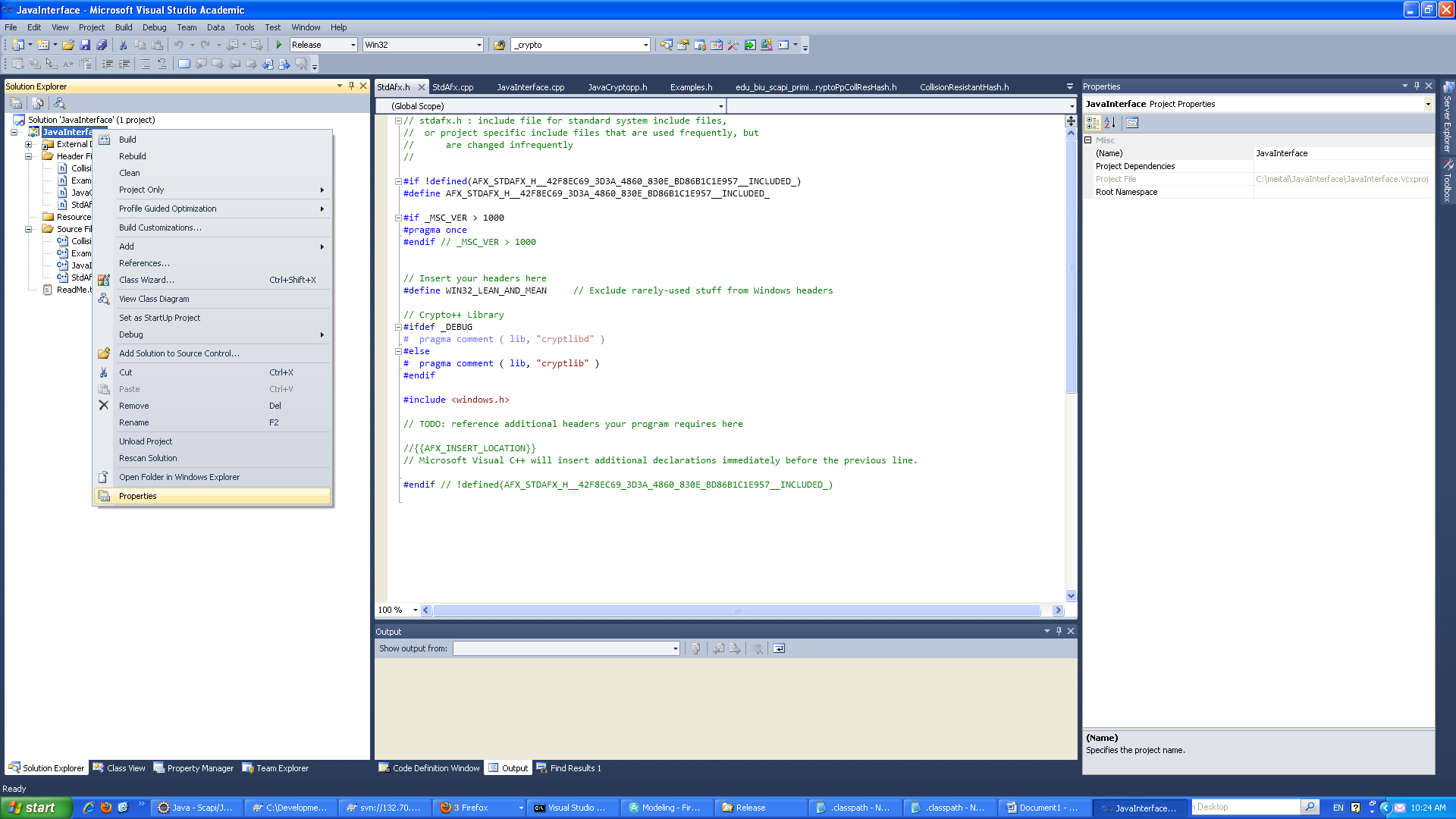
Working with crypto++ JNI in Scapi

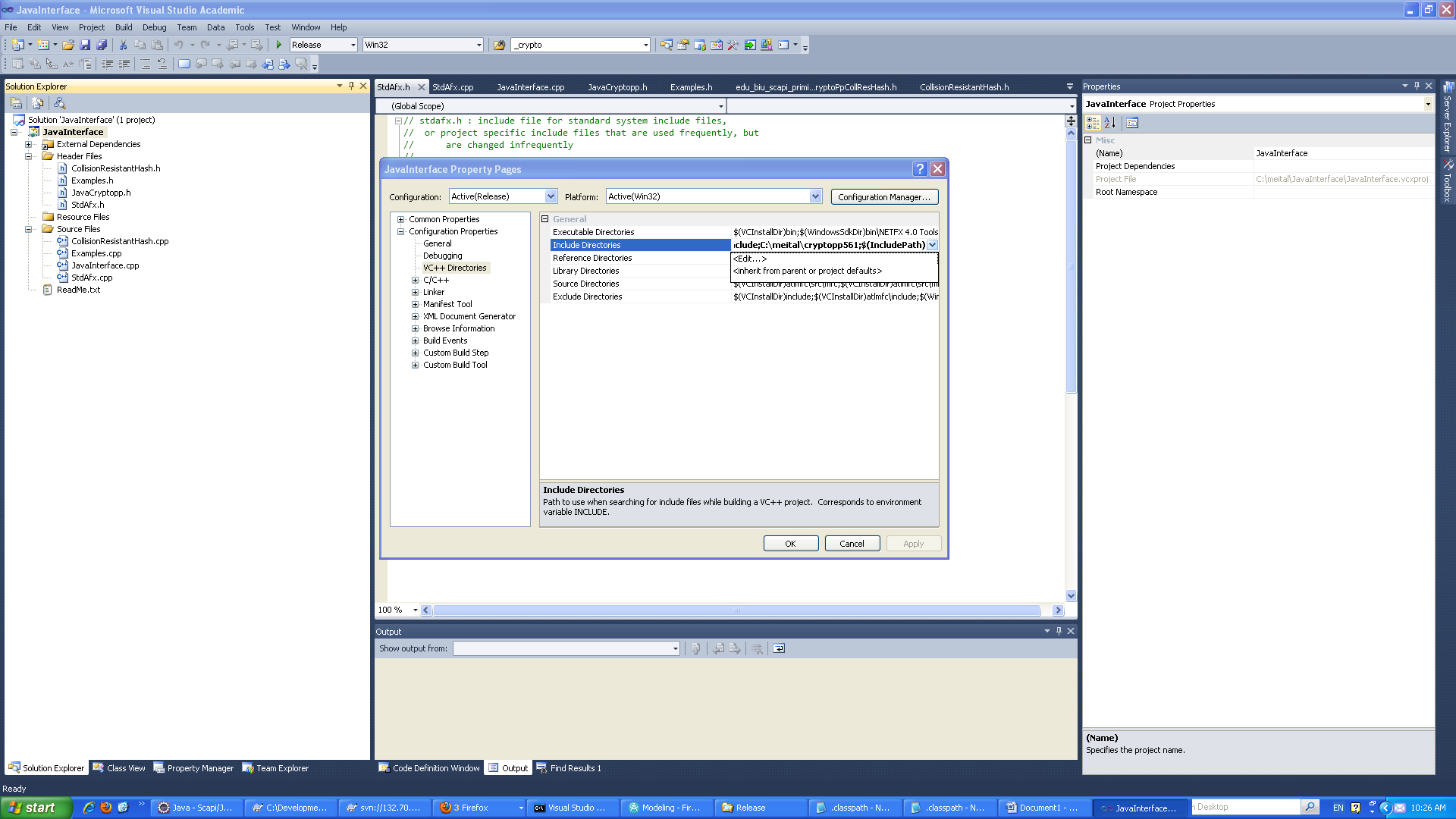
# Visual studio

1. Download existing code + solution of JavaInterface project from SCAPI's svn.
2. Download crypto++ project, or take only the h files of crypto++ from crypto++ web site.
3. Download compiled lib file of crypto++ called cryptlib.lib from SCAPI's svn . This file is compiled so it is compatible with the solution of the JavaInterface. It is compiled by us with visual studio 2010. It will be placed in the svn and will be compiled (by only one of us) each time crypto++ change their code. Each version will be stored together with its original crypto++ version number.
4. Place cryptlib.lib in some location on your computer.
5. Change in the project settings so it will take the relevant h files (of crypto++ and jni.h) and lib file from the right path as follows.

