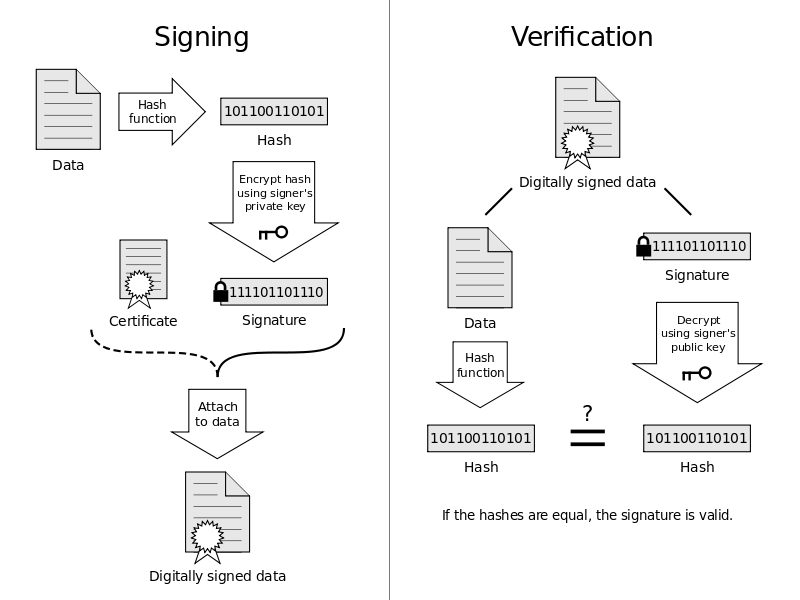
THE SIGNATURE SCHEME DEMO

1. Introduction

This documentation describes the demo for the signature scheme based on the Sodium crypto library. In summary**:**

- A console application (64 bit Windows environment) that supports to sign and verify a digital signature as mentioned in this diagram.



- Using the libsodium open source library (Signature: Ed25519, Hash: SHA-2).

1. The demo

You can run the demo by two ways:

* build and run by Visual Studio .NET 2013 or
* run the libsodium.exe file (in the source code zip file)

The demo includes 2 sub modules:

- The SIGNING module:

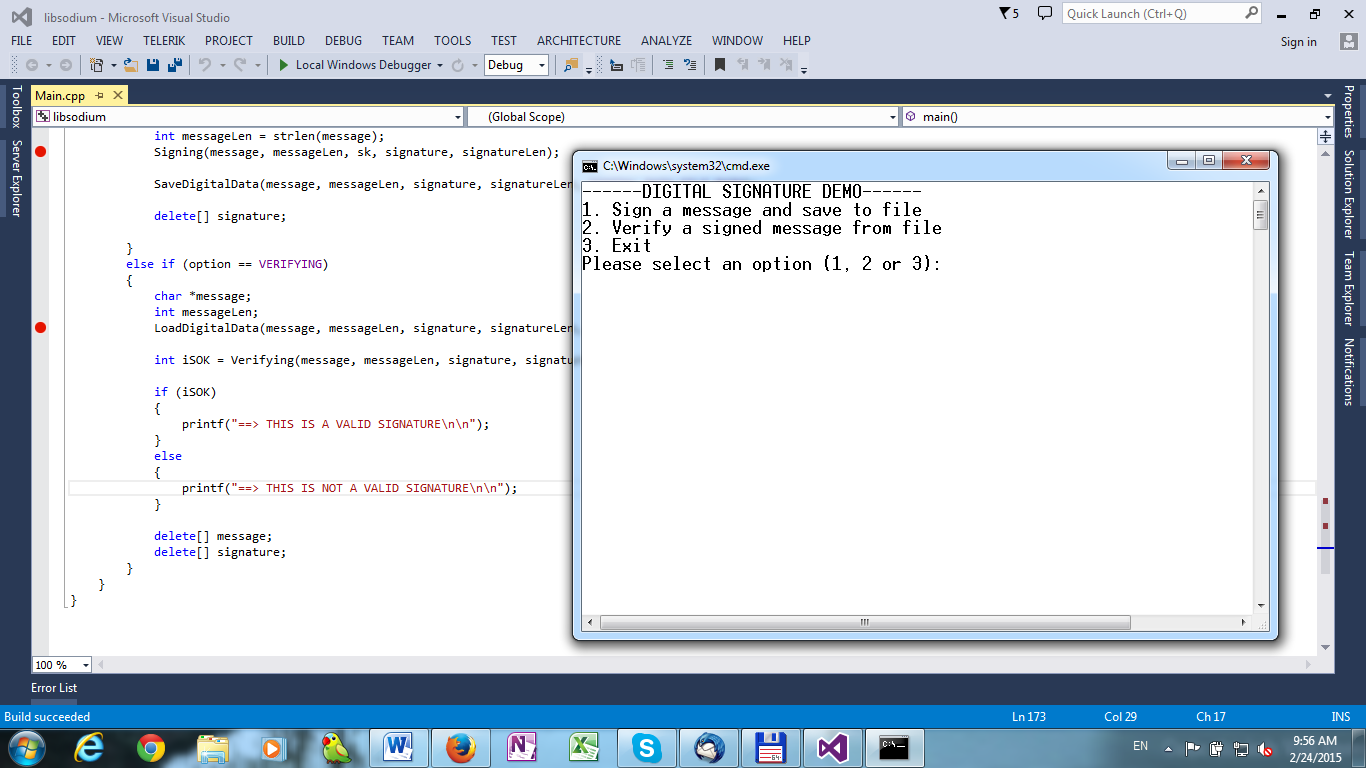
* User enter a test message
* The module hash this message by SHA-2
* The module encrypts the hashed message by a private key to have the signature.
* The module save the both the original message and the signature to a file

