Data Encryption Workshop

FAQs

Issue 06

Date 2025-12-16

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

Contents

1 About DEW.......1
1.1 What Cryptography Algorithms Does DEW Use?.......1
1.2 What Is the Resource Allocation Mechanism of DEW?.......3
1.3 What Are Regions and AZs?.......3
1.4 How Do I Access the Functions of DEW?.......4
1.5 Why Do DEW Permissions Fail to Take Effect Immediately?.......4
2 KMS Related.......6
2.1 Which Cloud Services Are Provided with Default Keys by KMS?.......6
2.2 What Are the Differences Between a Custom Key and a Default Key?.......7
2.3 How Do I Use a DEK?.......7
2.4 Why Cannot I Delete a CMK Immediately?.......8
2.5 Which Cloud Services Can Use KMS for Encryption?.......8
2.6 How Do Huawei Cloud Services Use KMS to Encrypt Data?.......11
2.7 What Are the Benefits of Envelope Encryption?.......12
2.8 Is There a Limit on the Number of Custom Keys That I Can Create on KMS?.......13
2.9 Can I Export a CMK from KMS?.......13
2.10 Can I Decrypt My Data if I Permanently Delete My Custom Key?.......13
2.11 Can I Update CMKs Created by KMS-Generated Key Materials?.......14
2.12 How Are Default Keys Generated?.......14
2.13 What Should I Do If I Do Not Have the Permissions to Perform Operations on KMS?.......15
2.14 Why Can't I Wrap Asymmetric Keys by Using -id-aes256-wrap-pad in OpenSSL?.......15
2.15 Key Algorithms Supported by KMS.......17
2.16 What Should I Do If KMS Failed to Be Requested and Error Code 401 Is Displayed?.......18
2.17 What Is the Relationship Between the Ciphertext and Plaintext Returned by the encrypt-data API?.......18
2.18 How Does KMS Protect My Keys?.......19
2.19 How Do I Use an Asymmetric Key to Verify the Signature Result of a Public Key Pair?.......19
2.20 Does an Imported Key Support Rotation?.......21
2.21 Does KMS Support Offline Data Encryption and Decryption?.......21
2.22 How Do I Convert an Original EC Private Key into a Private Key in PKCS8 Format?.......23
3 CSMS Related.......26
3.1 Why Cannot I Delete the Version Status of a Secret?.......26

4 KPS Related....27

4.1 How Do I Handle an Import Failure of a Key Pair Created Using PuTTYgen?....27

4.2 What Should I Do When I Fail to Import a Key Pair Using Internet Explorer 9?....30

4.3 How Do I Handle the Failure in Binding a Key Pair?....30

4.4 How Do I Handle the Failure in Replacing a Key Pair?....34

4.5 How Do I Handle the Failure in Resetting a Key Pair?....35

4.6 How Do I Handle the Failure in Unbinding a Key Pair?....36

4.7 Do I Need to Restart Servers After Replacing Its Key Pair?....37

4.8 How Do I Enable the Password Login Mode for an ECS?....37

4.9 How Do I Handle the Failure in Logging In to ECS After Unbinding the Key Pair?....39

4.10 What Should I Do If My Private Key Is Lost?....41

4.11 How Do I Convert the Format of a Private Key File?....41

4.12 Can I Change the Key Pair of a Server?....43

4.13 Can a Key Pair Be Shared by Multiple Users?....43

4.14 How Do I Obtain the Public or Private Key File of a Key Pair?....43

4.15 What Can I Do If an Error Is Reported When an Account Key Is Created or Upgraded for the First Time?....44

4.16 Will the Account Key Pair Quota Be Occupied After a Private Key Pair Is Upgraded to an Account Key Pair?....44

5 Dedicated HSM Related....45

5.1 How Does Dedicated HSM Ensure the Security for Key Generation?....45

5.2 Does an Equipment Room Administrator Have the Super Management Permission to Insert Privileged Ukeys to Steal Information?....45

5.3 What HSMs Are Used for Dedicated HSM?....45

5.4 What APIs Does Dedicated HSM Support?....46

1 About DEW

1.1 What Cryptography Algorithms Does DEW Use?

Key Algorithms and Specifications Supported by KMS

Table 1-1 Key algorithms supported by KMS

Key Type	Algorithm Type	Key Specifications	Description	Scenario
Symmetric key	AES	AES_256 (AES-256-GCM authentication encryption)	AES symmetric key	• Data encryption and decryption • DEK encryption and decryption **NOTE** You can encrypt and decrypt a small amount of data using the online tool on the console. You need to call APIs to encrypt and decrypt a large amount of data.
Digest key	SHA	• HMAC_256 • HMAC_384 • HMAC_512	Digest key	• Data tampering prevention • Data integrity verification

Key Type	Algorithm Type	Key Specifications	Description	Scenario
Asymmetric key	RSA	● RSA_2048 ● RSA_3072 ● RSA_4096	RSA asymmetric key	● Digital signature and signature verification ● Data encryption and decryption **NOTE** Asymmetric keys are applicable to signature and signature verification scenarios. Asymmetric keys are not efficient enough for data encryption. Symmetric keys are suitable for encrypting and decrypting data.
Asymmetric key	ECC	● EC_P256 ● EC_P384	Elliptic curve recommended by NIST	Digital signature and signature verification

Table 1-2 describes the encryption and decryption algorithms supported for user-imported keys.

