HowTo – Create a Hash-Plugin for Cryptool 2.0 using Visual Studio 2008

This document is dedicated for the developer of a plugin for the e-learning program Cryptool 2.0. It describes step-by-step how you can create a new plugin in VS2008.

How this can be done is described in this document, building a sample plugin which delivers the functionality for a “new” **hash algorithm**.

The new plugin contains the according hash code and offers its functionality to the Cryptool application using the interface “IHashAlgorithm” from the CrypPluginBase.
Additionally it delivers the according controls (like buttons, text boxes) and information to the TaskPane (icon, caption, descriptions).

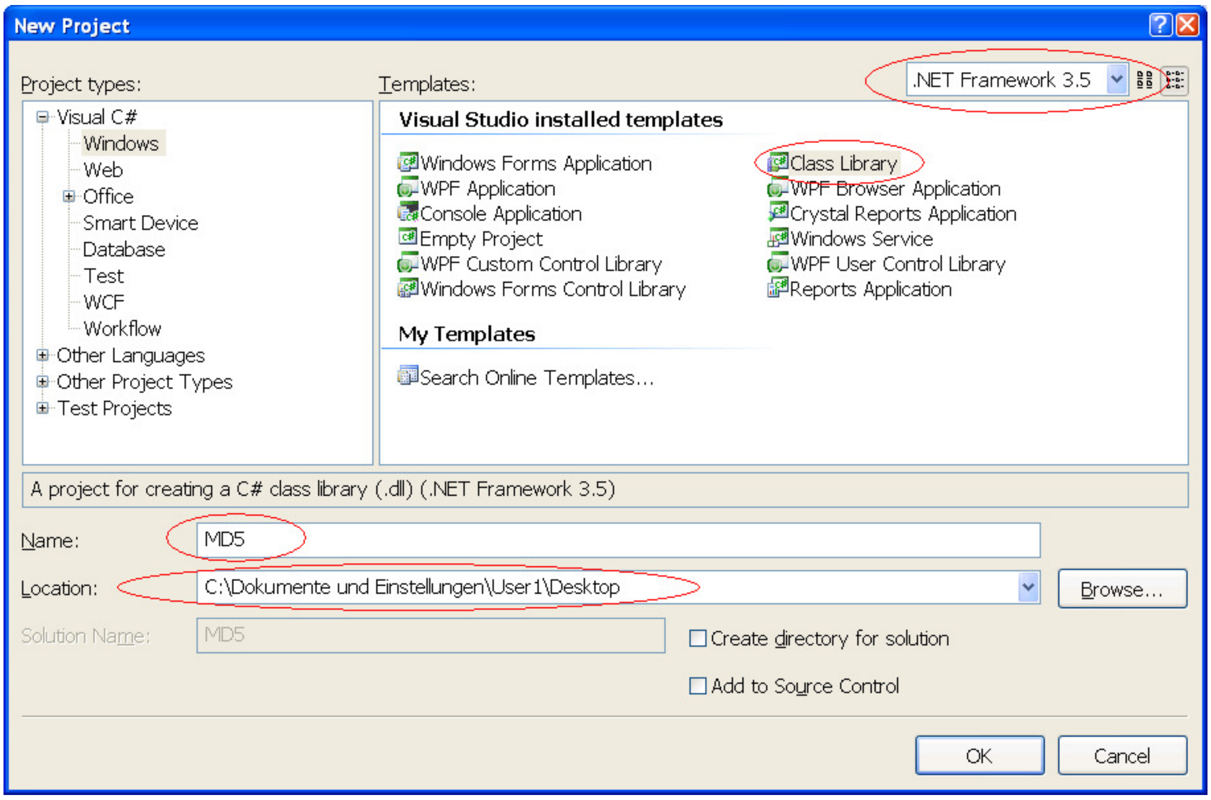
Please be aware that the Cryptool 2.0 project is under development. Therefore, the screenshots in this document might represent the latest state of the project. However, we aim to provide a documentation and screenshots that can always guide you through the first steps on the development of a Plugin. If you anyhow detect a screenshot or description that does not lead you to success, we appreciate your feedback. Please visit the Cryptool project website (www.cryptool.org) to get in touch with us.

Author:	Sebastian Przybylski
Date:	2008-04-21
Version:	0.9
Reviewed by:	Bernhard Esslinger, …

Content

1. Create a new project in VS2008 for your plugin

Open Visual Studio 2008 and create a new project:


New Project

Project types:
- Visual C#
  - Windows
  - Web
  - Office
  - Smart Device
  - Database
  - Test
  - WCF
  - Workflow
- Other Languages
- Other Project Types
- Test Projects

Templates: .NET Framework 3.5

Visual Studio installed templates

Windows Forms Application	Class Library
WPF Application	WPF Browser Application
Console Application	Crystal Reports Application
Empty Project	Windows Service
WPF Custom Control Library	WPF User Control Library
Windows Forms Control Library	Reports Application

My Templates

Search Online Templates...

A project for creating a C# class library (.dll) (.NET Framework 3.5)

Name: MD5

Location: C:\Dokumente und Einstellungen\User1\Desktop

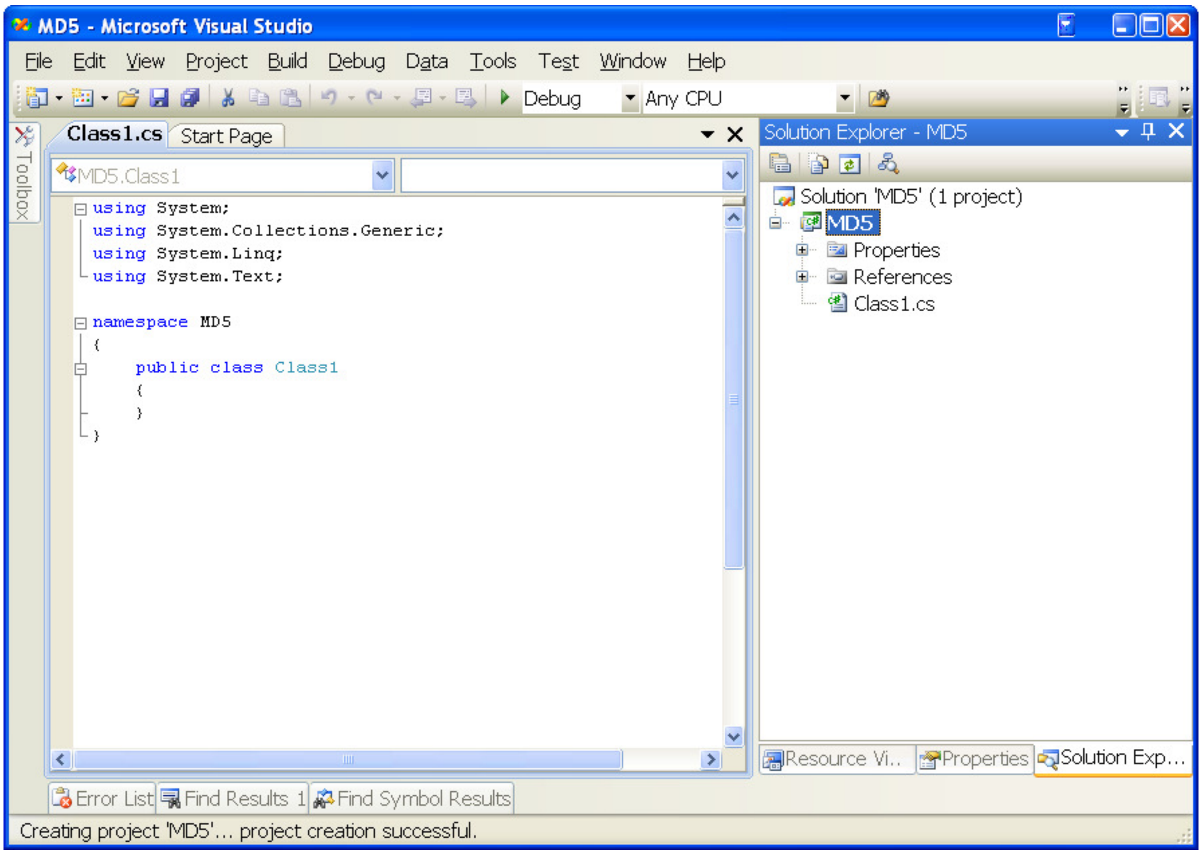
Solution Name: MD5

☐ Create directory for solution

☐ Add to Source Control

Select ".NET-Framework 3.5" as the target framework (the Visual Studio Express edition does not provide this selection because it automatically chooses the actual target framework), and "Class Library" as default template to create a DLL file. Give the project a unique and significant name (here: "MD5"), and choose a location where to save (the Express edition will ask later for a save location when you close your project or your environment). Finally confirm by pressing the "OK" button.

