sharedKey, token);
        string decrypted = SSOAESEncryption.Decrypt(sharedKey, encrypted);

        Console.WriteLine("Constructed token: " + token);
        Console.WriteLine("Encrypted token:   " + encrypted);
        Console.WriteLine("Decrypted token:   " + decrypted);

        if (token == decrypted)
            Console.WriteLine("The decrypted string matches.");
        else
            Console.WriteLine("The decrypted string does not match. Something has gone wrong.");
    }
}
```

Helper Class

```
using System;


public class SSOAESURLConstructor
{
    private readonly string key;
    private readonly string code;

    public SSOAESURLConstructor(string clientCode, string sharedKey)
    {
        code = clientCode;
        key = sharedKey;
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
    }

    /// <summary>
    /// Returns an SSO refund URL for a student with the specified ID.
    /// </summary>
    /// <returns>A student refund URL.</returns>
    /// <param name="studentId">A student identifier.</param>
    public string URLForStudent(string studentId)
    {
        string timestamp = DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");

        string token = ConstructToken(code, studentId, timestamp);

        string encryptedToken = SSOAESEncryption.Encrypt(key, token);

        string url = "refundselection.com/#landing";
        url += "?token=" + encryptedToken;
        url += "&clientCode=" + code;

        return url;
    }

    /// <summary>
    /// Constructs a token string.
    /// </summary>
    /// <returns>The token string.</returns>
    /// <param name="clientCode">Client code.</param>
    /// <param name="studentId">Student identifier.</param>
    /// <param name="timestamp">Timestamp.</param>
    public string ConstructToken(string clientCode, string studentId, string timestamp)
    {
        return clientCode + "&" + studentId + "&" + timestamp;
    }
}
```

Implementation

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

public static class SSOAESEncryption
{
    /// <summary>
    /// Encrypts a string with the specified shared key.
    /// </summary>
    /// <returns>An encrypted string.</returns>
    /// <param name="key">Shared key to encrypt with.</param>
    /// <param name="unencrypted">String to encrypt.</param>
    public static string Encrypt(string key, string unencrypted)
    {
        byte[] encrypted = EncryptStringToBytes_Aes(unencrypted, StringToBytes(key));

        return BytesToString(encrypted);
    }
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
/// <summary>
/// Decrypts a string with the specified shared key.
/// </summary>
/// <returns>A decrypted string.</returns>
/// <param name="key">Shared key to decrypt with.</param>
/// <param name="encrypted">String to decrypt.</param>
public static string Decrypt(string key, string encrypted)
{
    byte[] bytes = StringToBytes(encrypted);

    return DecryptStringFromBytes_Aes(bytes, StringToBytes(key));
}

/// <summary>
/// Converts a byte array to a hex string.
/// </summary>
/// <returns>A hex string.</returns>
/// <param name="bytes">A byte array.</param>
private static string BytesToString(byte[] bytes)
{
    return BitConverter.ToString(bytes).Replace("-", String.Empty);
}

/// <summary>
/// Converts a hex string to a byte array.
/// </summary>
/// <returns>A byte array.</returns>
/// <param name="value">A hex string.</param>
private static byte[] StringToBytes(string value)
{
    if (value.Length % 2 == 1)
        throw new Exception("Hex string must have an even number of digits ");

    byte[] arr = new byte[value.Length >> 1];

    for (int i = 0; i < value.Length >> 1; ++i)
        arr[i] = (byte)((HexValue(value[i << 1]) << 4) + (HexValue(value[(i << 1) + 1])));

    return arr;
}

/// <summary>
/// Returns the hex value for a specified character
/// </summary>
private static int HexValue(char c)
{
    if (c >= '0' && c <= '9')
        return c - '0';
    else if (c >= 'A' && c <= 'F')
        return c - 'A' + 10;
    else if (c >= 'a' && c <= 'f')
        return c - 'a' + 10;

    throw new ArgumentException("Invalid hex digit: " + c);
}
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
/// <summary>
/// AES encrypts a string.
/// </summary>
private static byte[] EncryptStringToBytes_Aes(string plainText, byte[] Key)
{
    // Check arguments.
    if (string.IsNullOrEmpty(plainText))
        throw new ArgumentNullException("Missing token to encrypt");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Missing shared key");

    byte[] encrypted;

    // Create an AesManaged object with the specified key
    using (AesManaged aesAlg = new AesManaged())
    {
        aesAlg.Padding = PaddingMode.PKCS7; // PKCS5 and PKCS7 use the same padding algorithm
        aesAlg.Mode = CipherMode.ECB; // Non-default
        aesAlg.Key = Key;

        // Create a decrytor to perform the stream transform
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

        // Create the streams used for encryption
        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    // Write all data to the stream.
                    swEncrypt.Write(plainText);
                }
                encrypted = msEncrypt.ToArray();
            }
        }
    }

    // Return the encrypted bytes from the memory stream
    return encrypted;
}

/// <summary>
/// AES decrypts a byte array.
/// </summary>
private static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key)
{
    // Check arguments.
    if (cipherText == null || cipherText.Length <= 0)
        throw new ArgumentNullException("Missing text to decrypt");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Missing shared key");

    // Declare the string used to hold the decrypted text.
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
        string plaintext = null;

        // Create an AesManaged object with the specified key
        using (AesManaged aesAlg = new AesManaged())
        {
            aesAlg.Padding = PaddingMode.PKCS7; // PKCS5 and PKCS7 use the same padding algorithm
            aesAlg.Mode = CipherMode.ECB; // Non-default
            aesAlg.Key = Key;

            // Create a decrytor to perform the stream transform
            ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

            // Create the streams used for decryption
            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                    {
                        // Read the decrypted bytes from the decrypting stream and place them in a string
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }

        }

        return plaintext;
    }
}

/// <summary>
    /// Decrypts a string with the specified shared key.
    /// </summary>
    /// <returns>A decrypted string.</returns>
    /// <param name="key">Shared key to decrypt with.</param>
    /// <param name="encrypted">String to decrypt.</param>
    public static string Decrypt(string key, string encrypted)
    {
        byte[] bytes = StringToBytes(encrypted);

        return DecryptStringFromBytes_Aes(bytes, StringToBytes(key));
    }

    /// <summary>
    /// Converts a byte array to a hex string.
    /// </summary>
    /// <returns>A hex string.</returns>
    /// <param name="bytes">A byte array.</param>
    private static string BytesToString(byte[] bytes)
    {
        return BitConverter.ToString(bytes).Replace("-", String.Empty);
    }

    /// <summary>
    /// Converts a hex string to a byte array.
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
/// </summary>
/// <returns>A byte array.</returns>
/// <param name="value">A hex string.</param>
private static byte[] StringToBytes(string value)
{
    if (value.Length % 2 == 1)
        throw new Exception("Hex string must have an even number of digits ");

    byte[] arr = new byte[value.Length >> 1];

    for (int i = 0; i < value.Length >> 1; ++i)
        arr[i] = (byte)((HexValue(value[i << 1]) << 4) + (HexValue(value[(i << 1) + 1])));

    return arr;
}

/// <summary>
/// Returns the hex value for a specified character
/// </summary>
private static int HexValue(char c)
{
    if (c >= '0' && c <= '9')
        return c - '0';
    else if (c >= 'A' && c <= 'F')
        return c - 'A' + 10;
    else if (c >= 'a' && c <= 'f')
        return c - 'a' + 10;

    throw new ArgumentException("Invalid hex digit: " + c);
}

/// <summary>
/// AES encrypts a string.
/// </summary>
private static byte[] EncryptStringToBytes_Aes(string plainText, byte[] Key)
{
    // Check arguments.
    if (string.IsNullOrEmpty(plainText))
        throw new ArgumentNullException("Missing token to encrypt");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Missing shared key");

    byte[] encrypted;

    // Create an AesManaged object with the specified key
    using (AesManaged aesAlg = new AesManaged())
    {
        aesAlg.Padding = PaddingMode.PKCS7; // PKCS5 and PKCS7 use the same padding algorithm
        aesAlg.Mode = CipherMode.ECB; // Non-default
        aesAlg.Key = Key;

        // Create a decrytor to perform the stream transform
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

        // Create the streams used for encryption
        using (MemoryStream msEncrypt = new MemoryStream())
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved

```
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    // Write all data to the stream.
                    swEncrypt.Write(plainText);
                }
                encrypted = msEncrypt.ToArray();
            }
        }
    }

    // Return the encrypted bytes from the memory stream
    return encrypted;
}
 /// <summary>
/// AES decrypts a byte array.
/// </summary>
private static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key)
{
    // Check arguments.
    if (cipherText == null || cipherText.Length <= 0)
        throw new ArgumentNullException("Missing text to decrypt");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Missing shared key");

    // Declare the string used to hold the decrypted text.
    string plaintext = null;

    // Create an AesManaged object with the specified key
    using (AesManaged aesAlg = new AesManaged())
    {
        aesAlg.Padding = PaddingMode.PKCS7; // PKCS5 and PKCS7 use the same padding algorithm
        aesAlg.Mode = CipherMode.ECB; // Non-default
        aesAlg.Key = Key;

        // Create a decrytor to perform the stream transform
        ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

        // Create the streams used for decryption
        using (MemoryStream msDecrypt = new MemoryStream(cipherText))
        {
            using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
            {
                using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                {
                    // Read the decrypted bytes from the decrypting stream and place them in a string
                    plaintext = srDecrypt.ReadToEnd();
                }
            }
        }

    }
    return plaintext;
}
```


©2018 BankMobile Technologies, a wholly owned subsidiary of Customers Bank. All Rights Reserved