Table 1-2 Key wrapping algorithms

Algorithm	Description	Configuration
RSAES_OAEP_SHA_256	RSA algorithm that uses OAEP and has the **SHA-256** hash function	Select an algorithm based on your HSM functions. If the HSMs support the **RSAES_OAEP_SHA_256** algorithm, use **RSAES_OAEP_SHA_256** to encrypt key materials. **NOTICE** The **RSAES_OAEP_SHA_1** algorithm is no longer secure. Exercise caution when performing this operation.
RSAES_OAEP_SHA_1	RSA algorithm that uses Optimal Asymmetric Encryption Padding (OAEP) and has the **SHA-1** hash function	

Cryptographic Algorithms Supported by KPS

- The SSH key pairs created on the management console support the following cryptographic algorithms:
  - SSH-ED25519
  - ECDSA-SHA2-NISTP256

  - ECDSA-SHA2-NISTP384
  - ECDSA-SHA2-NISTP521
  - SSH_RSA: The length can be 2,048, 3,072, or 4,096 bits.
- The SSH keys imported to the KPS console support the following cryptographic algorithms:
  - SSH-DSS (not recommended)
  - SSH-ED25519
  - ECDSA-SHA2-NISTP256
  - ECDSA-SHA2-NISTP384
  - ECDSA-SHA2-NISTP521
  - SSH_RSA: The length can be 2,048, 3,072, or 4,096 bits.

Supported Cryptography Algorithms

You can use international common cryptographic algorithms to meet various user requirements.

Table 1-3 Supported cryptography algorithms

Category	Common Cryptographic Algorithm
Symmetric cryptographic algorithm	AES
Asymmetric cryptographic algorithm	RSA, DSA, ECDSA, DH, and ECDH
Digest algorithm	SHA1, SHA256, and SHA384

1.2 What Is the Resource Allocation Mechanism of DEW?

DEW uses regions as large resource pools and independent resources or services of each customer as small resource pools. The background has default traffic limits. For a single user, if the traffic exceeds the threshold, the service speed is slow. For customers who have heavy traffic requirements, background resources can be changed based on the actual situation and requirements.

If your traffic volume exceeds the limit, you can submit a service ticket to increase the quota. DEW will adjust your limit in the background to support your provisioning of dedicated configuration clusters and ensure stable service running.

1.3 What Are Regions and AZs?

Concepts

A region or an availability zone (AZ) identifies the location of a data center. You can create resources in a specific region or an AZ.

- Regions are divided from the dimensions of geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are shared within the same region. Regions are classified as universal regions and dedicated regions. A universal region provides universal cloud services for common tenants. A dedicated region provides services of the same type only or for specific tenants.
- An AZ contains one or more physical data centers. Each AZ has independent cooling, fire extinguishing, moisture-proof, and electricity facilities. Within an AZ, computing, network, storage, and other resources are logically divided into multiple clusters. AZs within a region are interconnected using high-speed optical fibers to allow you to build cross-AZ high-availability systems.

Selecting a Region

If you or your users are in Europe, select the **EU-Dublin** region.

Selecting an AZ

When determining whether to deploy resources in the same AZ, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs in the same region.
- For low network latency, deploy resources in the same AZ.

1.4 How Do I Access the Functions of DEW?

You can use DEW on the web console or call the functions of DEW by using HTTPS-based APIs.

- Console

  If you have registered with the public cloud, you can log in to the management console directly. Click ☰ on the left and choose **Security** > **Data Encryption Workshop**.
- API

  You can access DEW using APIs. For details, see *Data Encryption Workshop API Reference*.

  DEW supports representational state transfer (REST) APIs, allowing you to call APIs using HTTPS. You can use provided APIs to perform operations on keys and key pairs, such as creating, querying, and deleting keys.

  DEW APIs use the HTTPS protocol to encrypt and secure transmission, preventing man-in-the-middle attacks.

1.5 Why Do DEW Permissions Fail to Take Effect Immediately?

Generally, if you have configured DEW-related permissions, such as **KMS Administrator** and **KMS CMKFullAccess**, the permissions take effect immediately.

However, permissions cannot take effect in time in the following scenarios:

1. You did not log out of the console in time and the session is cached. Log in to the console again.
2. You used services that have permissions to cache data, such as OBS. In this case, when the permission takes effect depends on the service cache duration.
3. APIG is called. In this case, when the permission takes effect depends on the time when IAM broadcasts the permission change to the gateway.

2 KMS Related

2.1 Which Cloud Services Are Provided with Default Keys by KMS?

A default key is automatically created by another cloud service using KMS, such as Object Storage Service (OBS). The alias of a default key ends with **/default**.

You can use the management console to query but cannot disable or schedule the deletion of default keys.

Default keys are hosted for free, and are charged based on the number of the API requests for them. If API requests exceed the free limit, the excess part will be charged.

Table 2-1 Default master keys

Alias	Cloud Service
obs/default	Object Storage Service (OBS)
evs/default	Elastic Volume Service (EVS)
ims/default	Image Management Service (IMS)
vbs/default	Volume Backup Service (VBS)
sfs/default	Scalable File Service (SFS)
kps/default	Key Pair Service (KPS)
csms/default	Cloud Secret Management Service (CSMS)
dlf/default	DataArts Studio
dds/default	Document Database Service (DDS)
elb/default	Elastic Load Balance (ELB)
coc/default	Cloud Operations Center (COC)

Alias	Cloud Service
cce/default	Cloud Container Engine (CCE)

NOTE

A default key is automatically created when a user employs the KMS encryption function for the first time in another cloud service.