Now your Visual Studio solution should look like this:

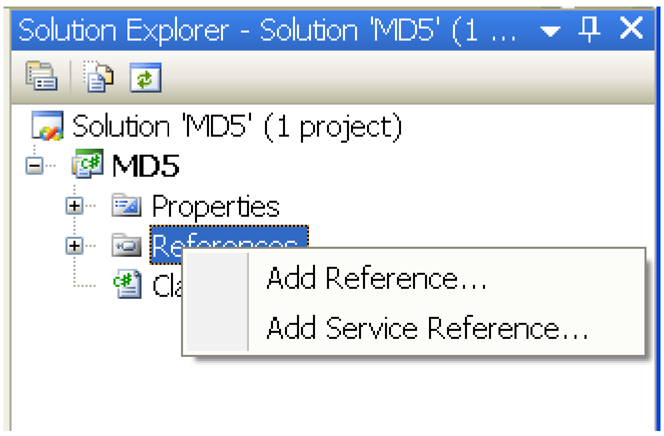


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MD5
{
    public class Class1
    {
    }
}
```

Creating project 'MD5'... project creation successful.

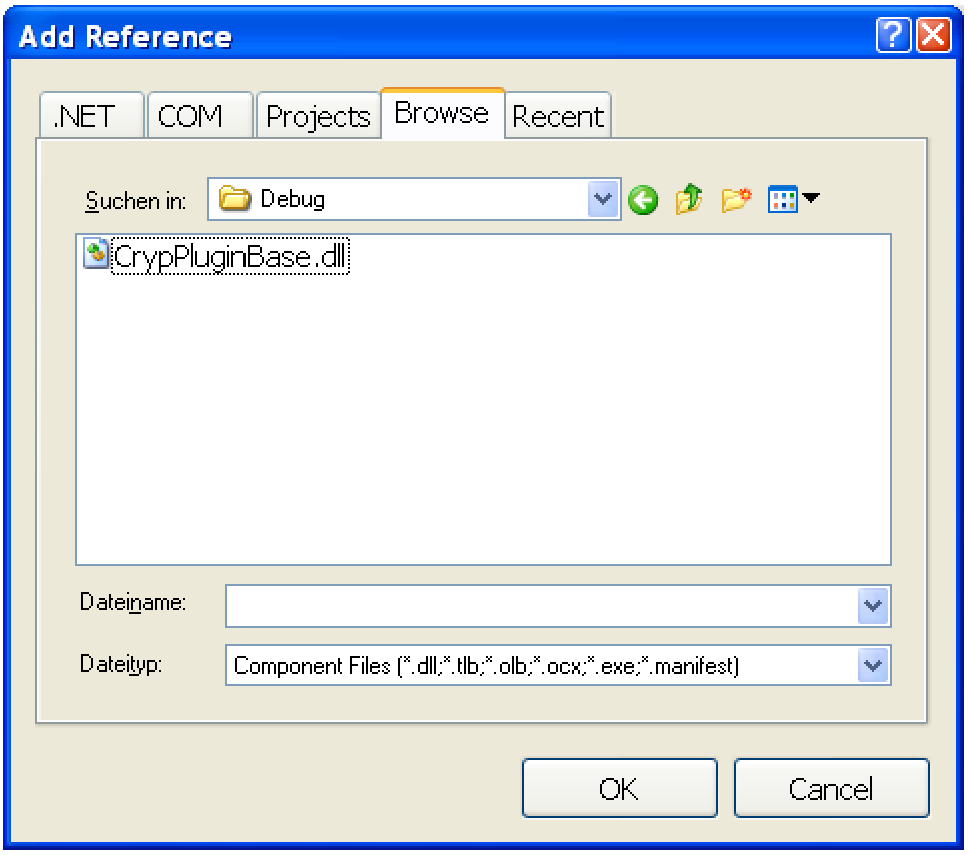
2. Select the interface, your plugin wants to serve

First we have to add a reference to the Cryptool library called “CrypPluginBase.dll” where all necessary Cryptool plugin interfaces are declared.

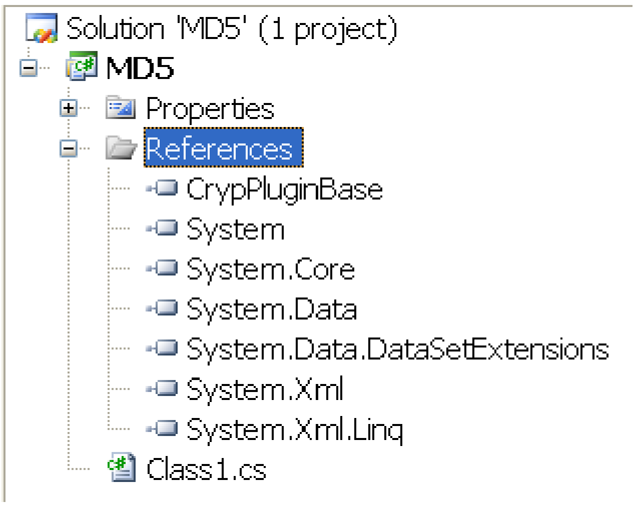

Solution Explorer - Solution 'MD5' (1 ...
Solution 'MD5' (1 project)
MD5
Properties
References
Cla
Add Reference...
Add Service Reference...

Right click on the “Reference” item in the Solution Explorer and choose “Add Reference…”. Now browse to the path where the library file is located (e.g. “c:\Documents and Settings\<Username>\My Documents\Visual Studio 2008\Projects\CrypPluginBase\bin\Debug”) and select the library by double clicking the file or pressing the “OK” button. You can also find the CrypPluginBase.dll on our Subversion repository.

The current version is always located in the following path: "https://www.vs.uni-due.de/svn/CrypTool2/trunk/CrypPluginBase/bin/Debug/"



Afterwards your reference tree view should look like this:



If your plugin will be based on further libraries, you have to add them in the same way.

3. Create the classes for the algorithm and for its settings

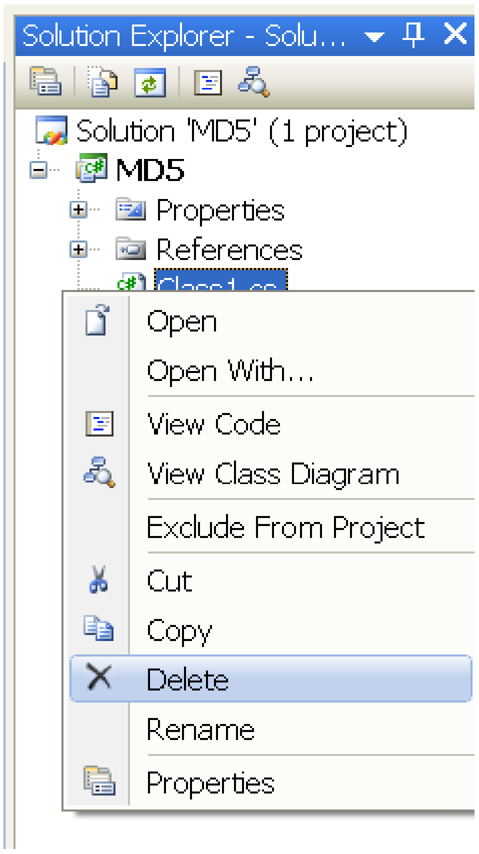
In the next step we have to create two classes. The first class named "MD5" has to inherit from IHashAlgorithm, and the second class named "MD5Settings"from IHashAlgorithmSettings.

3.1 *Create the class for the algorithm (MD5)*