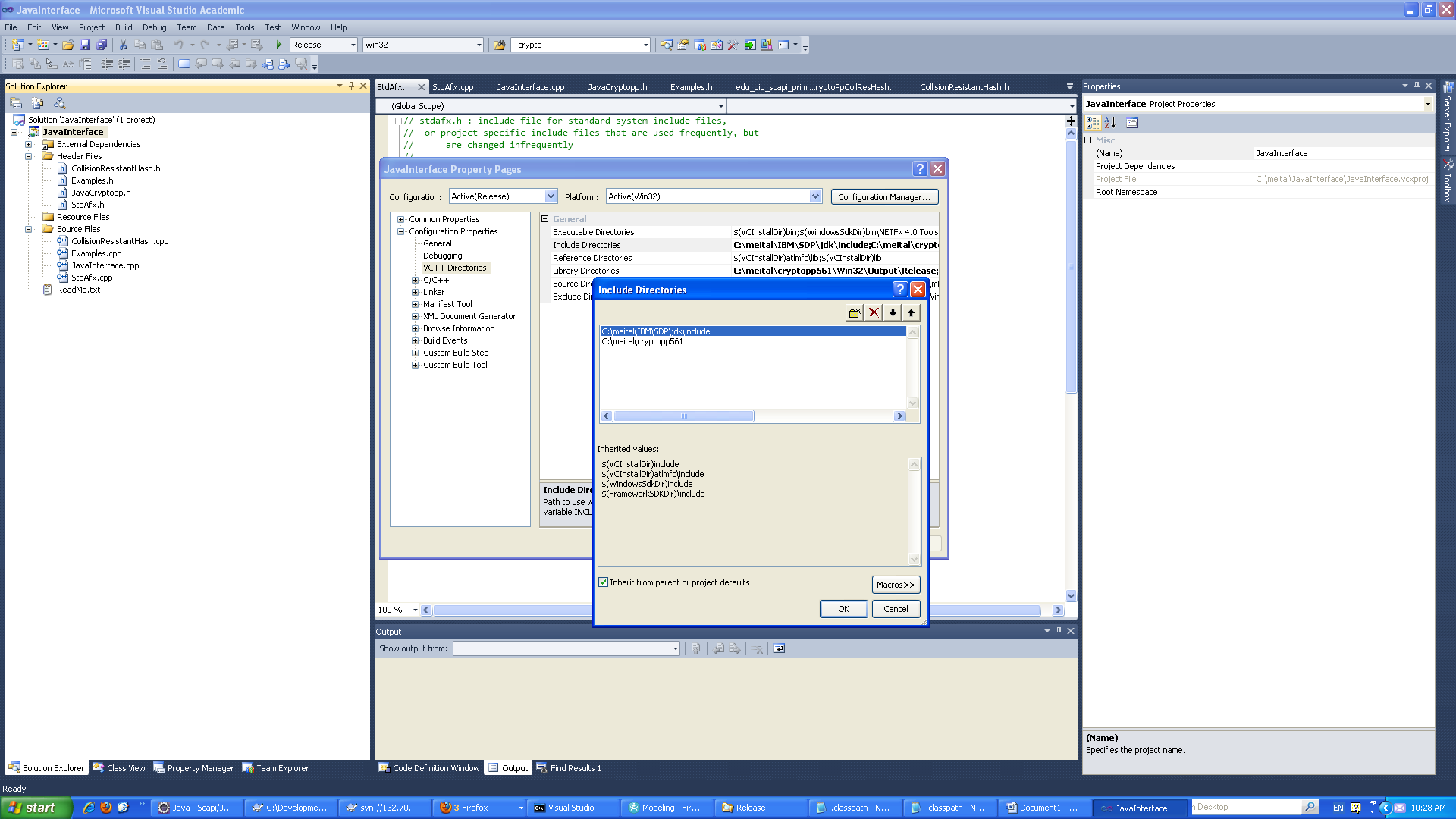
Right click on JavaInterface project and press properties