2.2 What Are the Differences Between a Custom Key and a Default Key?

The following table describes the differences between a custom key and a default key.

Table 2-2 Differences between a custom key and a default key

Item	Definition	Difference
Custom key	A Key Encryption Key (KEK) created using KMS. The key is used to encrypt and protect DEKs. A custom key can be used to encrypt multiple DEKs.	● It can be disabled and scheduled for deletion. ● It is billed per use after the being created or imported.
Default key	Automatically generated by the system when you use KMS to encrypt data in another cloud service for the first time. The suffix of the key is **/ default**. Example: **evs/default**	● It cannot be disabled or scheduled for deletion. ● You are not charged when you use the cloud service automatically generated by the system. If the number of API requests exceeds 20,000, you will be billed.

2.3 How Do I Use a DEK?

A data encryption key (DEK) is used to encrypt data.

Using KMS, you can create, encrypt, and decrypt DEKs. The KMS system does not save, manage, or track your DEKs, neither does it use the DEKs to encrypt or decrypt data.

Creating a DEK

KMS supports the creation, encryption, and decryption of DEKs only by calling APIs. You can create a DEK in either of the following ways:

- If you call the **create-datakey** API, it returns the plaintext DEK and the ciphertext DEK encrypted using the specified CMK.
- If you call the **create-datakey-without-plaintext** API, it returns the ciphertext DEK encrypted using the specified CMK. You can call the **decrypt-datakey** API to decrypt the ciphertext DEK, if you need the plaintext DEK afterwards.

Encrypting Data with a DEK

KMS does not support data encryption with DEKs. You can use other encryption libraries (for example, OpenSSL) to encrypt data with DEKs.

1. Obtain a plaintext DEK by referring to **Creating a DEK**.
2. Use the plaintext DEK to encrypt data.
3. Delete the plaintext DEK and safely store the ciphertext DEK with the encrypted.

Decrypting Data with a DEK

KMS does not support data decryption with DEKs. You can use other encryption libraries (for example, OpenSSL) to decrypt data with DEKs.

1. Make sure the encrypted data and the ciphertext DEK are available and ready.
2. Call the **decrypt-datakey** API to decrypt the ciphertext DEK, and then it returns the plaintext DEK that you have used to encrypt the data.
3. Use the plaintext DEK to decrypt the data.
4. Delete the plaintext DEK.

2.4 Why Cannot I Delete a CMK Immediately?

The decision to delete a CMK should be considered with great caution. Before deletion, confirm that the CMK's encrypted data has all been migrated. As soon as the CMK is deleted, you will not be able to decrypt data with it. Therefore, KMS offers a user-specified period of 7 to 1096 days for the deletion to finally take effect. On the scheduled day of deletion, the CMK will be permanently deleted. However, prior to the scheduled day, you can still cancel the pending deletion. This is a means of precaution within KMS.

2.5 Which Cloud Services Can Use KMS for Encryption?

Object Storage Service (OBS), Elastic Volume Service (EVS), Image Management Service (IMS), and Relational Database Service (RDS) can use KMS for encryption.

Table 2-3 Cloud services supported by KMS

Service Name	How to Use	Reference
Object Storage Service (OBS)	You can upload objects to and download them from OBS in common mode or server-side encryption mode. When you upload objects in encryption mode, data is encrypted at the server side and then securely stored on OBS in ciphertext. When you download encrypted objects, the data in ciphertext is decrypted at the server side and then provided to you in plaintext. OBS supports the server-side encryption with KMS-managed keys (SSE-KMS). In this mode, OBS uses the keys provided by KMS for server-side encryption.	**Encrypting Data in OBS**
Elastic Volume Service (EVS)	If you enable the encryption function when creating an EVS disk, the disk will be encrypted with the DEK generated by using your CMK. Data stored in the EVS disk will be automatically encrypted.	**Encrypting Data in EVS**
Image Management Service (IMS)	When creating a private image using an external image file, you can enable the private image encryption function and select a CMK provided by KMS to encrypt the image.	**Encrypting Data in IMS**
Relational Database Service (RDS)	When purchasing a database instance, you can enable the disk encryption function of the database instance and select a CMK created on KMS to encrypt the disk of the database instance. Enabling the disk encryption function will enhance data security.	**Encrypting an RDS DB Instance**
Document Database Service (DDS)	When purchasing a DDS instance, you can enable the disk encryption function of the instance and select a CMK created on KMS to encrypt the disk of the instance. Enabling the disk encryption function will enhance data security.	**Encrypting a DDS DB Instance**