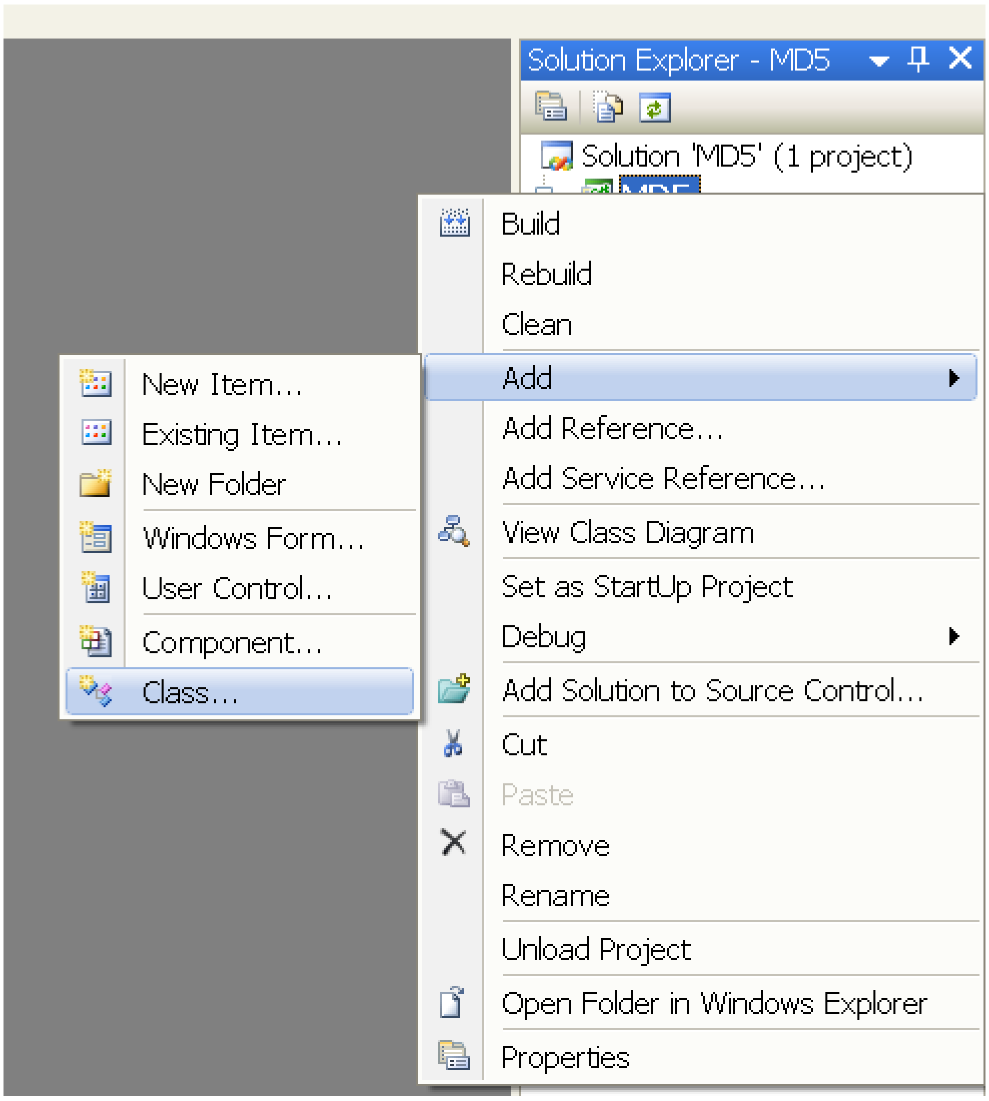
Visual Studio automatically creates a class which has the name "Class1.cs". There are two ways to change the name to "MD5.cs":

- Rename the existent class
- Delete the existent class and create a new one.

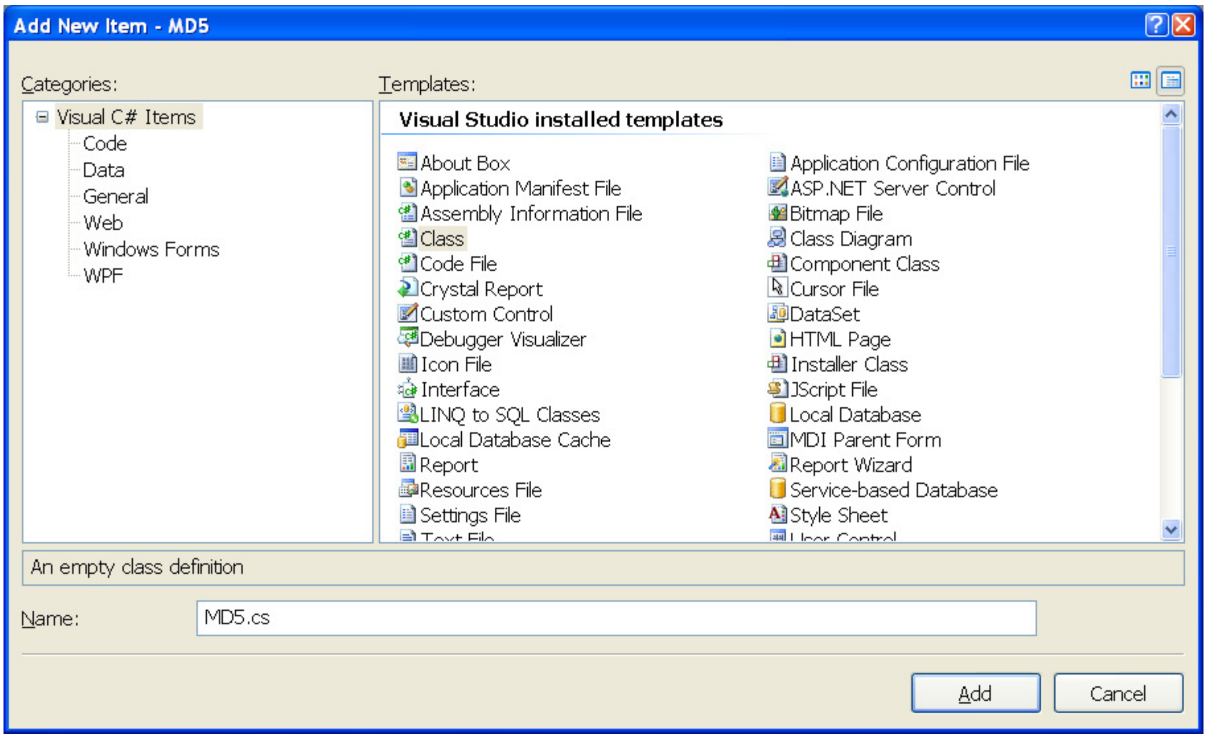
We choose the second way as you can see in the next screenshot:



Now right click on the project item “MD5” and select “Add->Class…”.



Give your class a unique name. We call the class as mentioned above "MD5.cs" and make it public to be available to other classes.

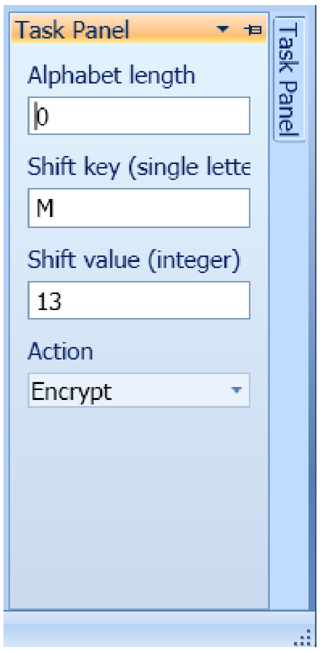


An empty class definition

Name: MD5.cs

3.2 *Create the class for the settings (MD5Settings)*

Act in the same way to add a second class for IHashAlgorithmSettings. We call the class "MD5Settings". The settings class provides the necessary information about controls, captions and descriptions and default parameters for e.g. key settings, alphabets, key length and action to build the **TaskPane** in Cryptool. How a **TaskPane** could look like you can see below for the example of a Caesar encryption.



Alphabet length: 0

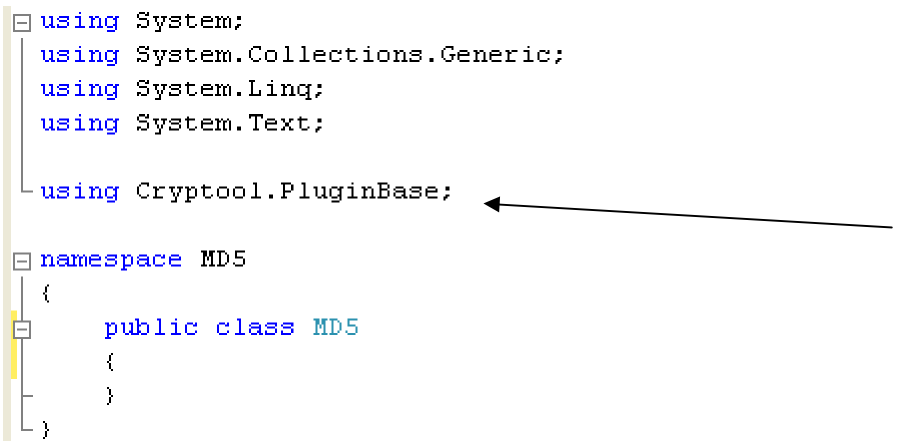
Shift key (single lette: M

Shift value (integer): 13

Action: Encrypt

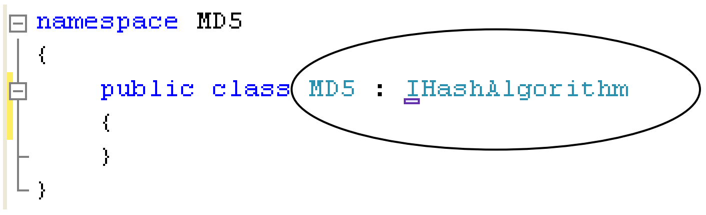
3.3 Add namespace for the class MD5 and the place from where to inherit

Now open the “MD5.cs” file by double clicking on it at the Solution Explorer and include the CrypPluginBase namespace to the class header by typing in the according “using” statement:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Cryptool.PluginBase;

namespace MD5
{
    public class MD5
    {
    }
}
```