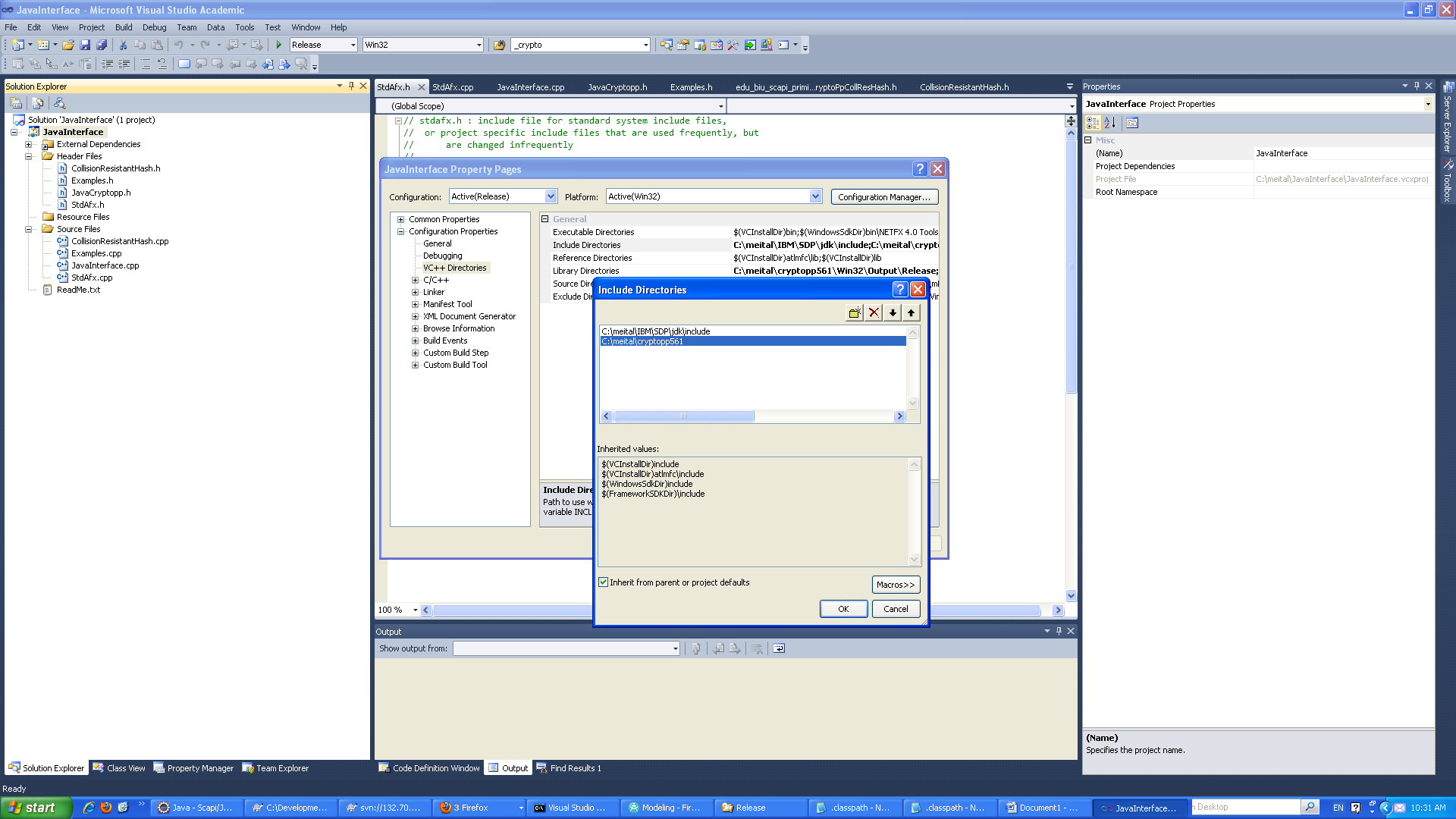
Go to Include Directories->edit



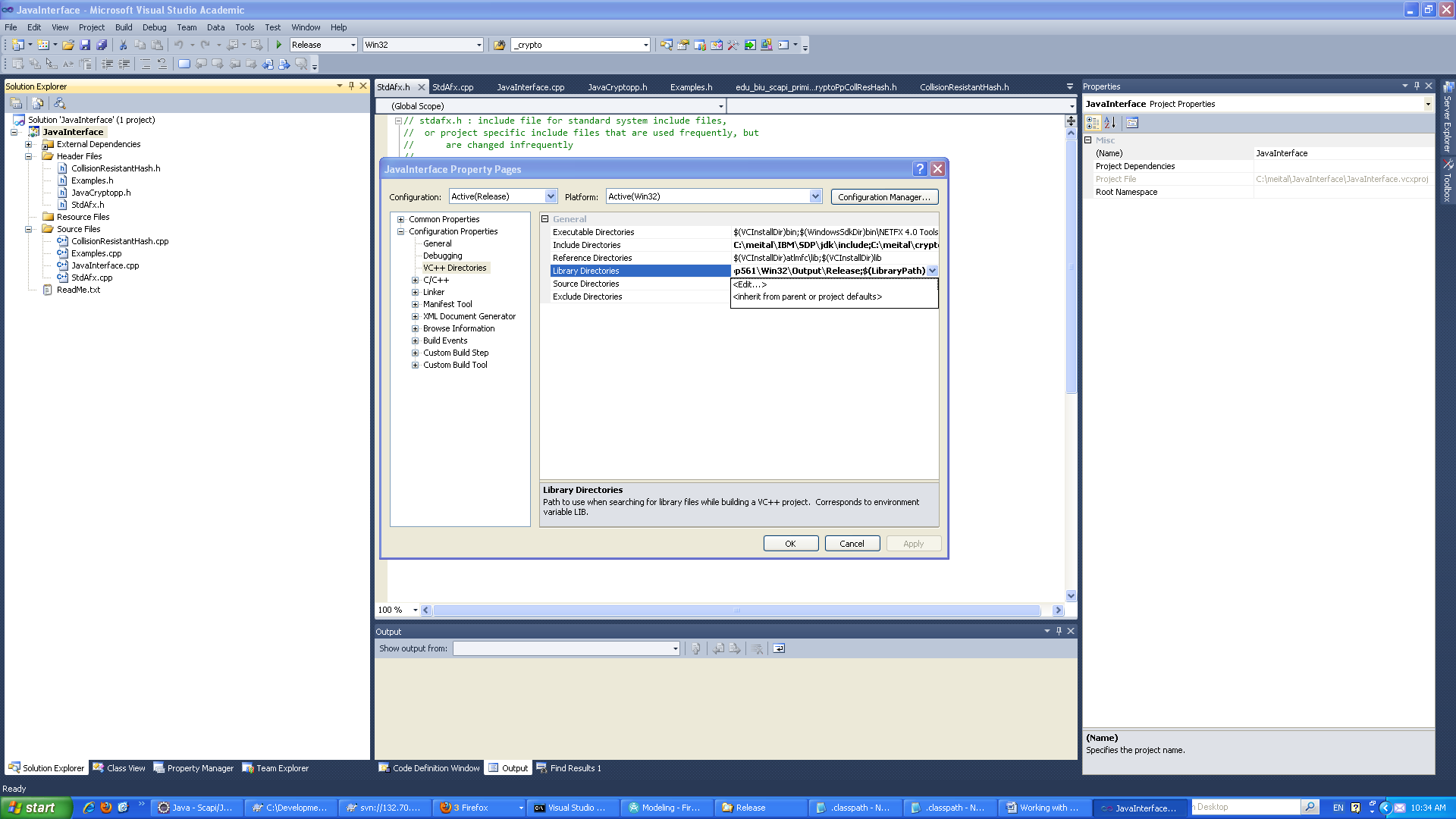
Locate your jni.h on your computer and add it to the include directories as follows.



