Alibaba Cloud KeyManagementService

Best Practices

Issue: 20190902

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Use envelope encryption to encrypt and decrypt local data

You must encrypt sensitive information in your IT assets that are deployed on Alibaba Cloud. Envelope encryption allows you to use data keys generated by Key Management Service (KMS) to encrypt large amounts of local data. You can call the corresponding cryptographic operations of Key Management Service (KMS) to generate a data key pair online and then use the data key pair to encrypt and decrypt local data. This encryption mechanism is known as envelope encryption.

Scenarios

You can use envelope encryption in many scenarios, including but not limited to the following:

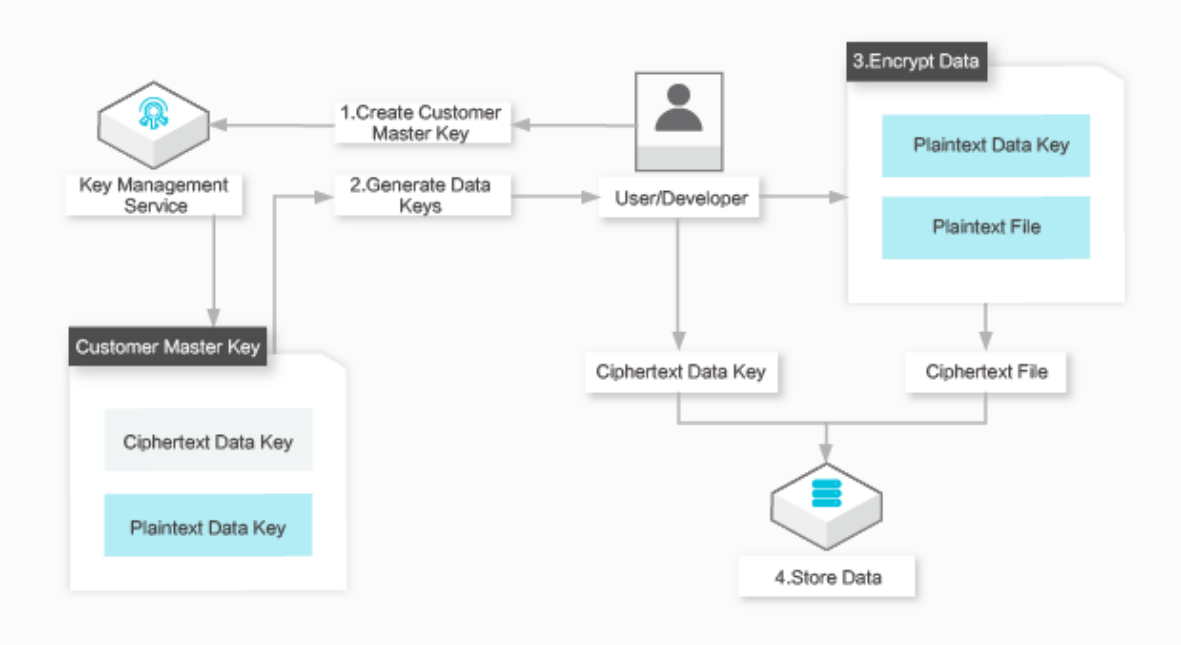
- Encrypt business data files.
- Encrypt all data stored on local disks.

This topic describes how to use envelope encryption to encrypt and decrypt local files.

How envelope encryption works

Use KMS to create a customer master key (CMK), use the CMK to generate a data key pair, and then use the plaintext data key to encrypt local files. Envelope encryption is suitable for encrypting large amounts of data. The following figure shows the entire envelope encryption procedure.