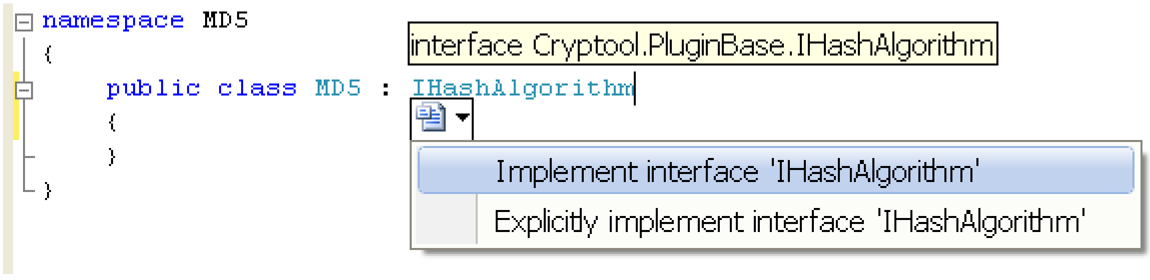
Next let your class “MD5” inherit from IHashAlgorithm by inserting of the following statement:



```
namespace MD5
{
    public class MD5 : IHashAlgorithm
    {
    }
}
```

3.4 Add the interface functions for the class MD5

There is an underscore at the IHashAlgorithm statement. Move your mouse over it or place the cursor at on and press “Shift+Alt+F10” and you will see the following submenu:



```
namespace MD5
{
    public class MD5 : IHashAlgorithm
    {
    }
}
```

interface Cryptool.PluginBase.IHashAlgorithm
Implement interface 'IHashAlgorithm'
Explicitly implement interface 'IHashAlgorithm'

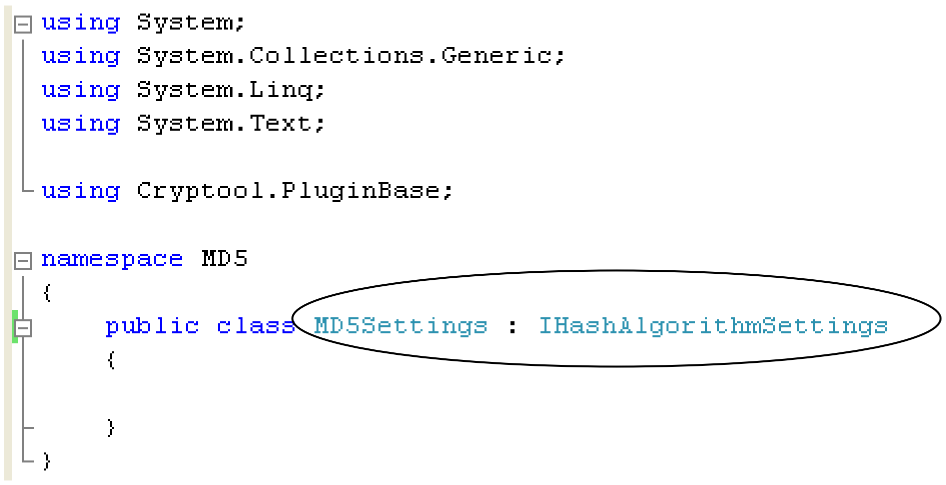
Choose the item “Implement interface ‘IHashAlgorithm’”. Visual Studio will now place all available and needed interface members to interact with the Cryptool core (this saves you also a lot of code typing).

Your code will now look like this:

```
namespace MD5
{
    public class MD5 : IHashAlgorithm
    {
        #region IHashAlgorithm Members

        public void Add(IHashAlgorithmVisualization visualization)
        {
            throw new NotImplementedException();
        }

        public void Hash()
        {
            throw new NotImplementedException();
        }

        public IHashAlgorithmSettings Settings
        {
            get { throw new NotImplementedException(); }
            set { throw new NotImplementedException(); }
        }

        #endregion

        #region IPlugin Members

        public void Dispose()
        {
            throw new NotImplementedException();
        }

        public void Initialize()
        {
            throw new NotImplementedException();
        }

        public event StatusBarProgressbarValueChangedHandler OnStatusBarProgressbarValueChanged;

        public event StatusBarTextChangedHandler OnStatusBarTextChanged;

        public System.Windows.Controls.UserControl PresentationControl
        {
            get { throw new NotImplementedException(); }
        }

        #endregion
    }
}
```

3.5 Add namespace and interfaces for the class MD5Settings

Let’s now take a look at the second class “MD5Settings” by double clicking on the “MD5Settings.cs” file in the Solution Explorer. First we also have to include the namespace of “Cryptool.PluginBase” to the class header and let the settings class inherit from “IHashAlgorithmSettings” analogous as seen before at the MD5 class. Visual Studio will here also automatically place code from the Cryptool interface if available.