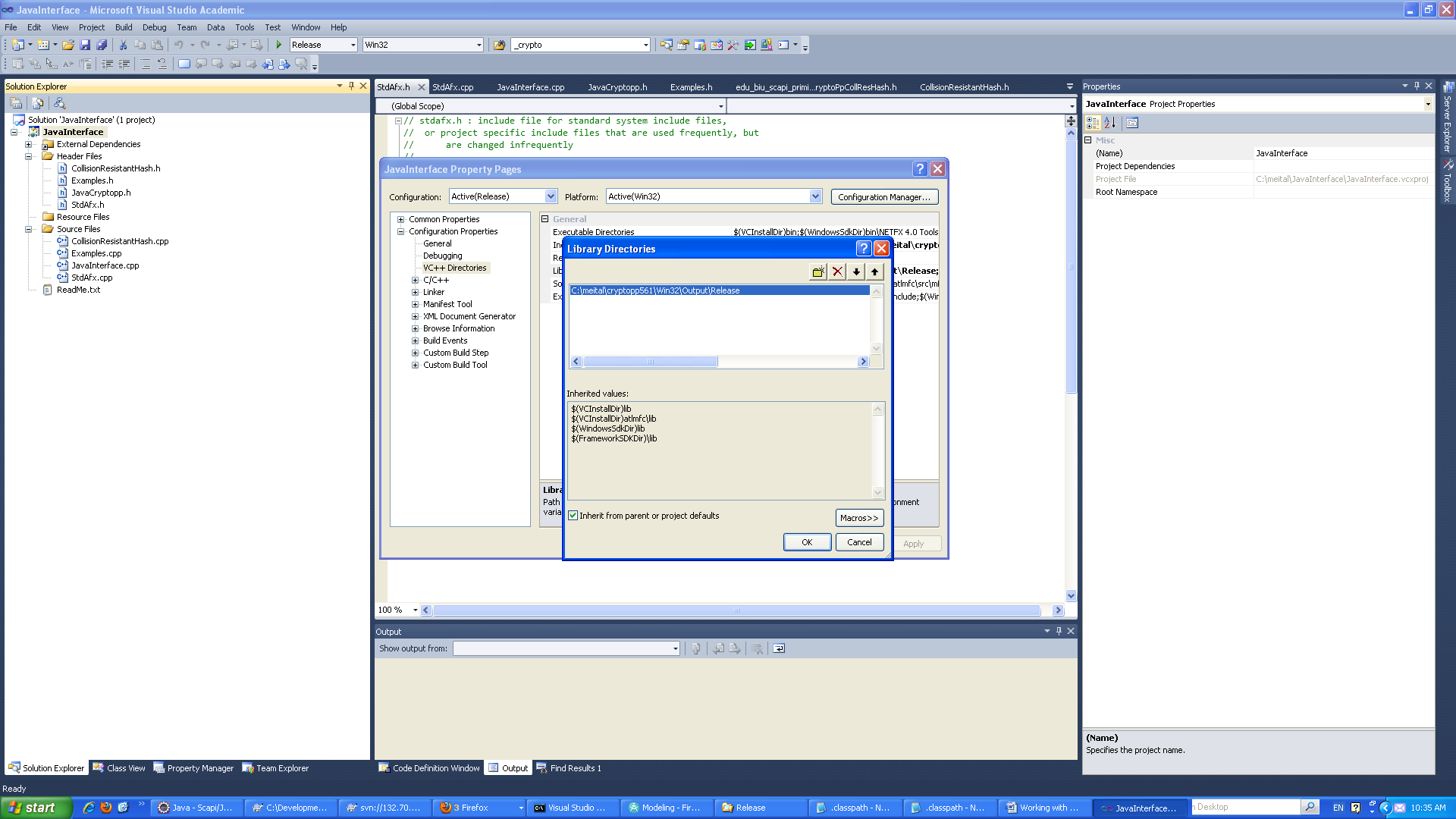
Locate your crypto++ header files directory on your computer and add it to the include directories as follows.