Service Name	How to Use	Reference
Elastic Cloud Server (ECS)	ECS uses image encryption or data disk encryption to encrypt ECS resources. ● When creating an ECS, if you select an encrypted image, the system disk of the created ECS automatically has encryption enabled, with its encryption mode same as the image encryption mode. For details about image encryption, see **Encrypting Data in IMS**. ● When creating an ECS, you can encrypt added data disks. For details about data disk encryption, see **Encrypting Data in IMS**.	**Encrypting Data in ECS**
Scalable File Service Turbo (SFS Turbo)	When creating an SFS Turbo file system, use the key provided by KMS to encrypt the file system for core data security.	**Creating an SFS Turbo File System**
FunctionGraph	To decrypt sensitive data, such as database passwords and API keys, during function runtime, you can use the KMS SDK to dynamically operate keys. You can host encryption and decryption keys in KMS and create an agency in IAM for FunctionGraph to access KMS.	**Asset Identification and Management**
Cloud Operations Center (COC)	COC uses KMS to encrypt your host accounts for better security. Before using KMS, create a key first.	**Key Management**
Cloud Data Migration (CDM)	When migrating files to a file system, CDM can encrypt and decrypt the files using the keys provided by KMS.	**Encryption and Decryption During File Migration**
Data Security Center (DSC)	You can use the encryption algorithms and encryption master keys to generate an encryption configuration for data masking.	**Configuring a Data Masking Rule**
Workspace	You can use the key provided by KMS to encrypt disks when purchasing a workspace.	**Purchasing Yearly/ Monthly-billed Desktops**
GeminiDB	You can use the key provided by KMS to encrypt static data in the database when purchasing a GeminiDB instance.	**Buying and Connecting to a Cluster Instance**

2.6 How Do Huawei Cloud Services Use KMS to Encrypt Data?

Generally, **Huawei Cloud services** use KMS **envelope encryption** to protect user data.

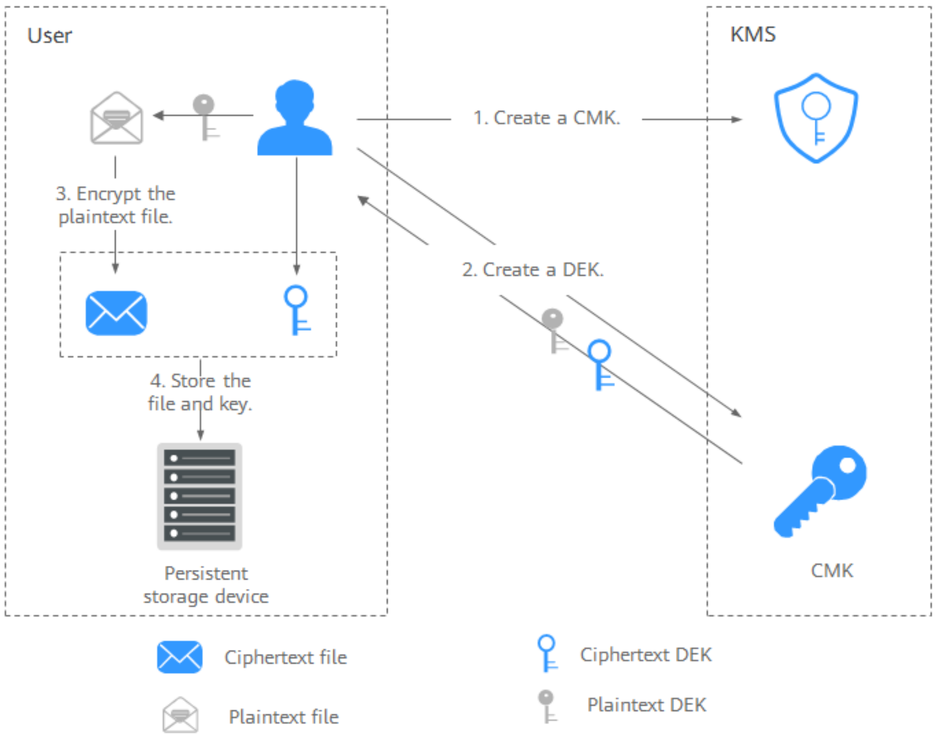
NOTE

Envelope encryption is the practice of encrypting data with a DEK and then encrypting the DEK with a root key that you can fully manage. In this case, CMKs are not required for encryption or decryption.

Envelope Encryption and Decryption Principles

- **Figure 2-1** illustrates the process for encrypting a local file.

Figure 2-1 Encrypting a local file



The procedure is as follows:

a. Create a CMK on KMS.

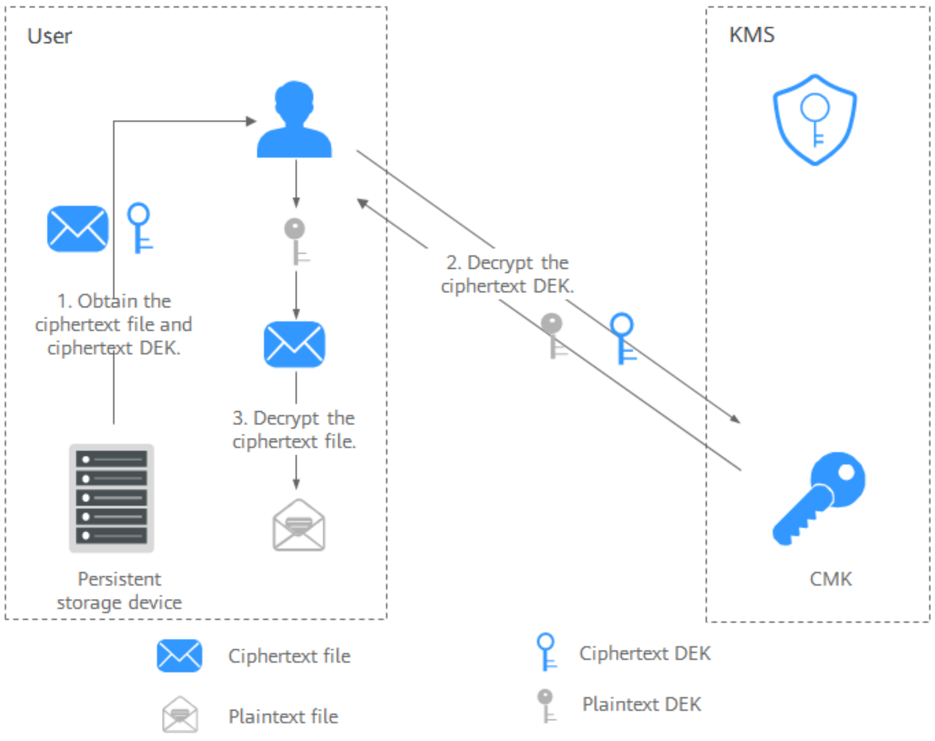
b. Call the **create-datakey** API of KMS to create a DEK. Then you get a plaintext DEK and a ciphertext DEK. The ciphertext DEK is generated when you use a CMK to encrypt the plaintext DEK.