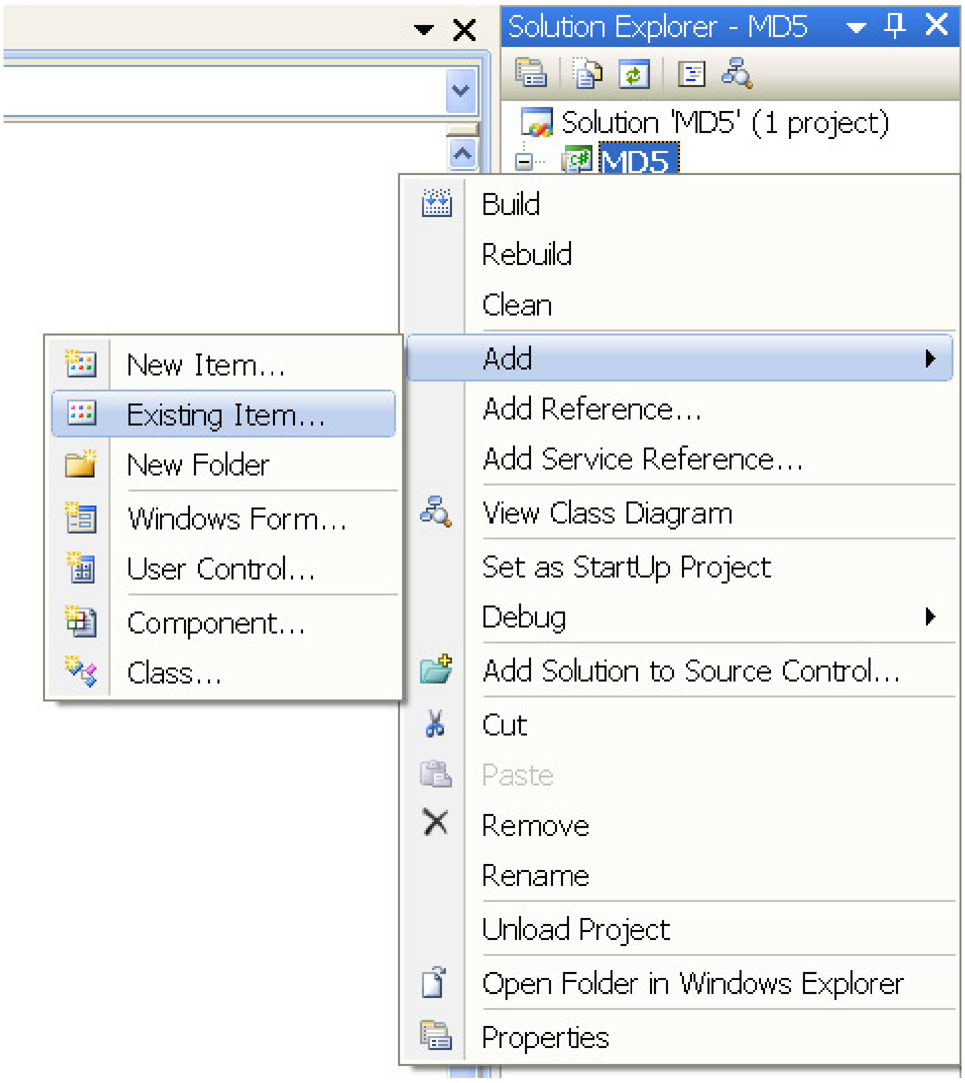
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Cryptool.PluginBase;