- The VERIFYING module

* The module load the original message and the signature from the file
* The module hash this message by SHA-2 (1)
* The module decrypt the signature by a public key (2)
* The module compares the result of (1) and (2) to see whether the signature is OK or not.

## The screenshots

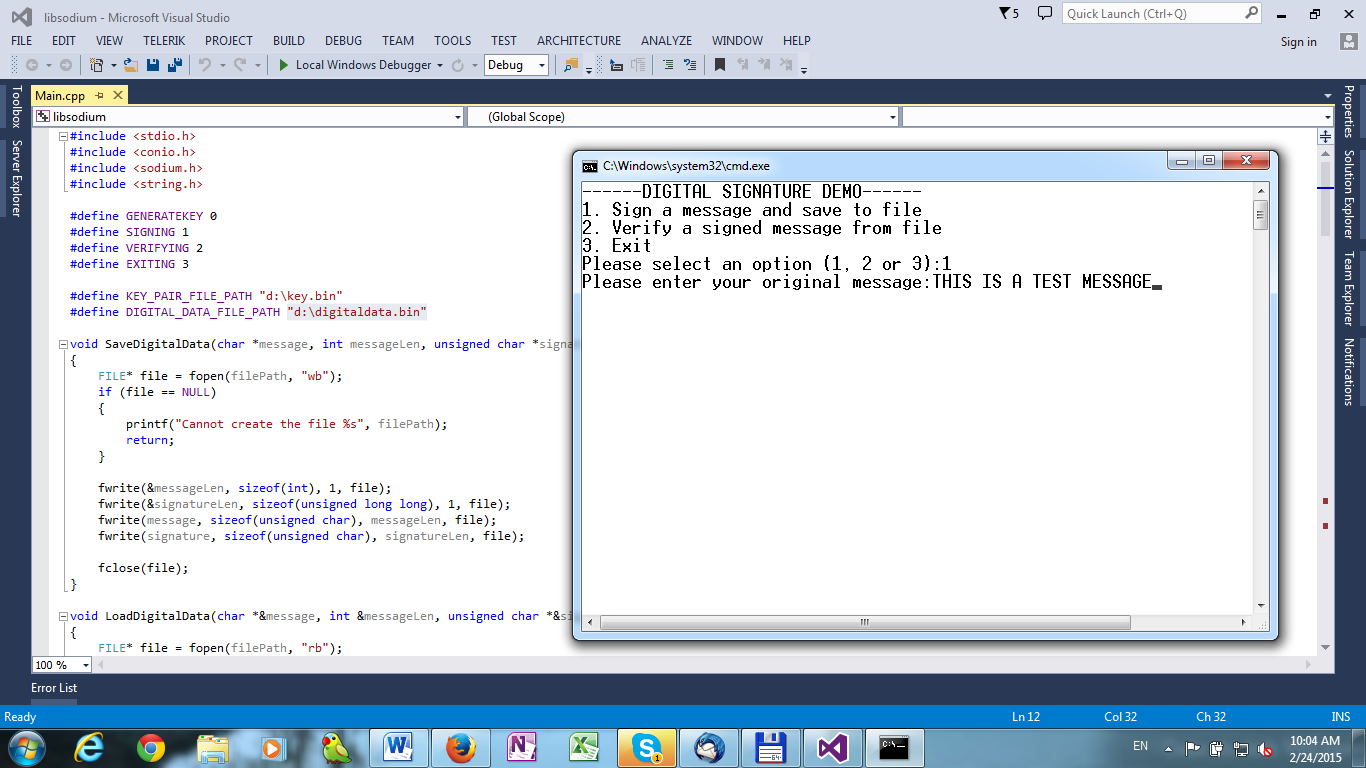
**Screenshot 1: The main menu**



At this screenshot, you can select the feature that you want to do. There are 3 options:

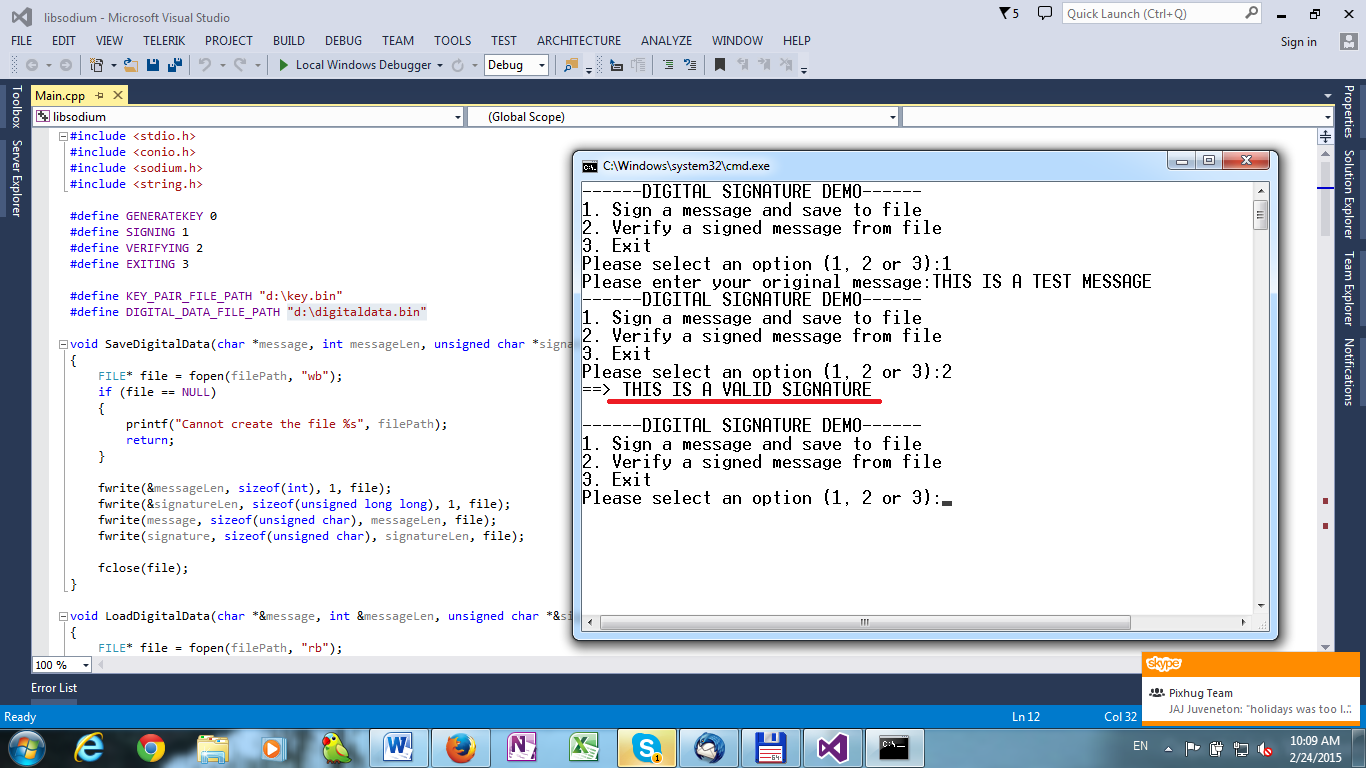
* **Option 1. Sign a message and save to file** =>You will enter a message, and then this message will be hashed and encrypted to create the digital signature (SIGNING module). This signature will be saved to a file (d:\digitaldata.bin).
* **Option 2. Verify a signed message from file** => The application will read the message and signature from the file (d:\digitaldata.bin), then check that whether the signature is OK or not (VERIFYING module).
* **Option 3. Exit the application**