c. Use the plaintext DEK to encrypt the file. A ciphertext file is generated.

d. Save the ciphertext DEK and the ciphertext file together in a persistent storage device or a storage service.

- **Figure 2-2** illustrates the process for decrypting a local file.

Figure 2-2 Decrypting a local file


User
KMS
1. Obtain the ciphertext file and ciphertext DEK.
2. Decrypt the ciphertext DEK.
3. Decrypt the ciphertext file.
Persistent storage device
CMK
Ciphertext file
Plaintext file
Ciphertext DEK
Plaintext DEK

The procedure is as follows:

a. Obtain the ciphertext DEK and file from the persistent storage device or the storage service.

b. Call the **decrypt-datakey** API of KMS and use the corresponding CMK (the one used for encrypting the DEK) to decrypt the ciphertext DEK. Then you get the plaintext DEK.

   If the CMK is deleted, the decryption fails. Therefore, properly keep your CMKs.

c. Use the plaintext DEK to decrypt the ciphertext file.

For details about how to use KMS to encrypt and decrypt data, see **Using KMS to Encrypt and Decrypt Data for Cloud Services**.

2.7 What Are the Benefits of Envelope Encryption?

Envelope encryption is the practice of encrypting data with a DEK and then encrypting the DEK with a root key that you can fully manage. In this case, CMKs are not required for encryption or decryption.

Benefits:

- Advantages over CMK encryption in KMS

  Users can use CMKs to encrypt and decrypt data on the KMS console or by calling KMS APIs.

  A CMK can encrypt and decrypt data no more than 4 KB. An envelope can encrypt and decrypt larger volumes of data.

  Data encrypted using envelopes does not need to be transferred. Only the DEKs need to be transferred to the KMS server.

- Advantages over encryption by using cloud services
  - Security

    Data transferred to the cloud for encryption is exposed to risks such as interception and phishing.

    During envelope encryption, KMS uses Hardware Security Modules (HSMs) to protect keys. All CMKs are protected by root keys in HSMs to avoid key leakage.
  - Trustworthiness

    You will worry about data security on the cloud. It is also difficult for cloud services to prove that they never misuse or disclose such data.

    If you choose envelope encryption, KMS will control access to keys and record all usages of and operations on keys with traceable logs, meeting your audit and regulatory compliance requirements.
  - Performance and cost

    To encrypt or decrypt data using a cloud service, you have to send the data to the encryption server and receive the processed data. This process seriously affects your service performance and incurs high costs.

    Envelope encryption allows you to generate DEKs online by calling KMS cryptographic algorithm APIs, and to encrypt a large amount of local data with the DEKs.

2.8 Is There a Limit on the Number of Custom Keys That I Can Create on KMS?

There is a limit on the number of custom keys that can be created on KMS.

You can create a maximum of 100 custom keys, including those in enabled, disabled, and pending deletion states. Default keys are not included.

2.9 Can I Export a CMK from KMS?

No.

To ensure CMK security, users can only create and use CMKs in KMS.

2.10 Can I Decrypt My Data if I Permanently Delete My Custom Key?

No.

If you have permanently deleted your custom key, the data encrypted using it cannot be decrypted. Before the scheduled deletion date of the custom key, you can cancel the scheduled deletion.

2.11 Can I Update CMKs Created by KMS-Generated Key Materials?

No.

Keys created using KMS-generated materials cannot be updated. You can only use KMS to create new CMKs to encrypt and decrypt data.

2.12 How Are Default Keys Generated?

A default key is automatically created by another cloud service using KMS, such as Object Storage Service (OBS). The alias of a default key ends with **/default**. **Table 2-4** lists the default key aliases used by cloud services through KMS.

You can use the management console to query but cannot disable or schedule the deletion of Default Master Keys.

Default keys are hosted for free, and are charged based on the number of the API requests for them. If API requests exceed the free limit, the excess part will be charged.

For example, when you upload an object on OBS, enable **Server-Side Encryption**, and set **Encryption Key Type** to **Default**, OBS will use KMS to generate a default key whose alias is **obs/default**.

