



CryptoTools 3.0 Tutorial



Table of contents

TABLE OF CONTENTS	2
INTRODUCTION.....	4
CRYPTOTOOLS TUTORIAL FOR C	4
BEFORE STARTING	4
THE CODE	4
IN ONE PIECE	6
CRYPTOTOOLS TUTORIAL FOR C++.....	7
BEFORE STARTING	7
THE CODE	7
IN ONE PIECE	8
CRYPTOTOOLS TUTORIAL FOR COM/C++.....	10
BEFORE STARTING	10
THE CODE	10
IN ONE PIECE	11
CRYPTOTOOLS TUTORIAL FOR COM/VB.....	13
BEFORE STARTING	13
THE CODE	13
IN ONE PIECE	14
CRYPTOTOOLS TUTORIAL FOR COM/VBSCRIPT & ASP	15
BEFORE STARTING.....	15
THE CODE.....	16
IN ONE PIECE	17
<i>CryptoCOM.wsf</i>	17
<i>CryptoCOM.asp</i>	17
CRYPTOTOOLS TUTORIAL FOR COM/JSCRIPT.....	19
BEFORE STARTING.....	19
IN ONE PIECE	19
CRYPTOTOOLS TUTORIAL FOR .NET/C#.....	20
BEFORE STARTING	20
THE CODE	20
IN ONE PIECE	20
CRYPTOTOOLS TUTORIAL FOR .NET/VB.NET	21
BEFORE STARTING	21



THE CODE	21
IN ONE PIECE	22
CRYPTOTOOLS TUTORIAL FOR JAVA	22
BEFORE STARTING	22
THE CODE	22
IN ONE PIECE	23



Introduction

This document will guide you through the integration of CryptoTools with your application. A short piece of code will be provided for all major languages supported by CryptoTools.

CryptoTools tutorial for C

Before starting

To start a new C project using the CryptoTools library, you must first configure your project or makefile so that it links with the CryptoDLL.lib library located in the “[PROGRAM FILES]CryptoTools\CryptoDLL\bin” directory. You will also need to include the header file corresponding to the algorithm you want to use. The four header files available are ICCryptoDES.h, ICCryptoTripleDES.h, ICCryptoBase64.h and ICCryptoMD5.h.

```
#include "ICryptoDES.h"  
#include "ICryptoBase64.h"  
#include "ICryptoTripleDES.h"  
#include "ICryptoMD5.h"
```

The CryptoDLL.dll file must be located in your application’s working directory or in a Windows execution PATH folder. By default, the installation program installs a copy of the dll in the Windows System directory.

The code

The C programming language is the only non-object oriented language supported by CryptoTools. The API has been designed using “contexts” that can be interpreted as instances of CryptoTools objects.

Therefore, the first thing to do is to create a new context for the algorithm you want to use. Let’s say that we want to encrypt using DES and then encode the binary result into base64.

```
int iContext = 0;  
...  
iContext = CCryptoDES_CreateContext();
```

All further calls to this instance of the DES encryption object must be done by passing the context as first parameter.



Now, let's set the encryption key. The encryption key for the DES algorithm is an 8 byte key. To set a new key for encryption or decryption, call the `CryptoDES_SetKey` function.

```
BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
...
CCryptoDES_SetKey( iContext, _pbyKey );
```

It is now time to encrypt our data. The data we want to encrypt is an array of bytes.

```
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
...
CCryptoDES_Encrypt( iContext, _pbyDataIn, 8 );
```

You can also encrypt plain text using the same function call:

```
char* _strMyData = "Hello World!";
...
CCryptoDES_Encrypt( iContext, _strMyData, strlen(_strMyData) );
```

If the function returns with 0, the encryption succeeded; otherwise, an error code is returned.

Now, let's get the result. Since DES works on 64 bit blocks, the result length of the encrypted data will always be a multiple of 8 bytes.

```
BYTE* _pbyResult = NULL;
...
_pbyResult = (BYTE*)malloc( CCryptoDES_GetResultLength( iContext ) );
CCryptoDES_GetResult( iContext, _pbyResult );
```

The first line allocates the memory necessary to store the result. The second line gets the result and stores it in the allocated block.