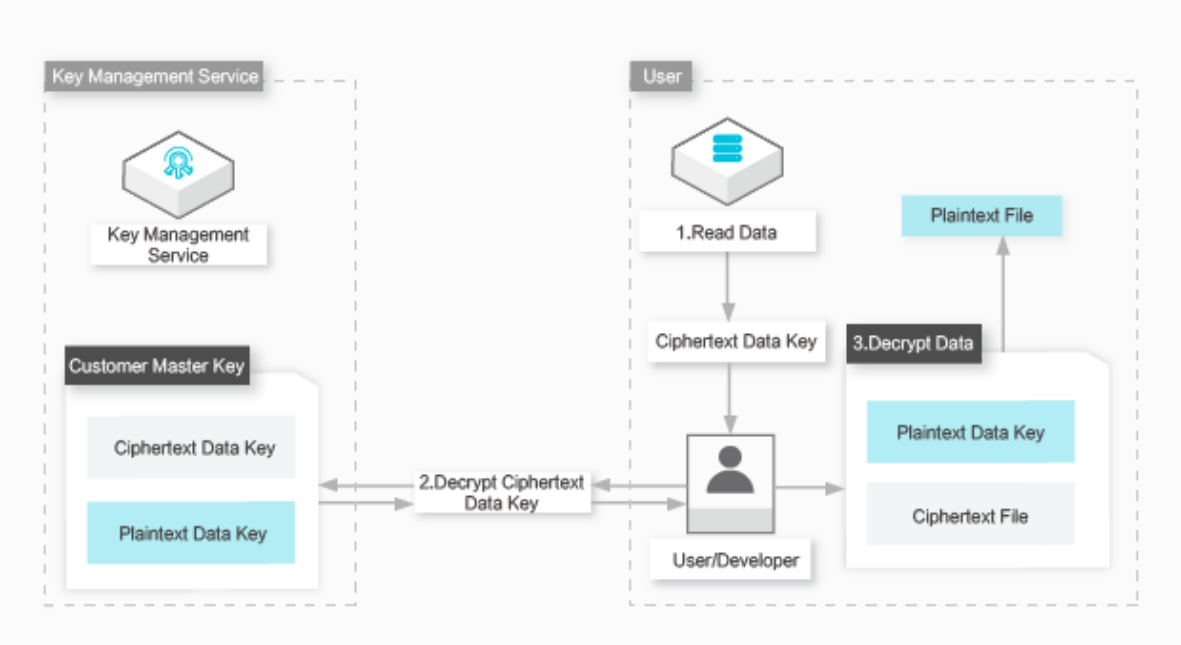
- Envelope encryption



Procedure:

1. Use the KMS console or call the CreateKey operation to create a CMK.
2. Call the GenerateDataKey operation of KMS to generate a data key pair. KMS returns a data key pair: a plaintext data key and a ciphertext data key.
3. Use the plaintext data key to encrypt the local files, and then clear the plaintext data key stored in Random Access Memory (RAM).
4. Store the ciphertext data key and encrypted data files on a storage device or service.

- Envelope decryption



Procedure:

1. Retrieve the ciphertext data key from the local device or service.
2. Call the Decrypt operation of KMS to decrypt the ciphertext data key. A plaintext copy of the data key is returned.
3. Use the plaintext data key to decrypt the local files, and then clear the plaintext data key stored in RAM.

Related API operations

You can call the following KMS API operations to encrypt and decrypt local data.

Operation	Description
#unique_4	Creates a CMK.
#unique_5	Assigns an alias to a CMK.
#unique_6	Generates a data key, uses the specified CMK to encrypt the data key, and then returns a plaintext data key and a ciphertext data key.
#unique_7	Decrypts data that is encrypted by KMS, including the ciphertext data key generated by calling the GenerateDataKey operation. You do not need to specify a CMK.