Go to Library Directories->edit



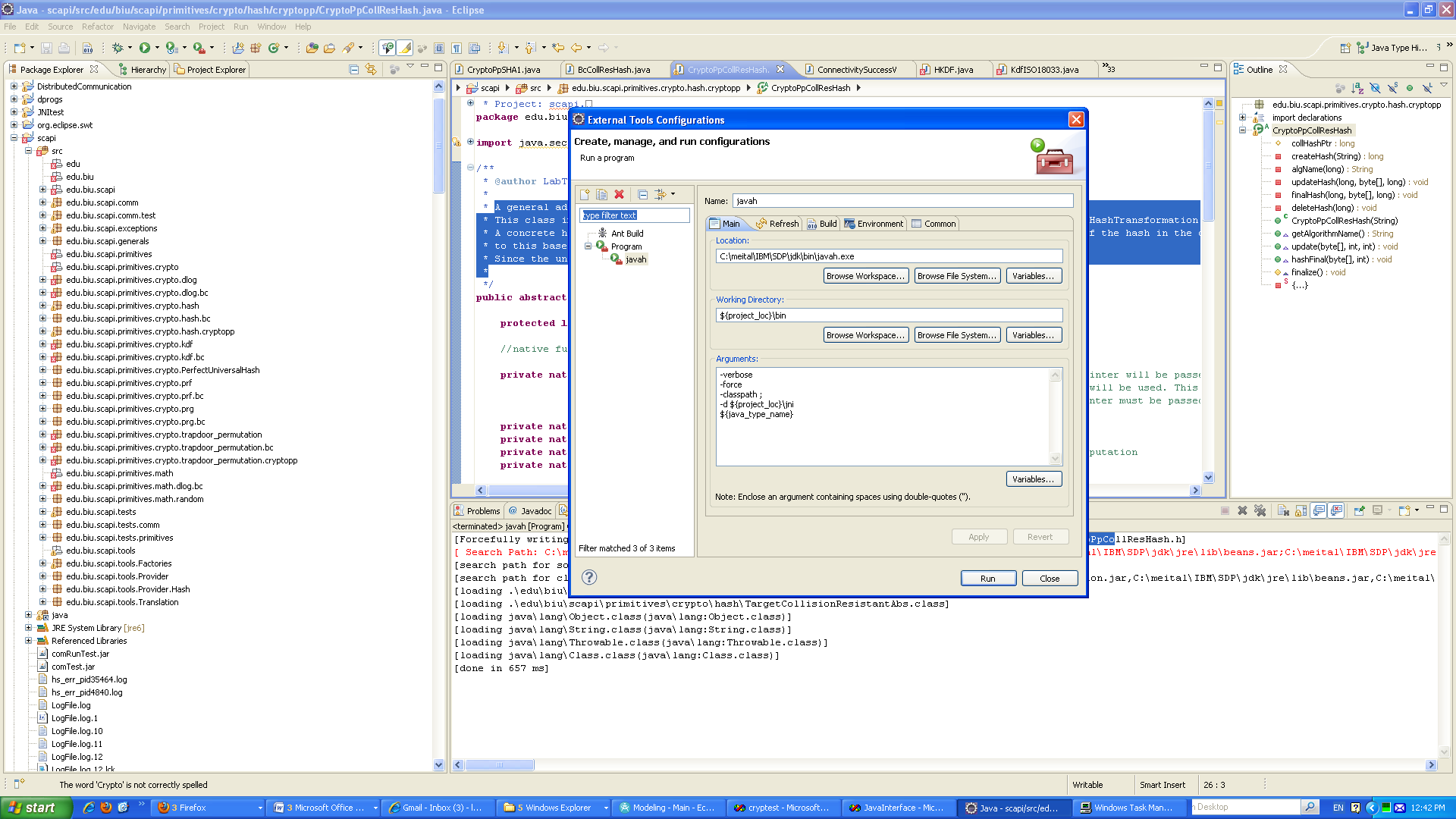
Locate your cryptlib.lib file on your computer and add it to the Library directories as follows.



