w



**Project:**

**DlpServer.KeySafe**

**Component: DlpServer.KeyManagement.exe**

|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Comments** |
| Emad Mohammad | 20th Feb 2017 | First version |
|  |  |  |
|  |  |  |

Contents

[1. Functionality of DlpServer.KeyManagement.exe 3](file:///C:\Users\Hooman\Downloads\GWERecTool%20(1).docx#_Toc403728813)

[2. Overview of KeyManagement project 3](file:///C:\Users\Hooman\Downloads\GWERecTool%20(1).docx#_Toc403728814)

[3. KeyManagement flowchart 5](file:///C:\Users\Hooman\Downloads\GWERecTool%20(1).docx#_Toc403728818)

## Functionality of KeyManagement.exe

The purpose of KeyManagement.exe is to generate the user private key in the server side upon request from the client and securely pass the encrypted userkey to the client and then decrypt at the client.

## Overview of KeyManagement project

KeyManagement (KeySafe) is a C# project in MS Visual Studio 2012.

Some project’s configuration properties that developers should keep in mind are:

1. Project uses the **Win32 / x 64** configurations.
2. Project uses the **Release** / **Debug** configurations.
3. Project uses Unicode
4. Project uses BouncyCastle library
5. Project uses EntityFramework 5.0
6. Lib files required GWECrypto

The list of files within KeyManagement(KeySafe) project are:

**Source Files**

* **Key\_safe.cs**

This file implements “GWECrypto.dll” and divides various encryption decryption in separate functions.

* **KeyGenerationService.cs**

*This file contains the main class that implements the whole process. The class runs a service , that connects to database, gets various keys, upon which operations are performed. This class actually generates the encrypted user private key, upon request from client, which sends userkey id over sockets. Then, the private key is decrypted and the required user private key is returned. For safe transfer of key from server to client, AES and RSA are implemented together. First, the required private key is encrypted with AES key and then encrypted private key is sent to the client. After this, RSA public/private keypair are generated in client , then public key is sent to the server. In the server, this public key is used to encrypt the AES key that was used to encrypt the private key. After this, encrypted AES key is sent to the client. At the client side, first encrypted AES key is decrypted with the RSA private key, then this decrypted AES key is used to decrypt the user private key.*

* KeyGenerationService.Designer.cs
* Program.cs

**Configuration Files**

* App.config

The data to the configuration file “dbsettings.config”, can be change here. The database connection string information is to be given here.

**Input Files**

* “C:\\ProgramData\Guardware\DLP\dbsettings.config”: contains information about the database connection string, particular keytype and password that is used to for the decryption of keys is saved in this file.

**Output Files**

* “LOGS\”: The folder contains debugging information about database connections, encryptions that are generated upon some error.

**Php Files**

* On the server side, as part of the client request php file “*get\_key.php*” is invoked by the client for the key generation. This php file gets the clients or user key Id. This file then passes the user key Id to another php file, named “clientSoc.php” which is placed inside the folder named,” *phpseclib1.0.4*”. The php is placed inside this folder for ease of implementation of algorithms like “RSA” and “AES” for secure transfer of key. The “clientSoc.php” uses the user keyId and generates the public key using RSA and sends both of them to the server using sockets. Then upon getting the encrypted user private key and AES key from the server, then RSA private key is used for the decryption of AES key and then decrypted AES key is used to decrypt the user private key.

The location of the GWERecTool project in e-safe Systems’ SVN Server repository is:

[http://10-SERVER:8084/svn/DLP/Beta4/Server/KeyServer\_Keysafe\_2](http://10-server:8084/svn/DLP/Beta4/Server/KeyServer_Keysafe_2)

## KeyManagement flowchart

SERVER SIDE CLIENTSIDE

In the server, encrypted private key is generated

User requests for the private key and userkeyId is sent

Then, the private key is decrypted

RSA private/public key pair generated

Encrypted private key is decrypted with AES key

Encrypted AES key is decrypted with RSA private key

Public key sent to server

The encrypted AES key sent to client

The AES key encrypted with RSA public key

Encrypted private key sent to client

Private key encrypted with AES