Encrypt and decrypt a local file

- **Envelope encryption**

1. **Create a CMK.**

```
$ aliyun kms CreateKey
{
  " KeyMetadat a ": {
    " CreationDa te ": " 2019 - 04 - 08T07 : 45 : 54Z ",
    " Descriptio n ": "",
    " KeyId ": " 1234abcd - 12ab - 34cd - 56ef - 12345678 ****",
    " KeyState ": " Enabled ",
    " KeyUsage ": " ENCRYPT / DECRYPT ",
    " DeleteDate ": "",
    " Creator ": " 1111222233 33 ",
    " Arn ": " acs : kms : cn - hangzhou : 1111222233 33 : key /
 1234abcd - 12ab - 34cd - 56ef - 12345678 ****",
    " Origin ": " Aliyun_KMS ",
    " MaterialEx pireTime ": ""
  },
  " RequestId ": " 2a37b168 - 9fa0 - 4d71 - aba4 - 2077dd9e80 df
 "
}
```

2. **Assign an alias to the CMK.**

   Aliases are optional to CMKs. If a CMK does not have an alias, you can use its ID.

```
$ aliyun kms CreateAlia s -- AliasName alias / Apollo /
 WorkKey -- KeyId 1234abcd - 12ab - 34cd - 56ef - 1234567890
 ab
```

> **Note:**
> **In this example, Apollo/WorkKey specifies the CMK in the Apollo project that is used to encrypt data keys. The alias of the CMK is WorkKey. This means that you can specify alias/Apollo/WorkKey to use the CMK WorkKey to encrypt a data key.**

3. **Encrypt a local data file.**

   Sample code:

   - **CMK: The alias of the CMK is** `alias / Apollo / WorkKey` **.**
   - **Plaintext data file:** *`./ data / sales . csv`*
   - **Ciphertext data file:** *`./ data / sales . csv . cipher`*

```
#! / usr / bin / env python
# coding = utf - 8

 import json
 import base64
```

```
from   Crypto . Cipher   import   AES

from   aliyunsdkc  ore   import   client
from   aliyunsdkk  ms . request . v20160120   import
GenerateDa  taKeyReque  st

def   KmsGenerat  eDataKey ( client ,  key_alias ):
    request  =  GenerateDa  taKeyReque  st . GenerateDa
taKeyReque  st ()
    request . set_accept  _format (' JSON ')
    request . set_KeyId ( key_alias )
    request . set_Number  OfBytes ( 32 )
    response  =  json . loads ( client . do_action ( request ))

    datakey_en  crypted  =  response [" Ciphertext  Blob "]
    datakey_pl  aintext  =  response [" Plaintext "]
    return  ( datakey_pl  aintext ,  datakey_en  crypted )

def   ReadTextFi  le ( in_file ):
  file  =  open ( in_file , ' r ')
  content  =  file . read ()
  file . close ()
  return   content

def   WriteTextF  ile ( out_file ,  lines ):
  file  =  open ( out_file , ' w ')
  for   ln   in   lines :
    file . write ( ln )
    file . write ('\ n ')
  file . close ()

#  Out   file   format  ( text )
#  Line   1 :  b64   encoded   data   key
#  Line   2 :  b64   encoded   IV
#  Line   3 :  b64   encoded   ciphertext
#  Line   4 :  b64   encoded   authentica  tion   tag
def   LocalEncry  pt ( datakey_pl  aintext ,  datakey_en  crypted
,  in_file ,  out_file ):
  data_key_b  inary  =  base64 . b64decode ( datakey_pl  aintext
)
  cipher  =  AES . new ( data_key_b  inary ,  AES . MODE_EAX )

  in_content  =  ReadTextFi  le ( in_file )
  ciphertext ,  tag  =  cipher . encrypt_an  d_digest (
in_content )

  lines  = [ datakey_en  crypted ,  base64 . b64encode ( cipher .
nonce ),  base64 . b64encode ( ciphertext ),  base64 . b64encode
( tag )];
  WriteTextF  ile ( out_file ,  lines )

clt  =  client . AcsClient (' Access - Key - Id ',' Access - Key
- Secret ',' Region - Id ')

key_alias  = ' alias / Apollo / WorkKey '

in_file  = './ data / sales . csv '
out_file  = './ data / sales . csv . cipher '

#  Generate   Data   Key
datakey  =  KmsGenerat  eDataKey ( clt ,  key_alias )

#  Locally   Encrypt   the   sales   record
```

```
  LocalEncry  pt ( datakey [ 0 ],  datakey [ 1 ],  in_file ,
  out_file )
```

- **Envelope decryption**

  Decrypt a local file.

