# Introduction

First, decide on the method for protecting the confidentiality of the OpenPGP Private Keys’ secret passphrase.

This solution offers 2 options:

1. Use AWS Secrets Manager [Recommended for Systems hosted in AWS]
2. Use Windows Data Protection API [Only works for Systems developed for Windows OS]

Please follow the Setup Steps provided in this document for the chosen option.

Contents

[Introduction 1](#_Toc130855945)

[1. Setup Steps to protect Secret Passphrase using AWS Secrets Manager 2](#_Toc130855946)

[2. Setup Steps to protect Secret Passphrase using Windows Data Protection API 8](#_Toc130855947)

# Setup Steps to protect Secret Passphrase using AWS Secrets Manager

1. Login to AWS Console and go to AWS Secrets Manager Console

<https://ap-southeast-1.console.aws.amazon.com/secretsmanager/newsecret?region=ap-southeast-1>

Graphical user interface, application

Description automatically generated

Figure 1: Choose to create a new secret. Select **Other type of secret**. Configure a **Key/Value** pair to store the secret passphrase itself.

Graphical user interface, text, application, email

Description automatically generated

Figure 2: Assign a **Secret name** and Description. The **Secret name** will be used to retrieve the secret programmatically.

Graphical user interface, application

Description automatically generated

Figure 3: Assign a **AccessProject** Tag for this secret. This is used to implement additional project-specific access-control for this Secret in the IAM Resource policy.

Graphical user interface, text, application, email

Description automatically generated

Figure 4: Assign IAM Resource Policy for this Secret. (example)

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": {  "Effect": "Allow",  "Principal": {  "AWS": [ARN for IAM User(s)/Role(s) that shall be granted permission to access to this secret]  },  "Condition": {  "StringEquals": {  "aws:ResourceTag/AccessProject": "${ aws:PrincipalTag/AccessProject }"  }  },  "Action": ["secretsmanager:GetSecretValue","secretsmanager:DescribeSecret"],  "Resource": "\*"  }  } |

Figure 5: Sample IAM Resource Policy to be assigned to this Secret.

1. Goto AWS IAM console

<https://us-east-1.console.aws.amazon.com/iamv2/home?region=ap-southeast-1#/home>

Graphical user interface, application

Description automatically generated

Figure 6: Ensure you assign the identical value for the **AccessProject** tag to the target IAM User/Role that was granted the permission to access the Secret in the IAM Resource Policy that was configured in the preceding step.

1. Setup **AWS Toolkit for Visual Studio** on your Development Machine. With this, you can securely store your AWS IAM user credentials on your development machine and need not hardcode the sensitive credentials into your source code or config file.

References:

* <https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/credentials.html>
* <https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/keys-profiles-credentials.html>
* <https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/basic-use.html>

A screenshot of a computer

Description automatically generated

Figure 7: Configuring AWS IAM Account Profile using AWS Toolkit for Visual Studio.NET

Text

Description automatically generated

Figure 8: Select the AWS IAM profile to use on your Development machine

|  |
| --- |
| **Important – setup for Production Environment:**  Use one of the following options for supplying AWS Credentials in an **AWS Production Environment**.   * Amazon EC2 Instance Metadata Service ([IAM role attached to an instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html))   The SDK attempts to fetch credentials from the [Instance Metadata Service](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html).   * Amazon ECS container credentials ([IAM roles for task](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-iam-roles.html))   The SDK attempts to resolve AWS\_CONTAINER\_CREDENTIALS\_RELATIVE\_URI or AWS\_CONTAINER\_CREDENTIALS\_FULL\_URI environment variables to fetch credentials from.   * AWS STS web identity (including Amazon Elastic Kubernetes Service (Amazon EKS))   The SDK attempts to resolve JVM system properties and environment variables to [assume a role using a web identity](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithWebIdentity.html). |

1. Configure the appSettings in app.config file of the **PgpCombinedCrypto** project.
   * Set the value for **IsUsingAWSSecretsManager** to true
   * Set value for **AliceEmailAddress.** Enter the email of the sender’s OpenPGP keypair.
   * Set value for **BobEmailAddress** Enter the email of the recipient’s OpenPGP keypair.

Note: The other appSettings parameters that aren’t mentioned above are irrelevant for this option.

**Tip:** You can still change the values in the app.config after building the application by accessing the **PgpCombinedCrypto.dll.config** file in the application folder.

Setup is complete!

