OpenPseudonymiser

DLL Integration Guide

Version No: 2.0.2b

Revision History

|  |  |  |
| --- | --- | --- |
| Revision date | Version | Summary of Changes |
| 17/09/2014 | 2.0.2 | Original |
| 18/09/2015 | 2.0.2a | Added information about the 128 random character generation of salt (new section section 4.3.1) |
| 20/02/2016 | 2.0.2.b | Added Licence |

OpenPseudonymiser documentation by the Julia Hippisley-Cox, University of Nottingham is licensed under a Creative Commons Attribution-NoDerivs 2.0 UK: England & Wales License. [www.openpseudonymiser.org](http://www.openpseudonymiser.org)

The OpenPseudonymiser software is issued under the GNU General Public License. University has made reasonable enquiries regarding granted and pending patent applications in the general area of this technology and is not aware of any granted or pending patent in Europe which restricts the use of this software. In the event that University receives a notice of perceived patent infringement, then University will inform users that their use of the software may need to or, if appropriate, must cease in the appropriate territory. University does not make any warranties in this respect and each user shall be solely responsible for ensuring that they do not infringe any third party patent.

Contents

[1 Overview 4](#_Toc443723266)

[2 Terms Used 4](#_Toc443723267)

[3 Licence 4](#_Toc443723268)

[4 Digest Creation 5](#_Toc443723269)

[4.1 Blank removal 5](#_Toc443723270)

[4.2 Order of columns 5](#_Toc443723271)

[5 Using the DLL 5](#_Toc443723272)

[5.1 DLL details and dependencies 5](#_Toc443723273)

[5.2 Using the DLL in Visual Studio 2010 5](#_Toc443723274)

[5.3 Example call 7](#_Toc443723275)

[5.3.1 Comparisons to the batch processor: 7](#_Toc443723276)

[5.4 Blank salt 7](#_Toc443723277)

[6 Encrypting the salt 8](#_Toc443723278)

[7 NHS Number validation and processing 9](#_Toc443723279)

[8 Appendix A 10](#_Toc443723280)

[8.1 Test1 10](#_Toc443723281)

[8.2 Test2 10](#_Toc443723282)

# Overview

The University Of Nottingham has created an Open Source standalone windows desktop application called OpenPseudonymiser which is available for download at www.openpseudonymiser.org

The application allows users to pseudonymise datasets by creating a digest of one or more columns of a CSV file.

The application uses a DLL for the digest creation. The DLL is made available to supplies who wish to integrate this in their system. This document describes how to use the DLL in other projects.

# Terms Used

**Input:** A concatenation of the fields the user has selected to use in the creation of the Digest (e.g. NHSNum + DOB)

**Salt:** Extra characters added to the input

**Digest:** The long string that comes out of the cryptographic hash function

# Licence

OpenPseudonymiser is free software: you can redistribute it and/or modify it under the terms of the GNUGeneral Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenPseudonymiser, including the website, software, documentation and key server technology, is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Organisations who wish to make use of the OpenPseudonymiser technology have full responsibility for regarding information governance and security considerations relevant to their purposes. The Key Server is intended for demonstration purposes only. Organisations wishing to use OpenPseudonymiser for production purposes should deploy an instance of the software/key server etc suitable for their own purposes in order to satisfy their own information governance and security requirements.

You should have received a copy of the GNU General Public License along with OpenPseudonymiser. If not, see <http://www.gnu.org/licenses/>

OpenPseudonymiser makes use of the following Open Source libraries:

RSAEncryption Class Version 1.00 which is Copyright (c) 2009 DudiBedner

BigInteger Class Version 1.03 which is Copyright (c) 2002 Chew Keong TAN

NHSNumber-Validation which can be found at <https://github.com/pfwd/NHSNumber-Validation>

# Digest Creation

The digest is a SHA-2 (SHA256 variant) hash of the concatenated columns with the salt appended to the end.

e.g. if the columns NHSNumber and DOB were selected with the salt “mackerel” then the digest creation would follow the steps:

* “29.11.2011” + “9434765919” + ”mackerel”
* concatenated to: “29.11.20119434765919mackerel”
* run through SHA256
* The digest = “5dfc32ba81ea3e016333687111ae2f63d97dad05adf92c61bf06438a08d8bc56”

Note that all the inputs are treated as strings, different formats for the DOB (slashes rather than dots, 2 rather than 4 digits” will change the digest. It is therefore very important to agree on standardised formats for all the fields you plan to use in the creation of the digest.

## Blank removal

Before any column is used in the digest all “blanks” are removed from it. Newline, Carriage Return, Space and Tab are all removed from any data before being used in the Digest. . This is a very important point and means OpenPseudonymiser should not be used to create digests for things like full names, postcodes and other data that may contain spaces.