**Screenshot 2: Sign a message**

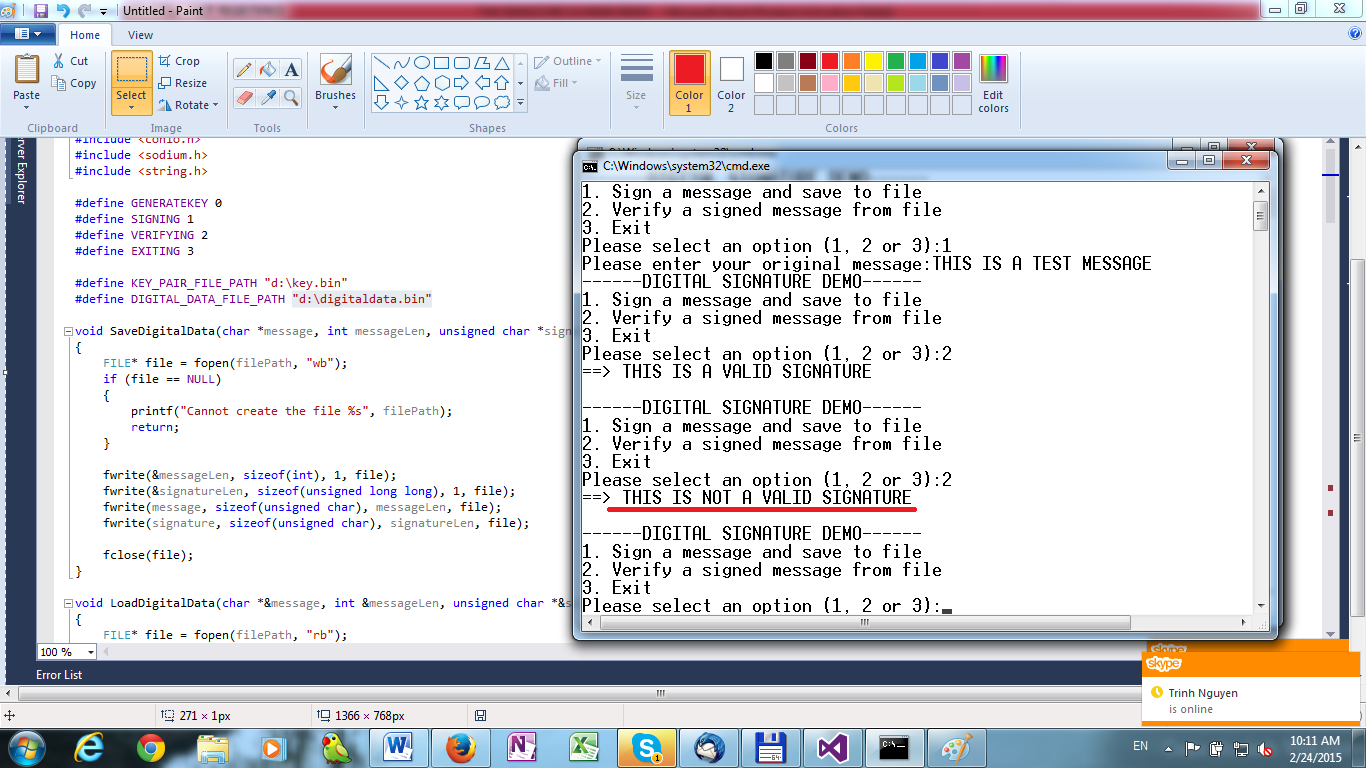


After this step, you will see that there is a new file located at “d:\digitaldata.bin” that stores the original message and the signature.