We can now work with the encrypted data *pbyResult*. *To decrypt the data you must follow exactly the same steps as for the encryption, except that you must use the `CryptoDES_Decrypt()` function.*

```
...
CCryptoDES_Decrypt( iContext,
    _pbyResult,
    CCryptoDES_GetResultLength( iContext ) );

free( _pbyResult );
_pbyResult = (BYTE*)malloc(CCryptoDES_GetResultLength( iContext ));
CCryptoDES_GetResult( iContext, _pbyResult );
```



We are now done with the TripleDES encryption object. We must free the associated resources before leaving. This is done by calling the `CryptoDES_ReleaseContext()` method.

```
...
CCryptoDES_ReleaseContext( iContext );
...
```

In one piece

```
#include <stdio.h>
#include "ICryptoDES.h"

...

int iContext = 0;
int iResultLength = 0;

BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
BYTE _pbyExpectedResult[8] = {0x95,0x8E,0x6E,0x62,0x7A,0x05,0x55,0x7B};
BYTE* _pbyResult = NULL;

iContext = CCryptoDES_CreateContext();
CCryptoDES_SetKey( iContext, _pbyKey );
CCryptoDES_Encrypt( iContext, _pbyDataIn, 8 );
_pbyResult = (BYTE*)malloc(CCryptoDES_GetResultLength( iContext ));
if( !_pbyResult )
{
    return 0;
}

CCryptoDES_GetResult( iContext, _pbyResult );
if( memcmp( _pbyResult, _pbyExpectedResult, 8 ) )
{
    free( _pbyResult );
    return 0;
}

CCryptoDES_Decrypt( iContext, _pbyResult, CCryptoDES_GetResultLength(
iContext ) );

free( _pbyResult );
_pbyResult = NULL;

_pbyResult = (BYTE*)malloc(CCryptoDES_GetResultLength( iContext ));
if( !_pbyResult )
{
    return 0;
}
```



```
CCryptoDES_GetResult( iContext, _pbyResult );

if( memcmp( _pbyDataIn, _pbyResult, 8 ) )
{
    free( _pbyResult );
    return 0;
}

free( _pbyResult );
CCryptoDES_ReleaseContext( iContext );

printf( "Done.\n" );

...
```

CryptoTools tutorial for C++

This section will cover C++ programming using the classes published in the CryptoDLL library. This option is only available with Microsoft Visual C++ because of the name-mangling. If you want to use CryptoTools in C++ using another compiler, you may have to use the COM version of the component. This will be covered in detail later in this document.

Before starting

As for the C version of CryptoTools, you will need to link with the CryptoDLL.lib library and include the header file corresponding to the algorithm you want to use. The following header files are available:

```
#include "ICryptoDES.h"
#include "ICryptoBase64.h"
#include "ICryptoTripleDES.h"
#include "ICryptoMD5.h"
```

To run the program, you must copy the CryptoDLL.dll file into the working directory of your application or into a folder that is in the Windows execution PATH. By default CryptoTools.dll is copied in the Windows system folder during installation.

The Code

All CryptoTools objects are accessible through a set of interfaces. These interfaces are defined in the four header files we saw earlier in this document. These header files contain both the C and the C++ definitions and prototypes. ICCryptoDES is the interface for the DES encryption object, and a new object can be created through the static method :Create().



```
ICryptoDES* des = ICryptoDES::Create();
```

Now that we have a reference to our encryption object we can set the encryption key. The encryption key will be required later to decrypt the data.

```
BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};  
int _iRet = des->SetKey( _pbyKey );
```

You are now ready to encrypt your data. We will encrypt the same data as in the previous C example.

```
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};  
int _iRet = des->Encrypt( _pbyDataIn, 8 );
```

If the function returns with 0, the encryption succeeded; otherwise, an error code is returned.