namespace MD5
{
    public class MD5Settings : IHashAlgorithmSettings
    {

    }
}
```

3.6 Add controls for the class MD5Settings (if needed)

Now we have to implement some kind of controls (like button, text box) if we need them in the Cryptool **TaskPane** to modify settings of the algorithm. Our MD5 hash algorithm doesn’t have any kind of settings, so we will let this class empty. In this case the **TaskPane** will be empty in Cryptool.

4. Select and add an image as icon for the class MD5

Before we go back to the code of the MD5 class, we have to add an icon image to our project, which will be shown in the Cryptool **ribbon bar**. Note that an icon must have a size of at least 32 x 32 pixels. Smaller icons will be accepted too, but looks unsightly in the ribbon bar. Therefore, right click on the project item “MD5” within the Solution Explorer, and select “Add->Existing Item…”:



Then select “Image Files” as file type, and choose the icon for your plugin:

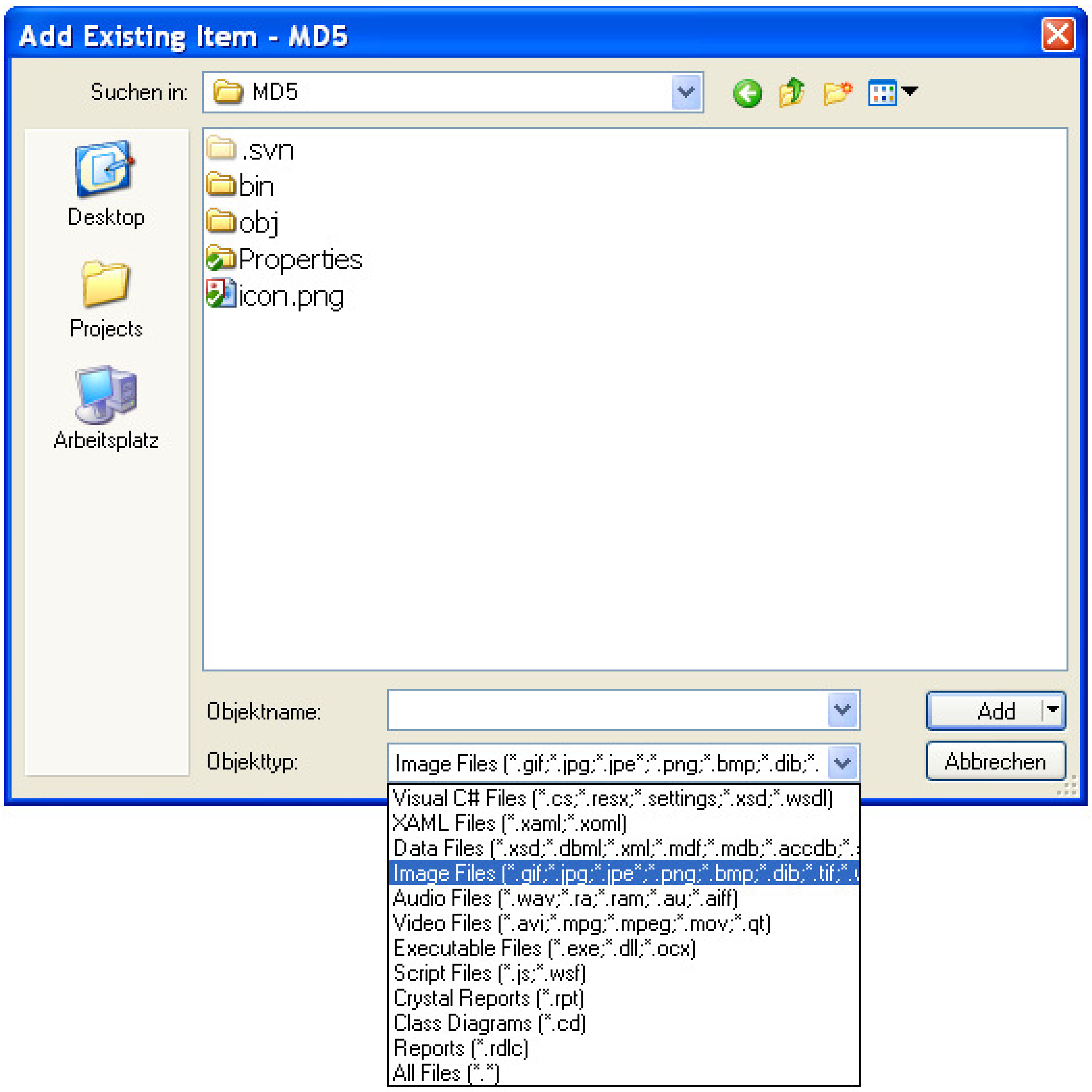
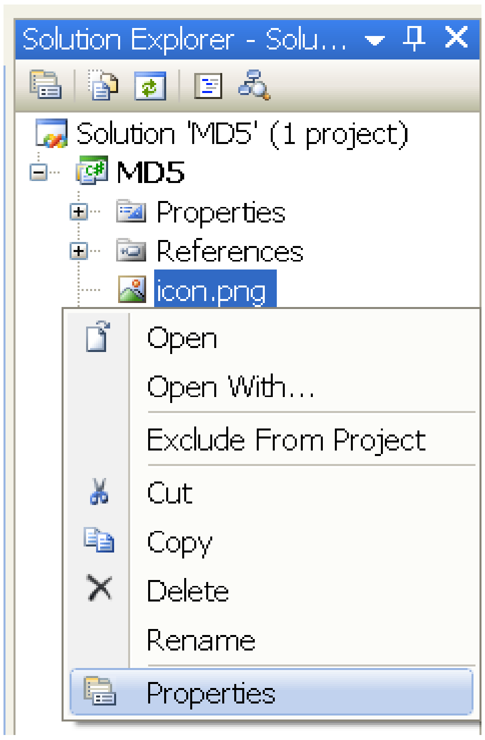


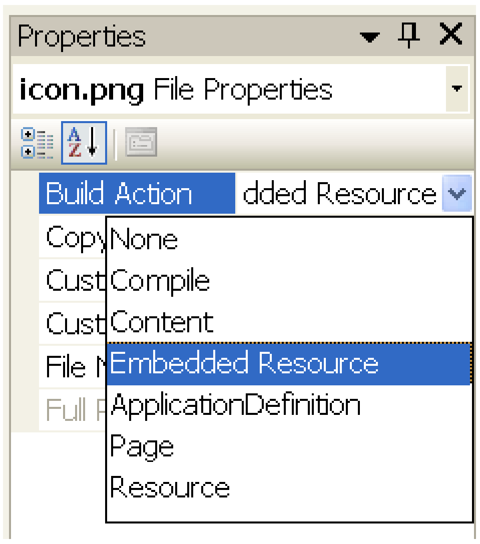
Image Files (*.gif;*.jpg;*.jpe*;*.png;*.bmp;*.dib;*.

Visual C# Files (*.cs;*.resx;*.settings;*.xsd;*.wsdl)
XAML Files (*.xaml;*.xoml)
Data Files (*.xsd;*.dbml;*.xml;*.mdf;*.mdb;*.accdb;*.
Image Files (*.gif;*.jpg;*.jpe*;*.png;*.bmp;*.dib;*.tif;*.
Audio Files (*.wav;*.ra;*.ram;*.au;*.aiff)
Video Files (*.avi;*.mpg;*.mpeg;*.mov;*.qt)
Executable Files (*.exe;*.dll;*.ocx)
Script Files (*.js;*.wsf)
Crystal Reports (*.rpt)
Class Diagrams (*.cd)
Reports (*.rdlc)
All Files (*.*)

Finally we have to set the icon as an “Embedded Resource” to avoid providing the icon as a separate file. Make a right click on the icon and select the item “Properties”:



In the “Properties” panel you have to set the “Build Action” to “Embedded Resource”:

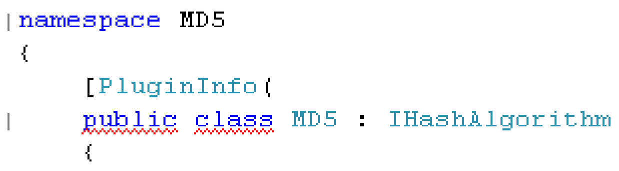


5. Set the attributes for the class MD5

Now let's go back to the code of the MD5 class ("MD5.cs" file). First we have to set the five elements of the attribute's list named "PluginInfo" of our class. These attributes are used by the Cryptool PluginBase and CrypCore to provide information about this plugin:

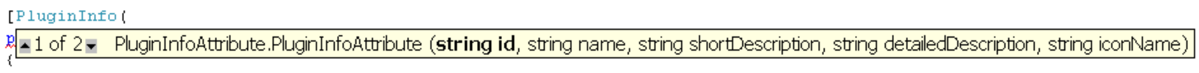
- its unique Guid (to identify the plugin) [Guid = Globally Unique Identifier]
- its name (e.g. to provide the button content)
- its short description (e.g. to provide the button tool tip)
- its long description (e.g. to provide long text area) and
- its icon path (needed to provide e.g. the button image).

Attributes are meta data and a new kind of **declarative** information to define for example classes and class fields. You can also define custom components using attributes. At runtime you are able to get necessary information from your plugins with the aid of attributes.



```
namespace MD5
{
    [PluginInfo(
    public class MD5 : IHashAlgorithm
    {
```

The „PluginInfo" attribute takes five parameters (all four are of type string):

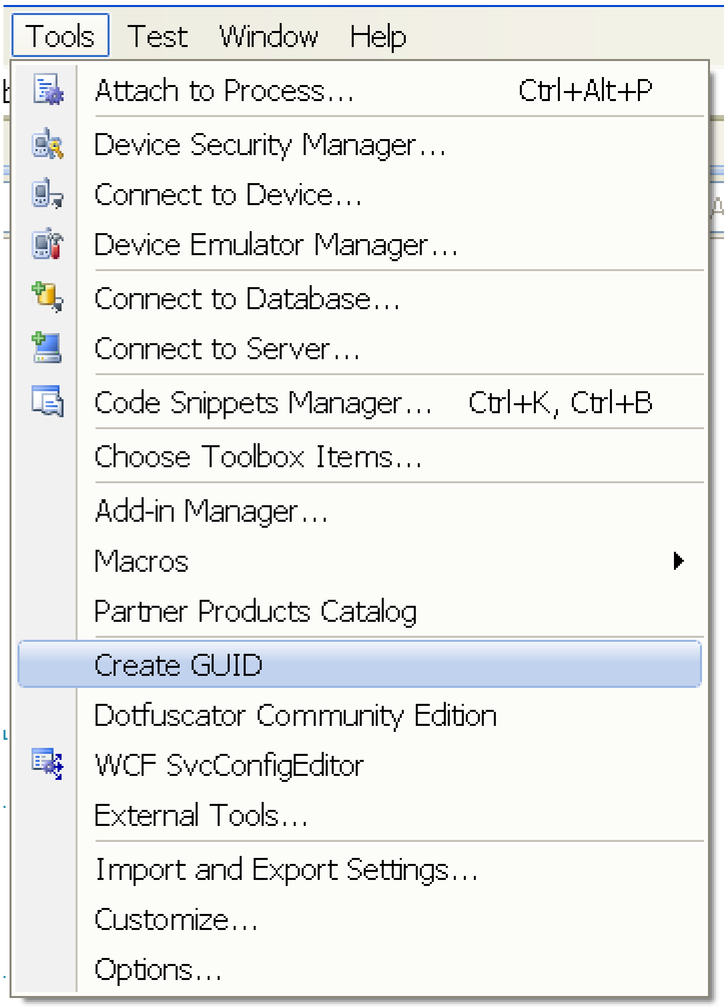