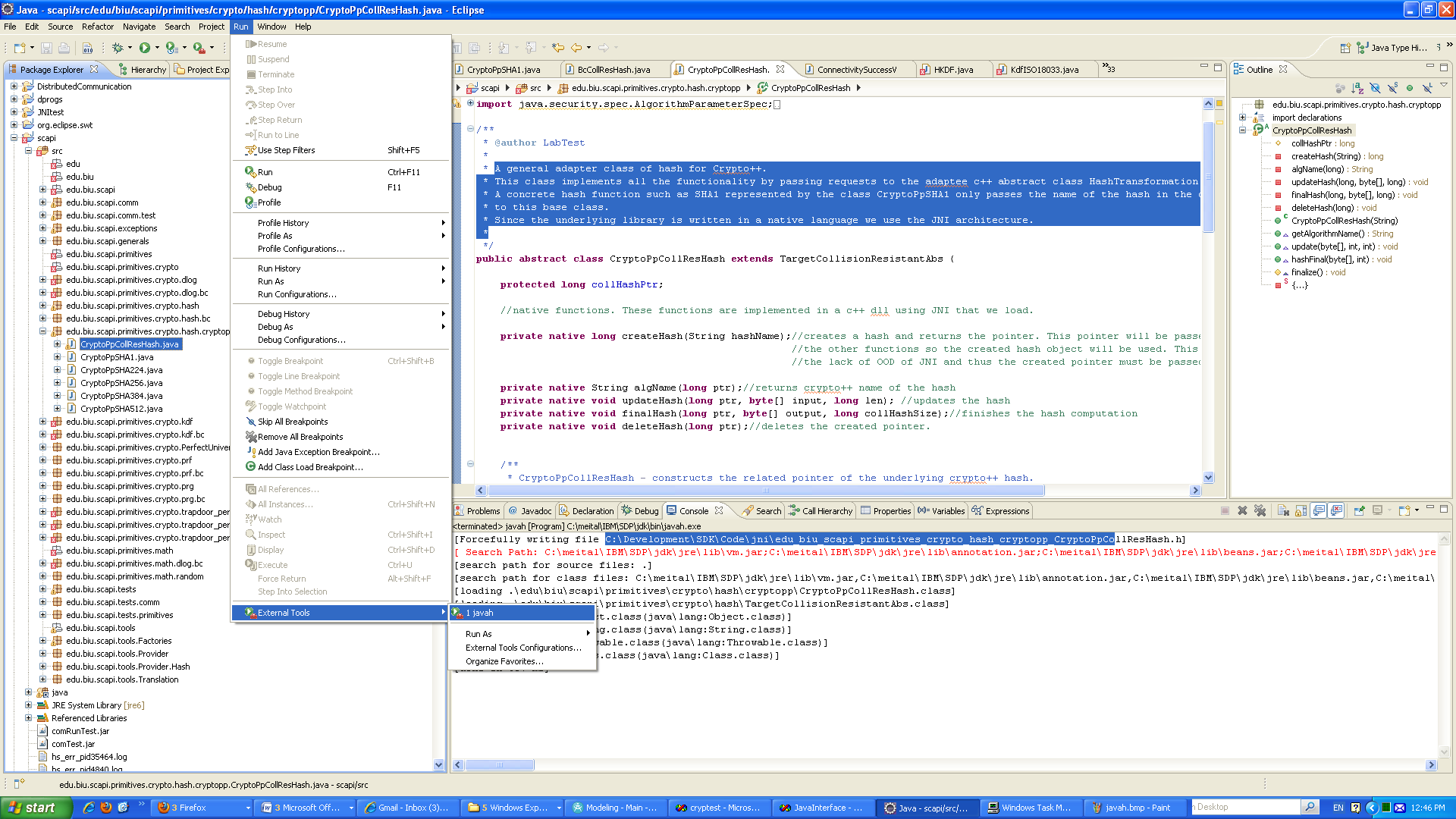
## Generating h file

1. **Create an external tool**. This should be done once.