To retrieve the result, we must first allocate a block of memory to store it. The function `GetResultLength()` returns the length of the resulting block of data.

```
BYTE* _pbyResult = new BYTE[des->GetResultLength()];
```

Now we can call `GetResult` to store the result into the allocated block of memory.

```
int _iRet = des->GetResult( _pbyResult );
```

To decrypt the data you must follow essentially the same steps as for encrypting. Once the key is set you must call the `Decrypt()` function.

```
int _iRet = des->Decrypt( _pbyResult, des->GetResultLength() );
```

Once you are done with the `CryptoDES` object you must release it by calling the `Dispose()` function.

```
ICryptoDES::Dispose( des );
```

In one piece

```
#include <stdio.h>  
#include "..\..\ICryptoDES.h"  
  
BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};  
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};  
BYTE _pbyExpectedResult[8] = {0x95,0x8E,0x6E,0x62,0x7A,0x05,0x55,0x7B};
```



```
int _iRet = 0;
ICryptoDES* des = ICryptoDES::Create();

if( !des )
    return false;

_iRet = des->SetKey( _pbyKey );
_iRet = des->Encrypt( _pbyDataIn, 8 );

BYTE* _pbyResult = new BYTE[des->GetResultLength()];
if( !_pbyResult )
{
    printf("Failed to allocate memory for encryption result.\n");
    ICryptoDES::Dispose( des );
    return false;
}
_iRet = des->GetResult( _pbyResult );
if( memcmp( _pbyResult, _pbyExpectedResult, 8 ) )
{
    printf("Encrypted data does not match expected result.\n");
    ICryptoDES::Dispose( des );
    delete[] _pbyResult;
    return false;
}

_iRet = des->Decrypt( _pbyResult, des->GetResultLength() );

delete[] _pbyResult;
_pbyResult = NULL;

_pbyResult = new BYTE[des->GetResultLength()];
if( !_pbyResult )
{
    printf("Failed to allocate memory for decryption result.\n");
    ICryptoDES::Dispose( des );
    return false;
}

_iRet = des->GetResult( _pbyResult );
if( memcmp( _pbyDataIn, _pbyResult, 8 ) )
{
    printf("Decrypted data does not match initial data.\n");
    ICryptoDES::Dispose( des );
    delete[] _pbyResult;
    return false;
}

delete[] _pbyResult;
ICryptoDES::Dispose( des );

printf( "Done.\n" );
```



...

CryptoTools tutorial for COM/C++

This section will cover COM programming using C++. COM objects can be used in almost all languages running on Microsoft Windows operating system. The following sample will show how to use it with C++.

Before starting

The COM version of CryptoTools is distributed through the CryptoCOM.dll file. This library must be registered using regsvr32.exe. The CryptoTools installation should register the library copied into your Windows system folder.

You will also need to include some header files with your project code. These files are located into " [PROGRAM FILES]\CryptoTools\CryptoCOM"

```
#include "CryptoCOM.h"  
#include "CryptoCOM_i.c"
```

The Code

The first step in COM programming is to initialize the COM library by calling CoInitialize(). You will find more details about COM programming and apartments' threading modes in the Microsoft documentation.

```
CoInitialize(NULL);
```

Next, we must create the encryption object by calling CoCreateInstance().

```
ICryptoDES* _pDES = NULL;  
CoCreateInstance( CLSID_CryptoDES,  
NULL,  
CLSCTX_INPROC_SERVER,  
IID_ICryptoDES,  
(void**)&_pDES )
```

We now have a reference to the encryption component through the _pDES interface pointer.

We must now set the encryption key prior to encrypt or decrypt data. The COM version of the component requires a one-dimensional SAFEARRAY for an encryption key. A helper function is provided in this example to create a SAFEARRAY out of an array of bytes.



```
BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
CComVariant _varKey = CreateUnidimensionalByteArray( _pbyKey, 8 );
_pDES->put_Key( _varKey );
```

Now that the key is set, we can proceed to encryption. As for the key, the Encrypt and Decrypt methods requires a SAFEARRAY for the data to be encrypted or decrypted.

```
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
CComVariant _var = CreateUnidimensionalByteArray ( _pbyDataIn, 8 );
_pDES->Encrypt( _var );
```

The data is encrypted. We must now retrieve the resulting block of encrypted data. The data returned by the get_Result() function is a VARIANT containing a one-dimensional SAFEARRAY that holds the encrypted data.

```
CComVariant _varResult;
_pDES->get_Result( &_varResult );
```

The result can be read through `_varResult.parray->pvData`.