Table 2-4 Default master keys

Alias	Cloud Service
obs/default	Object Storage Service (OBS)
evs/default	Elastic Volume Service (EVS)
ims/default	Image Management Service (IMS)
vbs/default	Volume Backup Service (VBS)
sfs/default	Scalable File Service (SFS)
kps/default	Key Pair Service (KPS)
csms/default	Cloud Secret Management Service (CSMS)
dlf/default	DataArts Studio
dds/default	Document Database Service (DDS)
elb/default	Elastic Load Balance (ELB)
coc/default	Cloud Operations Center (COC)
cce/default	Cloud Container Engine (CCE)

2.13 What Should I Do If I Do Not Have the Permissions to Perform Operations on KMS?

Symptom

A message indicating lack of permissions is displayed when you attempt to perform operations on keys, such as view, create, or import keys.

Possible Causes

Your account is not associated with the required KMS system policies.

Solution

Step 1 Check whether your account has been associated with **KMS Administrator** and **KMS CMKFullAccess** policies.

For details about how to check your user groups and permissions, see **User Groups and Authorization**.

If your account has been associated with required KMS system policies, go to **Step 2**.

Step 2 Associate your account with required system policies.

- For details about how to add administrator permissions, see **User Groups and Authorization**.
- For details about how to add a custom policy, see **Creating a Custom DEW Policy**.

----End

2.14 Why Can't I Wrap Asymmetric Keys by Using -id-aes256-wrap-pad in OpenSSL?

Symptom

By default, the -id-aes256-wrap-pad algorithm is not enabled in OpenSSL. To wrap a key, upgrade OpenSSL to the latest version and patch it first.

Solution

Use bash commands to create a local copy of the existing OpenSSL. You do not need to delete or modify the default OpenSSL client installation configurations.

Step 1 Switch to the **root** user.

sudo su -

Step 2 Run the following command and record the OpenSSL version:

openssl version

Step 3 Run the following commands to create the **/root/build** directory. This directory will be used to store the latest OpenSSL binary file.

mkdir $HOME/build

mkdir -p $HOME/local/ssl

cd $HOME/build

Step 4 Download the latest OpenSSL version from https://www.openssl.org/source/.

Step 5 Download and decompress the binary file.

Step 6 Replace **openssl-1.1.1d.tar.gz** with the latest OpenSSL version downloaded in **step 4**.

curl -O https://www.openssl.org/source/openssl-1.1.1d.tar.gz

tar -zxf openssl-1.1.1d.tar.gz

Step 7 Use the **gcc** tool to patch the version, and compile the downloaded binary file.

yum install patch make gcc -y

NOTE

If you are using a version other than OpenSSL-1.1.1d, you may need to change the directory and commands used, or this patch may not work properly.

Step 8 Run the following commands:

```
sed -i "/BIO_get_cipher_ctx(benc, &ctx);/a\ EVP_CIPHER_CTX_set_flags(ctx,
EVP_CIPHER_CTX_FLAG_WRAP_ALLOW);" $HOME/build/openssl-1.1.1d/apps/enc.c
```

Step 9 Run the following commands to compile the OpenSSL **enc.c** file:

cd $HOME/build/openssl-1.1.1d/

./config --prefix=$HOME/local --openssldir=$HOME/local/ssl

make -j$(grep -c ^processor /proc/cpuinfo)

make install

Step 10 Configure the environment variable **LD_LIBRARY_PATH** to ensure that required libraries are available for OpenSSL. The latest version of OpenSSL has been dynamically linked to the binary file in the **$HOME/local/ssl/lib/** directory, and cannot be directly executed in shell.

Step 11 Create a script named **openssl.sh** to load the **$HOME/local/ssl/lib/** path before running the binary file.

cd $HOME/local/bin/

echo -e '#!/bin/bash \nenv LD_LIBRARY_PATH=$HOME/local/lib/ $HOME/local/bin/openssl "$@"' > ./openssl.sh

Step 12 Run the following command to configure an execute bit on the script:

chmod 755 ./openssl.sh

Step 13 Run the following command to start the patched OpenSSL version:

$HOME/local/bin/openssl.sh

----End

2.15 Key Algorithms Supported by KMS

Table 2-5 Key algorithms supported by KMS

Key Type	Algorithm Type	Key Specifications	Description	Scenario
Symmetric key	AES	AES_256 (AES-256-GCM authentication encryption)	AES symmetric key	• Data encryption and decryption • DEK encryption and decryption **NOTE** You can encrypt and decrypt a small amount of data using the online tool on the console. You need to call APIs to encrypt and decrypt a large amount of data.
Digest key	SHA	• HMAC_256 • HMAC_384 • HMAC_512	Digest key	• Data tampering prevention • Data integrity verification
Asymmetric key	RSA	• RSA_2048 • RSA_3072 • RSA_4096	RSA asymmetric key	• Digital signature and signature verification • Data encryption and decryption **NOTE** Asymmetric keys are applicable to signature and signature verification scenarios. Asymmetric keys are not efficient enough for data encryption. Symmetric keys are suitable for encrypting and decrypting data.
Asymmetric key	ECC	• EC_P256 • EC_P384	Elliptic curve recommended by NIST	Digital signature and signature verification

2.16 What Should I Do If KMS Failed to Be Requested and Error Code 401 Is Displayed?

Symptom

An error is reported when KMS is requested or the cloud service encryption function is enabled.