```
[PluginInfo(
p ▲1 of 2▼  PluginInfoAttribute.PluginInfoAttribute (string id, string name, string shortDescription, string detailedDescription, string iconName)
{
```

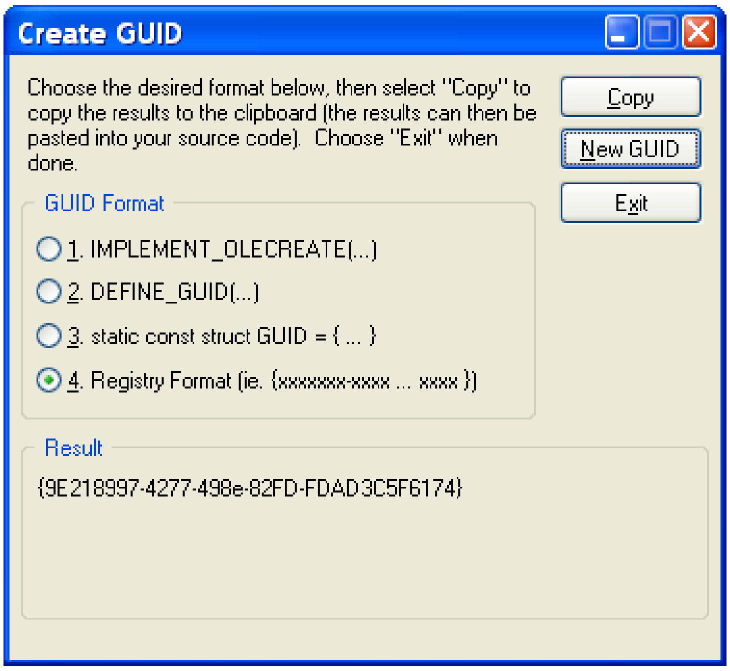
This first parameter takes a Guid to make a unique id of this plugin. Visual Studio gives us a tool to generate new Guids. The Visual C#-Express Edition doesn't provide this tool by default. You have to download it apart at:

http://www.microsoft.com/downloads/details.aspx?familyid=94551f58-484f-4a8c-bb39-adb270833afc&displaylang=en

In Visual Studio 2008 you will find this Guid generator at the menu: “Tools->Create GUID”.



Now you have to click on the “New GUID” button in the GUID creator dialog to generate a new id which is displayed within the “Result” group box. Now copy the generated Guid to your clipboard by pressing the “Copy” button and leave this window by pressing the “Exit” button.

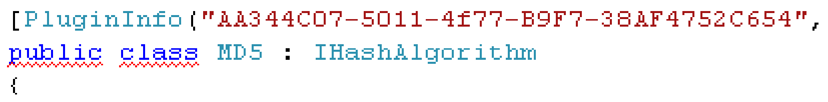


Choose the desired format below, then select "Copy" to copy the results to the clipboard (the results can then be pasted into your source code). Choose "Exit" when done.

- 1. IMPLEMENT_OLECREATE(...)
- 2. DEFINE_GUID(...)
- 3. static const struct GUID = { ... }
- [✓ 4. Registry Format (ie. {xxxxxxxx-xxxx ... xxxx })]

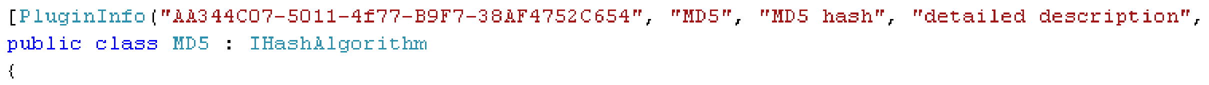
{9E218997-4277-498e-82FD-FDAD3C5F6174}

Set your cursor again at the first parameter of your class attribute “[PluginInfo…” and paste your generated Guid from the clipboard. Then delete the curly brackets surrounding the pasted Guid and instead, surround the Guid with quotation marks (to make a type string) as you can see at the following screenshot:



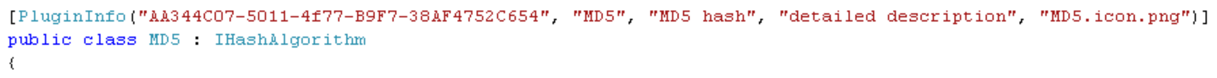
```
[PluginInfo("AA344C07-5011-4f77-B9F7-38AF4752C654",
public class MD5 : IHashAlgorithm
{
```

The next three parameters are needed to define the plugin’s name, its short description and its detailed description:



```
[PluginInfo("AA344C07-5011-4f77-B9F7-38AF4752C654", "MD5", "MD5 hash", "detailed description",
public class MD5 : IHashAlgorithm
{
```

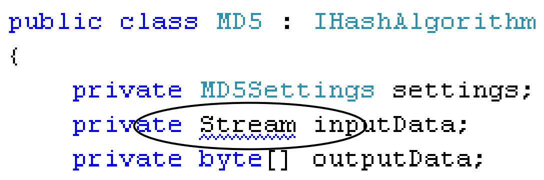
The last parameter tells Cryptool the name of the plugin’s icon. This parameter is made up by <namspace name>.<file name>, where <file name> is the file you chose in chapter 4.



```
[PluginInfo("AA344C07-5011-4f77-B9F7-38AF4752C654", "MD5", "MD5 hash", "detailed description", "MD5.icon.png")]
public class MD5 : IHashAlgorithm
{
```

6. Set the private variables for the settings in the class MD5

The next step is to define some private variables needed e.g. for the settings, input data, output data, input key and initialization vector which could look like this:



```
public class MD5 : IHashAlgorithm
{
    private MD5Settings settings;
    private Stream inputData;
    private byte[] outputData;
```

Please notice the sinuous line at the type “Stream” of the variable inputData. This means you have to include the appropriate namespace. In this case it means you have to include the namespace “System.IO”. How to implement the automatically the necessary namespace you can see at chapter 3.4.

Finally we will need to include the following namespaces:

System.IO – to handle e.g. with streams

System.Security.Cryptography – if we want to use .net algorithms

Cryptool.PluginBase – the master namespace provided by Cryptool

System.ComponentModel – needed for the PropertyChangedEventHandler

Also notice, that we use streams for input and output to perform a file handling in Cryptool with huge files.

7. Define the code of the class MD5 to fit the interface

Next we have to complete our code to correctly serve the interface.

First we add a constructor to our class where we can create an instance of our settings class:

```
public class MD5 : IHashAlgorithm
{
    private MD5Settings settings;
    private Stream inputData;
    private byte[] outputData;

    public MD5()
    {
        this.settings = new MD5Settings();
    }
}
```

Secondly, we have to complete the property function “Settings” provided by the interface:

```
public IHashAlgorithmSettings Settings
{
    get
    {
        return this.settings;
    }
    set
    {
        this.settings = (MD5Settings)value;
    }
}
```

Thirdly we have to define input and output properties with their according attributes. This step is necessary to bind our input and output data with the Cryptool editor plugins input and output controls.