You can then decrypt a VARIANT by simply calling the Decrypt() method.

```
_pDES->Decrypt( _varResult );
```

Once you have finish using the encryption object you must release your reference to it by calling the Release() method. This call will decrease the reference count associated to the object. Once the ref-count reaches 0, the object is freed.

```
_pDES->Release();
```

Before exiting your application, do not forget to call the CoUninitialize() function.

```
CoUninitialize();
```

In one piece

```
CoInitialize(NULL);

ICryptoDES* _pDES = NULL;
if( FAILED( CoCreateInstance( CLSID_CryptoDES,
NULL,
CLSCTX_INPROC_SERVER,
IID_ICryptoDES,
(void**)&_pDES ) ) )
```



```
{
return false;
}
BYTE _pbyKey[8] = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
BYTE _pbyDataIn[8] = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
BYTE _pbyExpectedResult[8] = {0x95,0x8E,0x6E,0x62,0x7A,0x05,0x55,0x7B};

CComVariant _varKey = CreateUnidimensionalByteArray ( _pbyKey, 8 );

_pDES->put_Key( _varKey );

CComVariant _var = CreateUnidimensionalByteArray ( _pbyDataIn, 8 );

_pDES->Encrypt( _var );

CComVariant _varResult;
_pDES->get_Result( &_varResult );

if( memcmp( _pbyExpectedResult, _varResult.parray->pvData, 8 ) )
{
    return false;
}

_pDES->Decrypt( _varResult );

CComVariant _varResult2;
_pDES->get_Result( &_varResult2 );

_pDES->Release();

if( memcmp( _var.parray->pvData, _varResult2.parray->pvData, 8 ) ||
    _var.parray->rgsabound->cElements !=
    _varResult2.parray->rgsabound->cElements )
{
    return false;
}

CoUninitialize();

...
```

The following function is a helper function to help you create a SAFEARRAY out of a regular BYTE array.

```
VARIANT CreateUnidimensionalByteArray ( BYTE* pbyDataIn,
    int length )
{
    SAFEARRAYBOUND rgsabound[1];
    rgsabound[0].lLbound = 0;
    rgsabound[0].cElements = length;
```



```
VARIANT _var;  
_var.vt = VT_ARRAY | VT_UI1;  
_var.parray = SafeArrayCreate( VT_UI1, 1, rgsabound );  
  
BYTE* _pbyPtr = NULL;  
SafeArrayAccessData( _var.parray, (void HUGE_PTR*)&_pbyPtr );  
memcpy( _pbyPtr, pbyDataIn, length );  
SafeArrayUnaccessData( _var.parray );  
  
return _var;  
}
```

CryptoTools tutorial for COM/VB

Before starting

To use CryptoTools in your Visual Basic, you must add it to the project references. A component can be added to the project references by selecting the menu item “Project/References...” From there you can add a reference to the “CryptoCOM 1.0 Type Library” library. You are now ready to use CryptoTools.

The code

The first step is to create the encryption component. The following command creates an instance of the encryption component implementing the algorithm of your choice:

```
Dim cryptoDES As New CryptoCOMLib.cryptoDES
```

Next, you must set the encryption key. This is done by constructing an array of bytes and assigning it to the Key attribute of the object.

```
Dim key(8) As Byte  
key(0) = &H30  
key(1) = &H0  
key(2) = &H0  
key(3) = &H0  
key(4) = &H0  
key(5) = &H0  
key(6) = &H0  
key(7) = &H0  
  
cryptoDES.key = key
```