  Sample code:

  - **Ciphertext data file:** *./ data / sales . csv . cipher*
  - **Plaintext data file:** *./ data / decrypted_ sales . csv*

```
#! / usr / bin / env   python
# coding = utf - 8

 import   json
 import   base64

 from   Crypto . Cipher   import   AES

 from   aliyunsdkc  ore   import   client
 from   aliyunsdkk  ms . request . v20160120   import   DecryptReq
 uest

 def   KmsDecrypt ( client ,  ciphertext ):
   request  =  DecryptReq  uest . DecryptReq  uest ()
   request . set_accept  _format (' JSON ')
   request . set_Cipher  textBlob ( ciphertext )
   response  =  json . loads ( clt . do_action ( request ))
   return   response . get (" Plaintext ")

 def   ReadTextFi  le ( in_file ):
   file  =  open ( in_file , ' r ')
   lines  = []
   for   ln   in   file :
     lines . append ( ln )
   file . close ()
   return   lines

 def   WriteTextF  ile ( out_file ,  content ):
   file  =  open ( out_file , ' w ')
   file . write ( content )
   file . close ()

 def   LocalDecry  pt ( datakey ,  iv ,  ciphertext ,  tag ,
 out_file ):
   cipher  =  AES . new ( datakey ,  AES . MODE_EAX ,  iv )
   data  =  cipher . decrypt_an  d_verify ( ciphertext ,  tag ).
 decode (' utf - 8 ')
   WriteTextF  ile ( out_file ,  data )

 clt  =  client . AcsClient (' Access - Key - Id ',' Access - Key -
 Secret ',' Region - Id ')

 in_file  = './ data / sales . csv . cipher '
 out_file  = './ data / decrypted_  sales . csv '

#  Read   encrypted   file
 in_lines  =  ReadTextFi  le ( in_file )

#  Decrypt   data   key
```

```
 datakey  =  KmsDecrypt ( clt ,  in_lines [ 0 ])

#  Locally   decrypt   the   sales   record
 LocalDecry  pt (
   base64 . b64decode ( datakey ),
   base64 . b64decode ( in_lines [ 1 ]), #  IV
   base64 . b64decode ( in_lines [ 2 ]), #  Ciphertext
   base64 . b64decode ( in_lines [ 3 ]), #  Authentica  tion   tag
   out_file
  )
```

2 Use CMK encryption to encrypt and decrypt data online

You must encrypt sensitive information in your IT assets that are deployed on Alibaba Cloud. You can call cryptographic operations of Key Management Service (KMS) to encrypt or decrypt data less than 6 KB online.