# Setup Steps to protect Secret Passphrase using Windows Data Protection API

**Encrypting the Secret Passphrase.**

1. Configure the appSettings in app.config file of the **EncryptStringWithWindowsDataProtectionAPI** project
   * Set the value for **entropy.** Enter a unique value for your application. This acts as a unique-identifier (ie.secondary entropy) for Windows Data Protection API (DPAPI) to perform encryption/decryption operations specifically for your application.

Text

Description automatically generated

**Tip:** You can still change the values in the app.config after building the application by accessing the **EncryptStringWithWindowsDataProtectionAPI.dll.config** file in the application folder.

1. Build the **EncryptStringWithWindowsDataProtectionAPI** project (F6). You should be able to build this project successfully without any errors. This project provides a console program for encrypting your secret passphrase.
2. Run the **EncryptStringWithWindowsDataProtectionAPI** Console Program at the following relative path: \EncryptStringWithDPAPI\bin\Debug\net6.0\EncryptStringWithWindowsDataProtectionAPI.exe

Or Run the project from Visual Studio.

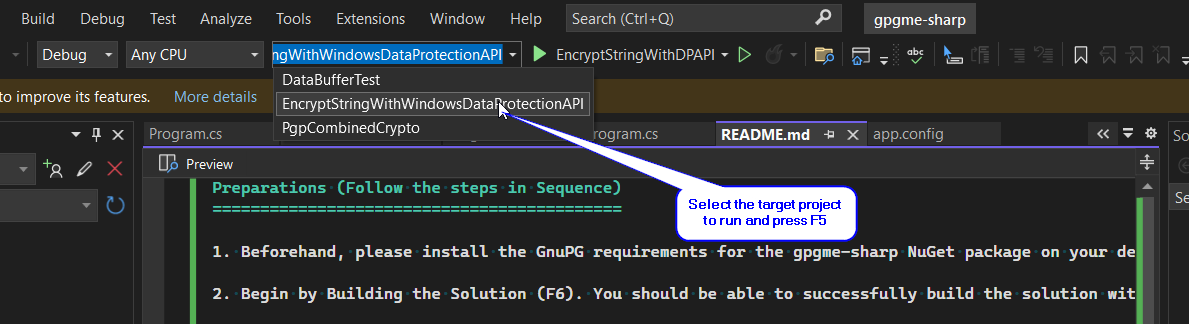


Figure 9: Run **EncryptStringWithWindowsDataProtectionAPI** project from Visual Studio

1. Use the **EncryptStringWithWindowsDataProtectionAPI** Console Program to encrypt the secret Passphrases of the OpenPGP Private Keys.

Graphical user interface, text, application, Teams

Description automatically generated

Figure 10: Encrypt the Secret Passphrase using the **EncryptStringWithWindowsDataProtectionAPI** Console Program. Copy the encrypted passphrase into the app.config.

|  |
| --- |
| **Important – setup for Production Environment:**  The Windows Data Protection API (DPAPI) is focused on providing data protection for each windows user accounts. This means that the encryption and decryption operations must be done using the same windows account.  <https://learn.microsoft.com/en-us/previous-versions/ms995355(v=msdn.10)?redirectedfrom=MSDN>  Hence, it is imperative to run the **EncryptStringWithWindowsDataProtectionAPI** Console Program (to encrypt the secret passphrases) on each production server, using the same service-account that will be used to run the program (that implements gnupg) on the same production server that performs the decryption operation on those encrypted passphrases.  Copy the **EncryptStringWithWindowsDataProtectionAPI** Console Program (ie. the entire application folder) and run it on the production server.    Figure 11: Open File Explorer and browse to the executable file you wish to run as different user. Simply hold down the Shift key and right-click on the executable file, select Run as different user from the context menu. |

1. Configure the appSettings in app.config file of the **PgpCombinedCrypto** project.
   * Set the value for **IsUsingAWSSecretsManager** to false
   * Set the value for **entropy.** Enter a unique value for your application. This acts as a unique-identifier (ie.secondary entropy) for Windows Data Protection API (DPAPI) to perform encryption/decryption operations specifically for your application.
   * Set the value for **AliceEmailAddress.** Enter the email of the sender’s OpenPGP keypair.
   * Set the value for **AliceEncryptedSecretPassPhrase.** Enter the encryptedpassphrase generated in the preceding step.
   * Set the value for **BobEmailAddress.** Enter the email of the recipient’s OpenPGP keypair.
   * Set the value for **BobEncryptedSecretPassPhrase.** Enter the encrypted passphrase generated in the preceding step.

Note: The other appSettings parameters that aren’t mentioned above are irrelevant for this option.

**Tip:** You can still change the values in the app.config after building the application by accessing the **PgpCombinedCrypto.dll.config** file in the application folder.

Setup is complete!