Next, we can encrypt the data. This can be done by calling the Encrypt() method of the encryption object. The encryption's result is accessible through the component's Result attribute.

```
Dim data(8) As Byte
    data(0) = &H10
    data(1) = &H0
    data(2) = &H0
    data(3) = &H0
    data(4) = &H0
    data(5) = &H0
    data(6) = &H0
    data(7) = &H1

cryptoDES.Encrypt (data)

Dim dataOut() As Byte
dataOut = cryptoDES.Result
```

This data must be decrypted in order to be read by the destination application. This can be done by calling the Decrypt() method of the encryption component. Once again, the Result is accessible through the Result object's attribute.

```
cryptoDES.Decrypt (cryptoDES.Result)
dataOut = cryptoDES.Result
```

In one piece

```
Private Function Test01() As Boolean

    Dim key(8) As Byte
    key(0) = &H30
    key(1) = &H0
    key(2) = &H0
    key(3) = &H0
    key(4) = &H0
    key(5) = &H0
    key(6) = &H0
    key(7) = &H0

    Dim data(8) As Byte
    data(0) = &H10
    data(1) = &H0
    data(2) = &H0
    data(3) = &H0
    data(4) = &H0
    data(5) = &H0
    data(6) = &H0
    data(7) = &H1

    Dim expectedResult(8) As Byte
```

CryptoTools 3.0

© 2005 [Deux Sortes Inc.](#)
All rights reserved.



```

expectedResult(0) = &H95
expectedResult(1) = &H8E
expectedResult(2) = &H6E
expectedResult(3) = &H62
expectedResult(4) = &H7A
expectedResult(5) = &H5
expectedResult(6) = &H55
expectedResult(7) = &H7B

Dim dataOut() As Byte
Dim cryptoDES As New CryptoCOMLib.cryptoDES

cryptoDES.key = key
cryptoDES.Encrypt (data)
dataOut = cryptoDES.Result

For i = 0 To 7
    If expectedResult(i) <> dataOut(i) Then
        MsgBox "CryptoComTestProjectVB : Test01 failed! Encryption
does not return the expected result."
        Test01 = False
        Return
    End If
Next

cryptoDES.Decrypt (cryptoDES.Result)
dataOut = cryptoDES.Result

For i = 0 To 7
    If data(i) <> dataOut(i) Then
        MsgBox "CryptoComTestProjectVB : Test01 failed! Decrypted
data does not match original data."
        Test01 = False
        Return
    End If
Next

Test01 = True
End Function

```

CryptoTools tutorial for COM/VBScript & ASP

Before Starting

Scripting is used by many products to extend their capabilities. VBScript is one of the most popular languages with JScript. The only thing to verify before you begin coding is to make sure that the CryptoCOM.dll dynamic library is registered using regsvr32.exe. This operation is normally done by the installation program.

CryptoTools 3.0

© 2005 [Deux Sortes Inc.](#)
All rights reserved.



In this tutorial we will encrypt a text string, given that encrypting text is more useful than encrypting binary blocks of data when working with scripting languages.

The Code

The first step is to create the encryption object. This is done in VBScript by calling the `CreateObject()` function with the class name of the object to create. In this example we create two encryption objects, one for DES and one for Base64.

```
' Create the encryption object
Set crDES = CreateObject ("CryptoCOM.CryptoDES")
Set cr64 = CreateObject ("CryptoCOM.CryptoBase64")
```

Once the objects are created, we can start encrypting text. We first generate an encryption key.

```
' Generate a key from a password
call crDES.DeriveKeyFromPassword( "1234567890", Empty, 1 )
```

Then we encrypt the text using the DES algorithm. To encrypt text you must call the `EncryptText()` method.

```
' Encrypt a text string
crDES.EncryptText( "String to encrypt ..." )
```

To make the result easier to display (or to store it in a database) we then encode the result into base64 and display it through a message box.

```
' Retrieve the result of the encryption
resultDES = crDES.Result

' Encode the result in base64 and display it
b64Ret = cr64.Encrypt( resultDES )

MsgBox b64Ret
```