Scenarios

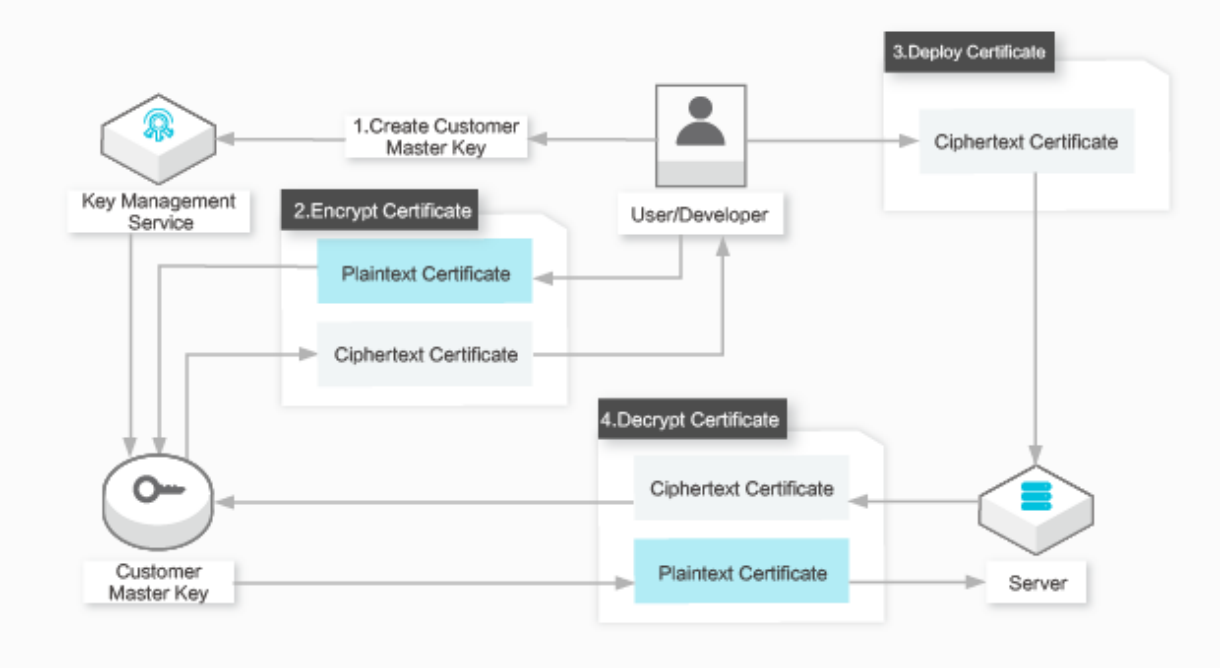
You can use CMK encryption in many scenarios, including but not limited to the following:

- Encrypt configuration files.
- Encrypt private keys of SSL certificates.

This topic describes how to call the KMS API to encrypt and decrypt private keys of SSL certificates online.

How CMK encryption works

User data is transmitted to the KMS server through an encrypted connection. The KMS server encrypts or decrypts the data, and then returns the data to the user through the encrypted connection. The following figure shows the entire procedure.



Procedure:

1. Use the KMS console or call the CreateKey operation to create a customer master key (CMK). For more information, see Create a CMK.
2. Call the Encrypt operation of KMS to encrypt the private key of an SSL certificate. A ciphertext copy of the private key is returned. For more information, see Encrypt a private key.
3. Install the SSL certificate and ciphertext private key on your cloud server.
4. When the cloud server needs to create an encrypted connection, it calls the Decrypt operation of KMS to decrypt the ciphertext private key. For more information, see Decrypt a private key.

Related API operations

You can call the following API operations to encrypt and decrypt data.

Operation	Description
#unique_4	Creates a CMK.
#unique_5	Assigns an alias to a CMK.
#unique_9	Encrypts data with a specified CMK.
#unique_7	Decrypts data that is encrypted by KMS. You do not need to specify a CMK.

Encrypt and decrypt the private key of an SSL certificate

1. Call the CreateKey operation to create a CMK.

```
$  aliyun   kms   CreateKey
{
  " KeyMetadat  a ": {
    " CreationDa  te ": " 2019 - 04 - 08T07 : 45 : 54Z ",
    " Descriptio  n ": "",
    " KeyId ": " 1234abcd - 12ab - 34cd - 56ef - 12345678 ****",
    " KeyState ": " Enabled ",
    " KeyUsage ": " ENCRYPT / DECRYPT ",
    " DeleteDate ": "",
    " Creator ": " 1111222233  33 ",
    " Arn ": " acs : kms : cn - hangzhou : 1111222233  33 : key /
 1234abcd - 12ab - 34cd - 56ef - 12345678 ****",
    " Origin ": " Aliyun_KMS ",
    " MaterialEx  pireTime ": ""
  },
  " RequestId ": " 2a37b168 - 9fa0 - 4d71 - aba4 - 2077dd9e80  df "
```

```
}
```

2. Assign an alias to the CMK.

   Aliases are optional to CMKs. If a CMK does not have an alias, you can use its ID.

```
$ aliyun kms CreateAlia s -- AliasName alias / Apollo /
 WorkKey -- KeyId 1234abcd - 12ab - 34cd - 56ef - 12345678 ****
```

Note:

In this example, `Apollo / WorkKey` specifies the CMK in the Apollo project that is used to encrypt the private key. The alias of the CMK is WorkKey. This means that you can specify `alias / Apollo / WorkKey` to use the CMK WorkKey to encrypt a private key.