## Order of columns

The input columns will always be arranged alphabetically. In the above example the DOB column comes **after the** NHS Number column in the input file, but DOB is concatenated **before** NHS number because “DOB” comes before “NHS Number” alphabetically. This is all handled automatically by the DLL.

# Using the DLL

This section describes how to use the DLL.

## DLL details and dependencies

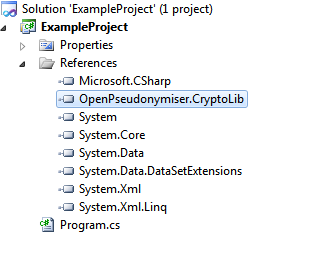
The DLL is called: OpenPseudonymiser.CryptoLib.dll

The assembly is built for .NET Framework 2.0 targeting “Any CPU”

## Using the DLL in Visual Studio 2010

Create a new console application in Visual Studio

Go to References, Add Reference  
Browse to the DLL and include it as a reference:



Add the line:  
  
using OpenPseudonymiser;

to the top of your code

You can now instantiate the Crypto object using the following line of code:

OpenPseudonymiser.Crypto crypto = new Crypto();

## Example call

The following code is an example of how to call the CryptoLib DLL

bool success = true;

OpenPseudonymiser.Crypto crypto = new Crypto();

// set the salt to a plain text word/phrase

string salt = "mackerel";

crypto.SetPlainTextSalt(salt);

// The input: a name/value pair

var nameValue = new SortedList<string, string>();

// any spaces in the special case field called 'NHSNumber' will be stripped out

nameValue.Add("NHSNumber", "9434765919");

// even though we add DOB after we add NHS, it will come before NHSNumber in the input, since the SortedList will always order by alphabetical key

nameValue.Add("DOB", "29.11.1973");

// Call the GetDigest method and receive the digest..

string digest = crypto.GetDigest(nameValue);

// we expect the following digest for the above values

success = (digest == "ED72F814B7905F3D3958749FA90FE657C101EC657402783DB68CBE3513E76087");

Console.WriteLine("Test for (nonEncryptedSalt): " + success);

Console.WriteLine("Press any key to finish...");

Console.ReadKey();

### Comparisons to the batch processor:

When creating salt on the OpenP website for use in the Batch Processor software please be aware that the salt is always created using a 128 character string. The text entered into the webpage to create the salt “seeds” this 128 character string. Therefore to compare outputs from the DLL and the Batch Processor you would need to use a plain text salt that is exactly 128 characters long in both systems. Please see Desktop (Batch Processor) User Guide Appendix A

## Blank salt

Blank salt is not allowed, the DLL will throw an exception if a call to GetDigest is made with either no salt set, or a blank string set as the salt.

# Encrypting the salt

It is possible to call the DLL without knowledge of the salt. Using encrypted salt provides another level of security by removing knowledge of the salt data from the users of DLL.

The site www.openpseudonymiser.org allows you to create encrypted salt files for use with the DLL. The salt file is encrypted using a PKI (Public Key Infrastructure) technique. The salt word is encrypted using a private key known only to the University of Nottingham (the owners of the www.openpseudonymiser.org site)

The encrypted salt file can be used with the DLL in the same way as the example call in section 4.3 with the following change:

Instead of calling:

// set the salt to a plain text word/phrase

string salt = "mackerel";

crypto.SetPlainTextSalt(salt);

Do this instead:

string pathToEncryptedSalt = Environment.CurrentDirectory;

string encryptedSaltFileLocation = pathToEncryptedSalt + "\\mackerel.EncryptedSalt";

byte[] encryptedSalt = File.ReadAllBytes(encryptedSaltFileLocation);

crypto.SetEncryptedSalt(encryptedSalt);

Replacing the location of your encrypted salt file as appropriate

# NHS Number validation and processing

There are two methods in the DLL that can be used to optionally process and validate an NHS Number as the following code demonstrates:

OpenPseudonymiser.Crypto crypto = new Crypto();

bool success = true;

string processedNHSNumber = "";

// A call to ProcessNHSNumber will strip all non-numeric characters from the string

processedNHSNumber = crypto.ProcessNHSNumber("4505577104");

// A call to the static "IsValidNHSNumber" method will return a true if the string passes the NHS number validation checksum as described here:

// http://www.datadictionary.nhs.uk/data\_dictionary/attributes/n/nhs\_number\_de.asp

success = success & (NHSNumberValidator.IsValidNHSNumber(processedNHSNumber));

Console.WriteLine("NHSNumber validation test 1: " + success);

**The source code for the application is available on the above site. There is also a simple console application that runs these examples.**

# Appendix A

Some results of speed testing on various systems:

## Test1

6 million rows of data with 14 columns (a 600MB CSV file) on a core i5 laptop takes just under 3 minutes.

## Test2

82 million rows with 11 columns (a 11GB CSV file) on a Dell Poweredge (6x 7,200 rpm drives in Raid5) takes 32 minutes.