Finally, we can now retrieve the original text by first decoding the base64 text string and then decrypting the resulting data.

```
' Decode the encoded result
cr64.Decrypt( b64Ret )

' Retrieve the result
```



```
result64 = cr64.Result

' Decrypt and display the original text string
ret = crDES.DecryptText( result64 )

MsgBox ret
```

In one piece

CryptoCOM.wsf

```
<?xml version="1.0" encoding="utf-8" ?>
<package xmlns="Windows Script Host">
  <job>
    <script language="vbscript">
      <![CDATA[

        ' Create the encryption object
        Set crDES = CreateObject ("CryptoCOM.CryptoDES")
        Set cr64 = CreateObject ("CryptoCOM.CryptoBase64")

        ' Generate a key from a password
        call crDES.DeriveKeyFromPassword( "1234567890", Empty, 1 )

        ' Encrypt a text string
        crDES.EncryptText( "String to encrypt ..." )

        ' Retrieve the result of the encryption
        resultDES = crDES.Result

        ' Encode the result in base64 and display it
        b64Ret = cr64.Encrypt( resultDES )
        MsgBox b64Ret

        ' Decode the encoded result
        cr64.Decrypt( b64Ret )

        ' Retrieve the result
        result64 = cr64.Result

        ' Decrypt and display the original text string
        ret = crDES.DecryptText( result64 )
        MsgBox ret

      ]]>
    </script>
  </job>
</package>
```

CryptoCOM.asp

```
<%@Language=VBScript%>
```

CryptoTools 3.0

© 2005 [Deux Sortes Inc.](#)
All rights reserved.



```
<html>
<head>
<title>CryptoCOM.asp</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" link="#000000" vlink="#000000" alink="#666666"
text="#000000">
Encrypted:<br>
<textarea cols="50" rows="10" ID="Textareal" NAME="Textareal">
<%
    ' Create the encryption object
    Set crDES = CreateObject ("CryptoCOM.CryptoDES")
    Set cr64 = CreateObject ("CryptoCOM.CryptoBase64")

    ' Generate a key from a password
    call crDES.DeriveKeyFromPassword( "1234567890", Empty, 1 )

    ' Encrypt a text string
    crDES.EncryptText( "String to encrypt ..." )

    ' Retrieve the result of the encryption
    resultDES = crDES.Result

    ' Encode the result in base64 and display it
    b64Ret = cr64.Encrypt( resultDES )

    Response.Write b64Ret
%>
</textarea><br>
Decrypted:<br>
<textarea cols="50" rows="10" ID="Textarea2" NAME="Textareal">
<%
    ' Decode the encoded result
    cr64.Decrypt( b64Ret )

    ' Retrieve the result
    result64 = cr64.Result

    ' Decrypt and display the original text string
    ret = crDES.DecryptText( result64 )

    Response.Write ret
%>
</textarea>
</body>
</html>
```



CryptoTools tutorial for COM/JScript

Before Starting

Since the code in JScript is very similar to the code found in the VBScript tutorial, we will only provide a sample HTML code using CryptoTools with JScript.

In one piece

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>CryptoCOM.html</TITLE>
<SCRIPT>
function encryptText()
{
    var crDES;
    crDES = new ActiveXObject("CryptoCOM.CryptoDES");

    var cr64;
    cr64 = new ActiveXObject("CryptoCOM.CryptoBase64");

    crDES.DeriveKeyFromPassword( "1234567890", null, 1 )

    crDES.EncryptText( "String to encrypt ..." )

    resultDES = crDES.Result

    b64Ret = cr64.Encrypt( resultDES )
    alert( b64Ret )

    cr64.Decrypt( b64Ret )

    result64 = cr64.Result

    ret = crDES.DecryptText( result64 )
    alert( ret )
}
</SCRIPT>
</HEAD>

<BODY>
<input type="submit" name="" id="" value="OK" onclick="encryptText();">
</BODY>
</HTML>
```



CryptoTools tutorial for .Net/C#

Before starting

Before starting to code, you must add the CryptoNET.dll assembly to your project references. To simplify the code, add a reference to the CryptoTools namespace.