Error information: **httpcode=401,code=APIGW.0301,Msg=Incorrect IAM authentication information: current ip:xx.xx.xx.xx refused**

Possible Causes

Access control is configured in IAM.

By default, IAM allows access from any IP addresses. If you configure ACL, the IP addresses and network segments out of the specified range cannot access KMS or use the cloud encryption feature.

Solution

- To access KMS through the cloud service console (for example, for OBS encryption purposes), allow access from network segments 10.0.0.0/8, 11.0.0.0/8, and 26.0.0.0/8.
- To call KMS via API, allow access from the source IP addresses.

Allowing Access from Specific IP Addresses

Step 1 Log in to the **DEW console**.

Step 2 Click ☰ on the left of the page and choose **Management & Governance** > **Identity and Access Management**. The **Users** page is displayed.

Step 3 Choose **Security Settings** and click the **ACL** tab. Check whether **IP Address Ranges** and **IPv4 CIDR Blocks** are properly configured.

NOTE

The source IP address you use must be specified on both the **Console Access** and **API Access** tabs.

----End

2.17 What Is the Relationship Between the Ciphertext and Plaintext Returned by the encrypt-data API?

The basic length of the ciphertext returned by the encrypt-data API is 124 bytes. The ciphertext consists of multiple fields, including the key ID, encryption algorithm, key version, and ciphertext digest.

The plaintext has 16 bytes in each block. A block with fewer than 16 bytes will be padded. Ciphertext length = 124 + Ceil(plaintext length/16) x 16. The conversion result is encoded using Base64.

Take 4-byte plaintext input as an example. The calculation result is 124 + Ceil(4/16) x 16 = 140. The 140 bytes are converted into 188 bytes after Base64 encoding.

NOTE

Ceil is a round-up function. Ceil(a) = 1. The value range of **a** is (0,1].

2.18 How Does KMS Protect My Keys?

The mechanism of KMS prevents anyone from accessing your keys in plaintext. KMS relies on hardware security modules (HSMs) that safeguard the confidentiality and integrity of your keys. Plaintext KMS keys are always encrypted by HSMs and are never stored on any disk. These keys are only utilized within the volatile memory of the HSMs for as long as necessary to perform the cryptographic operation you have requested.

2.19 How Do I Use an Asymmetric Key to Verify the Signature Result of a Public Key Pair?

In scenarios where public and private key pairs are used, the private key is used for signature and the public key is used for signature verification. The public key can be distributed to the service subject that needs to use the public key. The service subject verifies the signature of the key data. KMS provides the API **get-publickey** for obtaining public keys.

The **RSA_3072** CMK in this case is used to verify signatures. You can use KMS to sign APIs. The request body is as follows:

```
{
 "key_id": "key_id_value",
 "message": "MTIzNA==",
 "signing_algorithm": "RSASSA_PSS_SHA_256",
 "message_type": "RAW"
}
```

The result is as follows:

```
{
 "key_id": "key_id_value",
 "signature": "xxx"
}
```

After the public key is obtained, ensure that the signature is verified.

```
public class RawDataVerifyExample {

    /**
     * Basic authentication information:
     * - ACCESS_KEY: access key of the Huawei Cloud account
     * - SECRET_ACCESS_KEY: Huawei Cloud account secret access key, which is sensitive information. Store
this in ciphertext.
     * - IAM_ENDPOINT: endpoint for accessing IAM. For details, see https://developer.huaweicloud.com/
intl/en-us/endpoint?IAM.
```

```
     * - KMS_REGION_ID: regions supported by KMS. For details, see https://developer.huaweicloud.com/
intl/en-us/endpoint?DEW.
     * - KMS_ENDPOINT: endpoint for accessing KMS. For details, see https://developer.huaweicloud.com/
intl/en-us/endpoint?DEW.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String IAM_ENDPOINT = "https://<IamEndpoint>";
    private static final String KMS_REGION_ID = "<RegionId>";
    private static final String KMS_ENDPOINT = "https://<KmsEndpoint>";

    private static final int SALT_LENGTH = 32;
    private static final int TRAILER_FIELD = 1;
    public static final String RSA_PUBLIC_KEY_BEGIN = "-----BEGIN PUBLIC KEY-----\n";
    public static final String RSA_PUBLIC_KEY_END = "-----END PUBLIC KEY-----";

    // Sample signature data in Base64 encoding format. The original text is 1234.
    private static final String RWA_DATA = "MTIzNA==";

    // Signature value obtained through the sign API of KMS
    private static final String SIGN = "xxx";
    public static void main(String[] args) throws Exception {

        final String keyId = args[0];

        publicKeyVerify(keyId);
    }
   public static void publicKeyVerify(String keyId) throws Exception {

        // 1. Prepare the authentication information for accessing HUAWEI CLOUD.
        final BasicCredentials auth = new BasicCredentials()
                .withIamEndpoint(IAM_ENDPOINT).withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY);

        // 2. Initialize the SDK and transfer the authentication information and the address for the KMS to
access the client.
        final KmsClient kmsClient = KmsClient.newBuilder()
                .withRegion(new Region(KMS_REGION_ID, KMS_ENDPOINT)).withCredential(auth).build();

        // 3. Obtain the public key information. The returned information is in PKCS8 format.
        final ShowPublicKeyRequest showPublicKeyRequest = new ShowPublicKeyRequest()
                .withBody(new OperateKeyRequestBody().withKeyId(keyId));
        final ShowPublicKeyResponse showPublicKeyResponse =
kmsClient.showPublicKey(showPublicKeyRequest);

        // 4. Obtain the public key string.
        final String publicKeyStr = showPublicKeyResponse.getPublicKey().replace(RSA_PUBLIC_KEY_BEGIN, "")
                .replaceAll("\n", "").replace(RSA_PUBLIC_KEY_END, "");

        // 5. Parse the public key.
        final X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(publicKeyStr));
        final KeyFactory keyFactory = KeyFactory.getInstance("RSA", new BouncyCastleProvider());
        final PublicKey publicKey = keyFactory.generatePublic(keySpec);

        // 6. Verify the signature.
        final Signature signature = getSignature();
        signature.initVerify(publicKey);
        signature.update(commonHash(Base64.getDecoder().decode(RWA_DATA)));

        // 7. Obtain the verification result.
        assert signature.verify(Base64.getDecoder().decode(SIGN));

    }
    private static Signature getSignature() throws Exception {
        Signature signature= Signature.getInstance("NONEwithRSASSA-PSS", new BouncyCastleProvider());
        MGF1ParameterSpec mgfParam = new MGF1ParameterSpec("SHA256");
        PSSParameterSpec pssParam = new PSSParameterSpec("SHA256", "MGF1", mgfParam, SALT_LENGTH,
TRAILER_FIELD);
        signature.setParameter(pssParam);
        return signature;
    }
    private static byte[] commonHash(byte[] data) {
        byte[] digest;
        try {
            MessageDigest md = MessageDigest.getInstance("SHA256", 
BouncyCastleProvider.PROVIDER_NAME);
            md.update(data);
            digest = md.digest();
        } catch (Exception e) {
            throw new RuntimeException("Digest failed.");
        }
        return digest;
    }
}
```