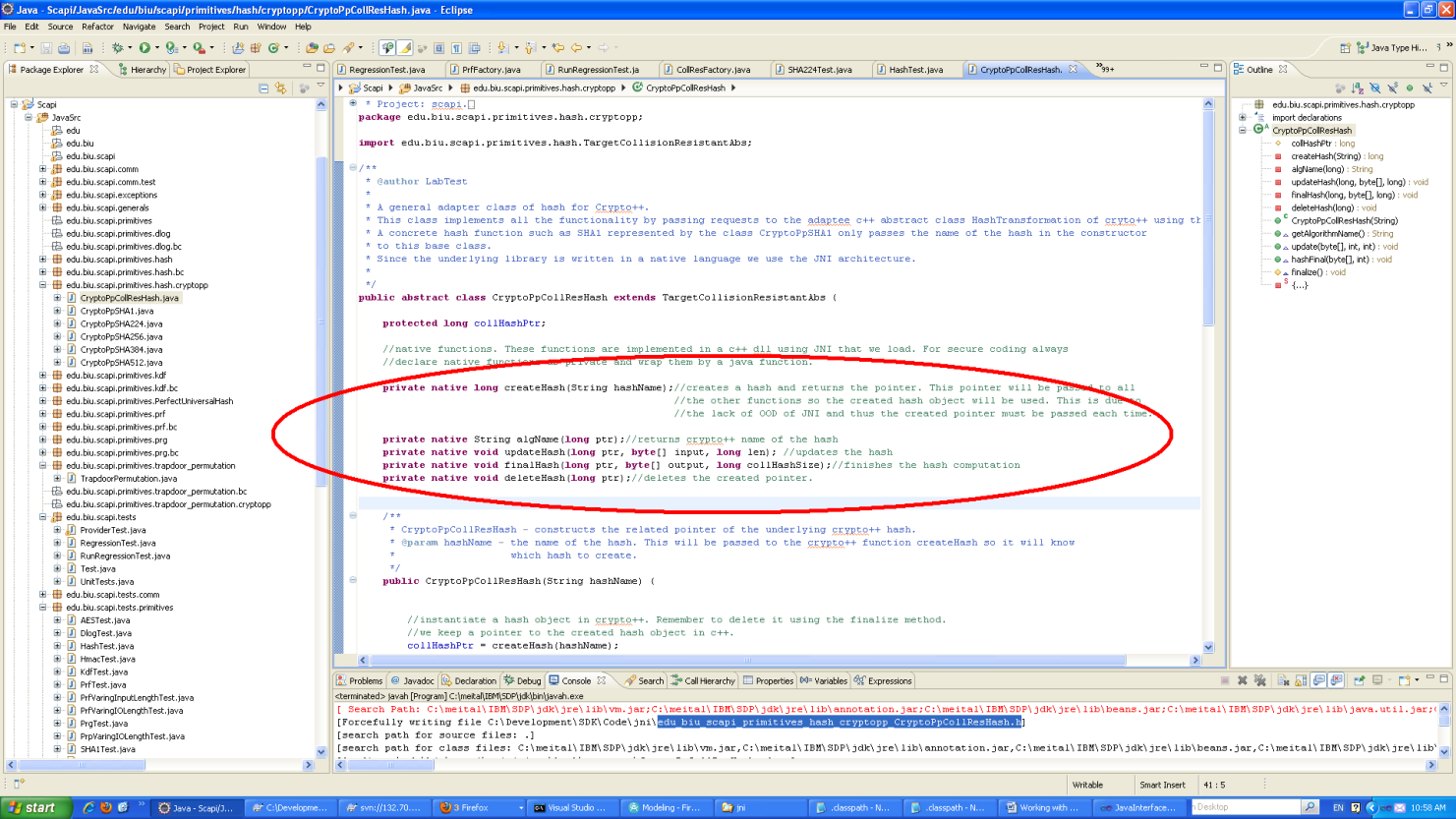
This h file is generated using the javah tool. For an easy use a simple external tool should be defined in eclipse as follows.



To generate the header file, select the related java file and run the javah external tool you have created as follows.



Note: In order for the tool to generate function signatures there must be some declaration of native functions. For example:



1. **Auto generated example file.**

An example of the generated h file from the external tool.

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

#include <jni.h>

/\* Header for class edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash \*/

#ifndef \_Included\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

#define \_Included\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

\* Method: createHash

\* Signature: (Ljava/lang/String;)J

\*/

JNIEXPORT jlong JNICALL Java\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash\_createHash

(JNIEnv \*, jobject, jstring);

/\*

\* Class: edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

\* Method: algName

\* Signature: (J)Ljava/lang/String;

\*/

JNIEXPORT jstring JNICALL Java\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash\_algName

(JNIEnv \*, jobject, jlong);

/\*

\* Class: edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

\* Method: updateHash

\* Signature: (J[BJ)V

\*/

JNIEXPORT void JNICALL Java\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash\_updateHash

(JNIEnv \*, jobject, jlong, jbyteArray, jlong);

/\*

\* Class: edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

\* Method: finalHash

\* Signature: (J[BJ)V

\*/

JNIEXPORT void JNICALL Java\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash\_finalHash

(JNIEnv \*, jobject, jlong, jbyteArray, jlong);

/\*

\* Class: edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash

\* Method: deleteHash

\* Signature: (J)V

\*/

JNIEXPORT void JNICALL Java\_edu\_biu\_scapi\_primitives\_hash\_cryptopp\_CryptoPpCollResHash\_deleteHash

(JNIEnv \*, jobject, jlong);

#ifdef \_\_cplusplus

}

#endif

#endif

1. **Cut and Paste to a file in the JavaInterface project.**

After the function signatures have been created cut and paste them to the related file in the JavaInterface project. IMPORTANT – DO NOT CHANGE THE SIGNATURE OF THE FUNCTIONS.

For example CollisionResistantHash.h.