**Screenshot 3: Verify a signature (SUCCESS scenario)**



**Screenshot 4: Verify a signature (FAILED scenario)**

****

## How to test the demo

## Verify a signature: SUCCESS scenario. Please follow these steps:

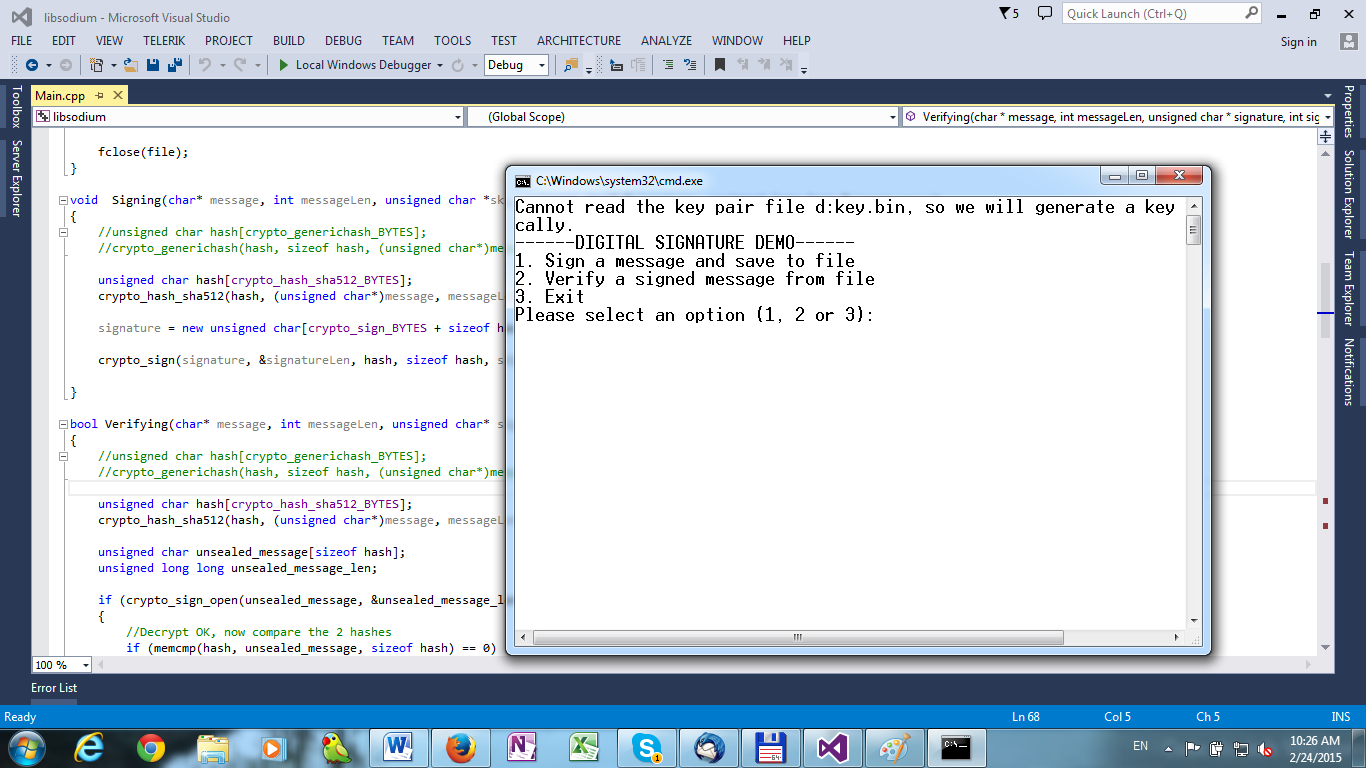
* Run the application
* Select option 1 (sign a message and save to file)
* Enter a test message (ex: “THIS IS A TEST MESSAGE”)
* Select option 2 (verify a signed message from file)
* You will see the screenshot 3.

## Verify a signature: FAILED scenario. Please follow these steps:

* Run the application
* Select option 1 (sign a message and save to file)
* Enter the test message (ex: “THIS IS A TEST MESSAGE”)
* Edit the “d:\digitaldata.bin” file (ex: add some more bytes, remove some bytes).
* Select option 2 (verify a signed message from file)
* You will see the screenshot 4.

## About the key pair

There is a public / secret key pair that stored at “d:\key.bin” file. The secret key is used when signing a message. And the public key is used when verifying a signature. When you run the application at the first time, you will see this screenshot.



If there is no key pair file, the application will generate a key pair automatically and saved it to the d:\key.bin file.

## What did I do to have this demo?

* Download the source code package of the Sodium library (http://download.dnscrypt.org/libsodium/releases/libsodium-1.0.1.tar.gz)
* Add a new code file Main.cpp at \libsodium-1.0.1\src
* Let the Visual Studio .NET 2013 build the exe file instead of the lib file.

1. The Sodium library

Sodium is a modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more. **It is a portable, cross-compilable, installable, packageable fork of NaCl, with a compatible API, and an extended API to improve usability even further.**

Its goal is to provide all of the core operations needed to build higher-level cryptographic tools.

**Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x64), iOS and Android**.

NaCl is a great library but it itself is not portable, only targeted for \*nix systems. The Sodium library solves this by making it portable and making a few minor changes to better suite being distributed as a compiled binary.

**Links:**

<http://labs.opendns.com/2013/03/06/announcing-sodium-a-new-cryptographic-library/>

<http://doc.libsodium.org/>