```
[Input("Clear Text Input","Description for Input Text")]
public Stream InputData
{
    get { return this.inputData; }
    set { this.inputData = value; }
}

[Output("Clear Text Output","Description for Output Text")]
public byte[] OutputData
{
    get { return this.outputData; }
    set { this.outputData = value; }
}
```

The input and output attributes provide two parameters for the caption and the descriptiion of the accordingly input and output value. The properties will be shown in the editor workspace by choosing this plugin.

The last step to make our code compileable is to add three assembly references to provide the necessary “Windows” namespace for our **user control** function called “PresentationControl”. As explained above how to add new references to our project, you have to add the following .NET components:

- PresentationCore
- PresentationFramework
- WindowsBase

The function “PresentationControl” is served for the PluginBase if a plugin developer wants to provide his own graphic user interface for Cryptool.

8. Complete the actual code for the class MD5

Up to now, the plugin is ready for the Cryptool base application to be accepted and been shown correctly in the Cryptool menu. What we need now, is the implementation of the actual algorithm function “Hash()” which is up to you as the plugin developer.

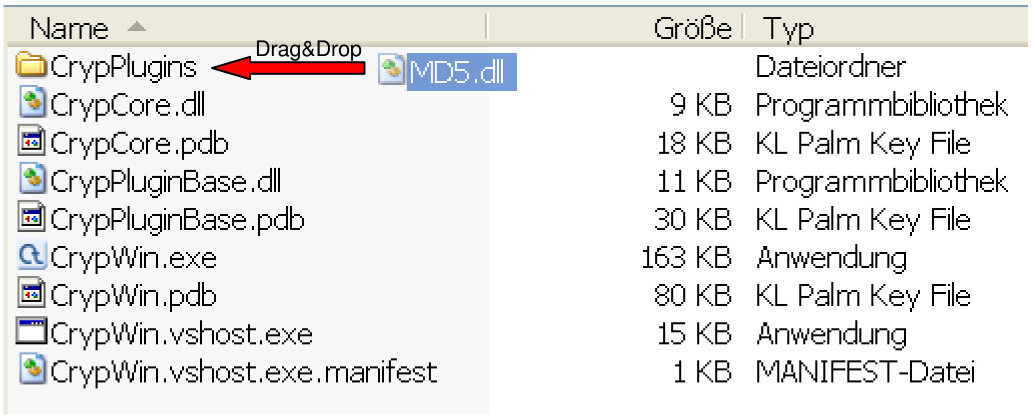
Let us demonstrate the Hash() function, too. Our algorithm is based on the .NET framework:

```
public void Hash()
{
    System.Security.Cryptography.MD5 md5Hash = System.Security.Cryptography.MD5.Create();
    byte[] data = md5Hash.ComputeHash(inputData);

    this.outputData = data;
}
```

9. Import the plugin to Cryptool and test it

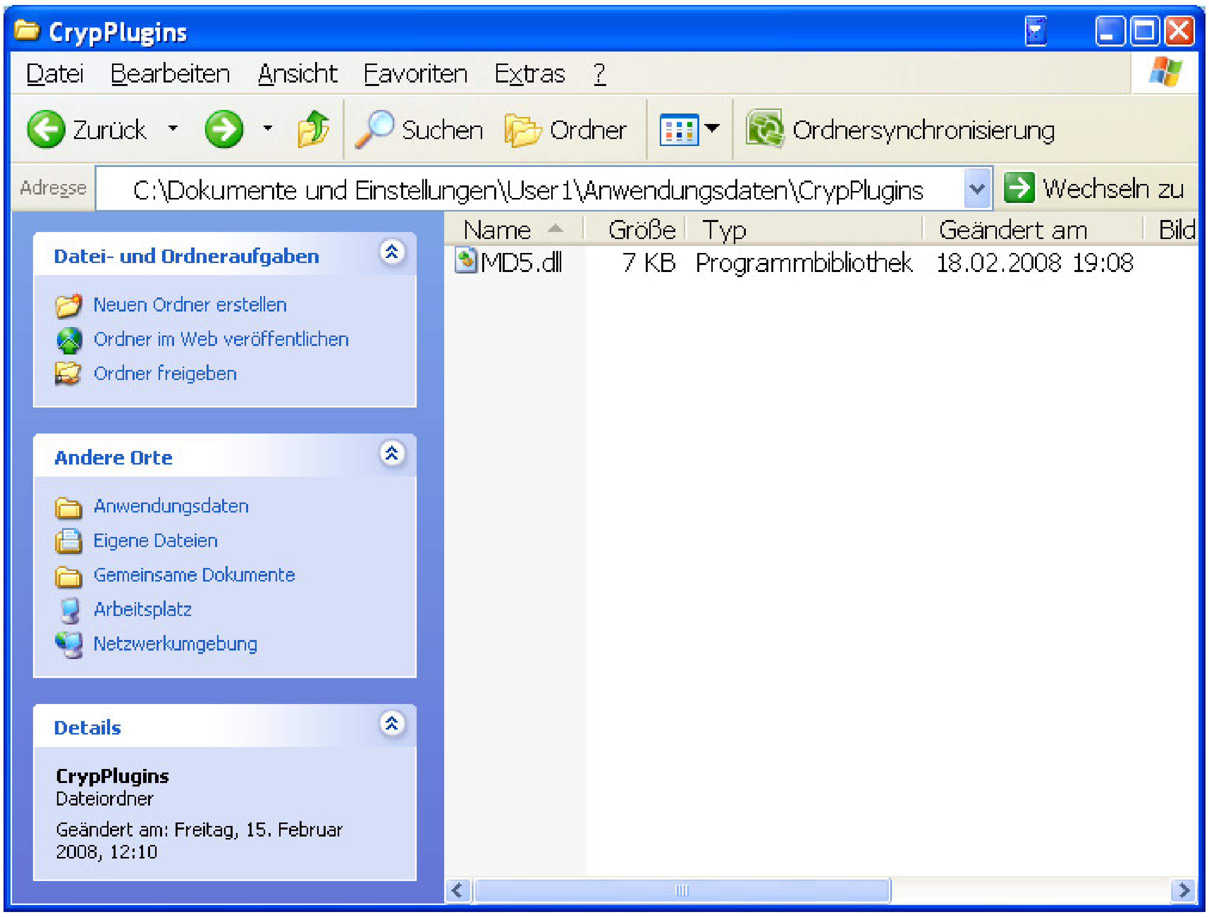
To test our new plugin in Cryptool, there are now three ways to do this:

1. Copy your plugin DLL file in the folder "CrypPlugins" which has to be in the same folder as the Cryptool executable, called "CrypWin.exe". If necessary, create the folder "CrypPlugins". This folder is called "Global storage" in the Cryptool architecture. Changes in this folder will take effect for all users on a multi user Windows. Finally restart Cryptool.



Name	Größe	Typ
CrypPlugins		Dateiordner
CrypCore.dll	9 KB	Programmbibliothek
CrypCore.pdb	18 KB	KL Palm Key File
CrypPluginBase.dll	11 KB	Programmbibliothek
CrypPluginBase.pdb	30 KB	KL Palm Key File
CrypWin.exe	163 KB	Anwendung
CrypWin.pdb	80 KB	KL Palm Key File
CrypWin.vshost.exe	15 KB	Anwendung
CrypWin.vshost.exe.manifest	1 KB	MANIFEST-Datei

2. Copy your plugin DLL file in the folder "CrypPlugins" which is located in your home path in the folder "ApplicationData" and restart Cryptool. This home folder path is called "Custom storage" in the Cryptool architecture. Changes in this folder will only take effect for current user. On a German Windows XP the home folder path could look like: „C:\Dokumente und Einstellungen\<User>\Anwendungsdaten\CrypPlugins"



C:\Dokumente und Einstellungen\User1\Anwendungsdaten\CrypPlugins

Name	Größe	Typ	Geändert am	Bild
MD5.dll	7 KB	Programmbibliothek	18.02.2008 19:08	

CrypPlugins
Dateiordner
Geändert am: Freitag, 15. Februar 2008, 12:10

3. You can also import new plugins directly from the Cryptool interface. Just execute CrypWin.exe and select the “Download Plugins” button. An “Open File Dialog” will open and ask where the new plugin is located. After selecting the new plugin, Cryptool will automatically import the new plugin in the custom storage folder. With this option you will not have to restart Cryptool. All according menu entries will be updated automatically. Notice, that this plugin importing function only accepts **signed** plugins.

   This third option is a temporary solution for importing new plugins. In the future this will be done online by a web service.

10. Source code and source template

Here you can download the whole source code which was presented in this “Howto” as a Visual Studio **solution**:

http://cryptool2.vs.uni-due.de/index.php/download

Here you can download the Visual Studio plugin **template** to begin with the development of a new Cryptool plugin:

http://cryptool2.vs.uni-due.de/index.php/development