3. Call the Encrypt operation to encrypt the private key. KMS then encrypts the private key.

   Sample code:

   - CMK: The alias of the CMK is `alias / Apollo / WorkKey` .
   - Plaintext private key: *./ certs / key . pem*
   - Ciphertext private key: *./ certs / key . pem . cipher*

```
#! / usr / bin / env python
# coding = utf - 8

import json

from aliyunsdkc ore import client
from aliyunsdkk ms . request . v20160120 import EncryptReq
uest
from aliyunsdkk ms . request . v20160120 import DecryptReq
uest

def KmsEncrypt ( client , plaintext , key_alias ):
  request = EncryptReq uest . EncryptReq uest ()
  request . set_accept _format (' JSON ')
  request . set_KeyId ( key_alias )
  request . set_Plaint ext ( plaintext )
  response = json . loads ( clt . do_action ( request ))
  return response . get (" Ciphertext Blob ")

def ReadTextFi le ( in_file ):
  file = open ( in_file , ' r ')
  content = file . read ()
  file . close ()
  return content

def WriteTextF ile ( out_file , content ):
  file = open ( out_file , ' w ')
  file . write ( content )
  file . close ()
```

```
clt = client . AcsClient ('< Access - Key - Id >',' Access - Key
- Secret ','< Region - Id >')

key_alias = ' alias / Apollo / WorkKey '

in_file = './ certs / key . pem '
out_file = './ certs / key . pem . cipher '

# Read private key file in text mode
in_content = ReadTextFi le ( in_file )

# Encrypt
ciphertext = KmsEncrypt ( clt , in_content , key_alias )

# Write encrypted key file in text mode
WriteTextF ile ( out_file , ciphertext )
```

4. **Call the Decrypt operation to decrypt the ciphertext private key. KMS then decrypts the private key that you have installed on your cloud server.**

   Sample code:

   - **Ciphertext private key:** *./ certs / key . pem . cipher*
   - **Plaintext private key:** *./ certs / decrypted_ key . pem*

```
#! / usr / bin / env python
# coding = utf - 8

import json

from aliyunsdkc ore import client
from aliyunsdkk ms . request . v20160120 import EncryptReq
uest
from aliyunsdkk ms . request . v20160120 import DecryptReq
uest

def KmsDecrypt ( client , ciphertext ):
  request = DecryptReq uest . DecryptReq uest ()
  request . set_accept _format (' JSON ')
  request . set_Cipher textBlob ( ciphertext )
  response = json . loads ( clt . do_action ( request ))
  return response . get (" Plaintext ")

def ReadTextFi le ( in_file ):
  file = open ( in_file , ' r ')
  content = file . read ()
  file . close ()
  return content

def WriteTextF ile ( out_file , content ):
  file = open ( out_file , ' w ')
  file . write ( content )
  file . close ()

clt = client . AcsClient ('< Access - Key - Id >',' Access - Key
- Secret ','< Region - Id >')

in_file = './ certs / key . pem . cipher '
out_file = './ certs / decrypted_ key . pem '

# Read encrypted key file in text mode
```

```
 in_content  =  ReadTextFi  le ( in_file )

#  Decrypt
 ciphertext  =  KmsDecrypt ( clt ,  in_content )

#  Write   Decrypted   key   file   in   text   mode
 WriteTextF  ile ( out_file ,  ciphertext )
```