```
using com.CryptoTools;
```

The code

As with the other languages, the first step is to create an instance of the encryption object we want to use. In this case, we will instantiate the DES algorithm component.

```
CryptoDES des = new CryptoDES();
```

Then we set the encryption key to be used for encryption.

```
byte[] key = { 0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00 };
des.Key = key;
```

Finally, we encrypt the data.

```
byte[] dataIn = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
des.Encrypt( dataIn );
```

We are done. If you want to decrypt the result, you only have to call the Decrypt() method like this:

```
des.Decrypt( des.Result );
```

In one piece

```
using com.CryptoTools;

...

byte[] key = { 0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00 };
byte[] dataIn = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};

CryptoDES des = new CryptoDES();

// Key setup
des.Key = key;
```



```
// Encrypt data
des.Encrypt( dataIn );

// Decrypt data
des.Decrypt( des.Result );

...
```

CryptoTools tutorial for .Net/VB.Net

Before starting

Using CryptoTools with VB.Net is pretty much the same thing as with C#. First of all, add a reference to the CryptoNET.dll assembly to your project.

It is recommended that you add a reference to the CryptoTools namespace in order to simplify code-writing.

```
Imports com.CryptoTools
```

The code

As for the C# tutorial, using CryptoTools in VB.Net is really simple. First of all, you create the encryption object implementing the algorithm you want to use.

```
Dim cryptoDES As New CryptoDES
```

Once you have the object created, you set the encryption key to use for either encryption or decryption.

```
Dim key() As Byte = {&H30, &H0, &H0, &H0, &H0, &H0, &H0, &H0}
cryptoDES.Key = key
```

You are then ready to encrypt the data. This can be done by calling the Encrypt() method of the encryption object.

```
cryptoDES.Encrypt(data)
```

Here it is. Again, like in the previous examples, you can check that the data can be decrypted using the same key by calling the Decrypt() method.

```
cryptoDES.Decrypt(cryptoDES.Result)
```



In one piece

```
Imports com.CryptoTools

...

Dim key() As Byte = {&H30, &H0, &H0, &H0, &H0, &H0, &H0, &H0}
Dim data() As Byte = {&H10, &H0, &H0, &H0, &H0, &H0, &H0, &H1}
Dim cryptoDES As New CryptoDES

cryptoDES.Key = key
cryptoDES.Encrypt(data)
cryptoDES.Decrypt(cryptoDES.Result)

...
```

CryptoTools tutorial for Java

Before starting

The Java version of CryptoTools allows you to exchange information with virtually any platform supporting Java.

To compile and run your application with CryptoTools for Java, you need only add the CryptoTools.jar library into the classpath of your application and add the following import instruction to your .java code file:

```
import com.CryptoTools.*;
```

The Code

In Java, like with all the other languages, the first step is to create an object implementing the encryption algorithm you want to use. In this case we instantiate a CryptoDES object.

```
CryptoDES des = new CryptoDES();
```

Once the object is instantiated, we can set the encryption key to be used for both encryption and decryption.

```
byte[] key = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
des.setKey( key );
```

We are now ready to encrypt the data. To do so, we need only call the Encrypt() method of the CryptoDES object.

```
byte[] binData = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};
```

CryptoTools 3.0

© 2005 [Deux Sortes Inc.](#)
All rights reserved.



```
des.encrypt( binData );
```

This is it. The data is encrypted. You can, later in the process, decrypt the data by calling the Decrypt() method.

```
des.decrypt(des.getResult() );
```

The final result for encryption and decryption can be accessed through the "Result" object attribute.

```
byte[] binResult = des.getResult();
```

In one piece

```
import com.CryptoTools.*;
...
CryptoDES des = new CryptoDES();

byte[] key = {0x30,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
byte[] binData = {0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x01};

des.setKey( key );
des.encrypt( binData );
des.decrypt(des.getResult() );
byte[] binResult = des.getResult();
...
```