OpenPseudonymiser  
SQL Server CLR

User Guide

Version No: 2.0.2b

Revision History

|  |  |  |
| --- | --- | --- |
| Revision date | Version | Summary of Changes |
| 03/12/2013 | 1.0 | First draft |
| 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.

# Overview

This document assumes knowledge of the OpenP software purpose and process.

Full documentation for the software can be found on the site: [http://www.openpseudonymiser.org](http://www.openpseudonymiser.org/)

OpenP\_CLR is a SQL server database with the functions of the Crypto.DLL exposed as SQL CLR functions. This document explains how to install the database and call the functions.

# 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>

# Installation

1. Restore the database OpenP\_CLR from the provided SQL Server Backup File. The backup was created using SQL Server 2008 so you’ll need this version of SQL server (or later)
2. SQL Server needs to be able to access your filesystem to be able to read the Encrypted Salt files. This requires that the database is 'trusted'.   
   You'll need to run this on the database once you've installed it:  
     
   /\* Trustworthy is required for the CLR  file system access to work in the GetDigestUsingEncryptedSaltFile function \*/

alter database [OpenP\_CLR] SET Trustworthy on  
RECONFIGURE;  
go  
  
You may also need enable CLR for the database

USE [OpenP\_CLR]    /\*\*CHANGE ME TO YOUR DATABASE NAME HERE\*\*/  
GO  
sp\_configure 'clr enabled', 1  
go  
RECONFIGURE;  
go  
  
SQLServer can only access files that is it allowed to see through the operating system. The SQL service runs under a 'security context' or a user on your system. You can check your setting in Control Panel/ Services.  
One option is to put your encrypted salt file in a folder that is generally available to users of the system. In Windows 7 there is a built in one called C:\Users\Public.

To test that SQL has access to the file try and call the GetDigestUsingEncryptedSaltFile method by giving it the path of a file. If it fails try changing the location of the file.

# Calling the functions

## GetDigest

This function will produce a digest of the first parameter using the second parameter as plan text salt:

Select dbo.GetDigest('foo', 'pie')

## GetDigestUsingEncryptedSaltFile

This function will produce a digest of the first parameter using the path to encrypted salt for the second parameter. If this function returns an error it is likely that SQL server does not have the ability to read the file you have provided. Try moving the file to a shared folder.

/\* should yield the same result with encrypted and non-encrypted versions of the salt\*/  
Select dbo.GetDigestUsingEncryptedSaltFile('foo', 'C:\Users\Public\thewordPie.EncryptedSalt')

## Pre loading the salt file

If you have millions of rows to process it will likely be quicker to load the salt file first, rather than using the GetDigestUsingEncryptedSaltFile method.   
The GetDigestUsingEncryptedSaltFile method has to read the salt file in for each row you process.

## Store the salt first with StoreEncryptedSalt

EXECUTE [OpenP\_CLR].[dbo].[StoreEncryptedSalt] @locationOfFile = 'C:\Users\Public\thewordPie.EncryptedSalt'

## Then call a function that uses this stored salt:

Select dbo.GetDigestUsingStoredEncryptedSalt('foo')  
  
or over a table:  
  
Select top 10000 dbo.GetDigestUsingStoredEncryptedSalt(([PATIENT\_ID]))  
, [PATIENT\_ID]  
  FROM [Nato\_Jan2011\_Filer3].[dbo].[OBSERVATIONS]