2.20 Does an Imported Key Support Rotation?

Imported keys do not support rotation. After the imported key materials are deleted, ensure that the same key materials are imported.

2.21 Does KMS Support Offline Data Encryption and Decryption?

Generally, KMS provides open APIs **encrypt-data** and **decrypt-data** for encrypting and decrypting a small volume of data. The calculation of the APIs is based on KMS, which wraps the ciphertext. So offline data encryption and decryption are not supported.

However, for asymmetric keys, KMS does not wrap the ciphertext. So public keys can be encrypted offline while private keys are decrypted online.

The following shows an example:

RSA_3072 is used for **ENCRYPT_DECRYPT**. After a public key is used to encrypt "*hello world!*" offline, **decrypt-data** is called to decrypt the message using a private key. The RSA/ECB/OAEPWithSHA-256AndMGF1Padding algorithm is used.

```
public class RsaEncryptDataExample {
    /**
     * Basic authentication information:
     * - ACCESS_KEY: Access key of the Huawei Cloud account. For details, see How Do I Obtain an Access 
Key (AK/SK)?.
     * - SECRET_ACCESS_KEY: Secret access key of the Huawei Cloud account. This is sensitive information. 
Store it in ciphertext. For details, see How Do I Obtain an Access Key (AK/SK)?.
     * - IAM_ENDPOINT: Endpoint for accessing IAM.
     * - KMS_REGION_ID: Regions supported by KMS.
     * - KMS_ENDPOINT: Endpoint for accessing KMS.
     * - There will be security risks if the AK/SK used for authentication is directly written into code. Encrypt 
the AK/SK in the configuration file or environment variables for storage.
     * - In this example, the AK/SK stored in the environment variables are used for identity authentication. 
Configure the environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local 
environment first.
     */
    private static final String ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_AK");
    private static final String SECRET_ACCESS_KEY = System.getenv("HUAWEICLOUD_SDK_SK");
    private static final String IAM_ENDPOINT = "https://<IamEndpoint>";
    private static final String KMS_REGION_ID = "<RegionId>";
    private static final String KMS_ENDPOINT = "https://<KmsEndpoint>";

    private static final String RSA_PUBLIC_KEY_BEGIN = "-----BEGIN PUBLIC KEY-----\n";
    private static final String RSA_PUBLIC_KEY_END = "-----END PUBLIC KEY-----";

    private static final String RSAES_OAEP_SHA_256 = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

    private static final String SHA_256 = "SHA-256";

    private static final String MGF1 = "MGF1";

    private static final String HELLO_WORLD = "hello world!";

    public static void main(String[] args) throws Exception {

        final String keyId = args[0];

        publicKeyEncrypt(keyId);
    }

    private static void publicKeyEncrypt(String keyId) throws NoSuchAlgorithmException,
InvalidKeySpecException,
            NoSuchPaddingException, InvalidAlgorithmParameterException, InvalidKeyException,
            IllegalBlockSizeException, BadPaddingException {

        // 1. Prepare the authentication information for accessing Huawei Cloud.
        final BasicCredentials auth = new BasicCredentials()
                .withIamEndpoint(IAM_ENDPOINT).withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY);

        // 2. Initialize the SDK and transfer the authentication information and the address for the KMS to
access the client.
        final KmsClient kmsClient = KmsClient.newBuilder()
                .withRegion(new Region(KMS_REGION_ID, KMS_ENDPOINT)).withHttpConfig(new HttpConfig()
                        .withIgnoreSSLVerification(true)).withCredential(auth).build();

        // 3. Obtain the public key information. The returned information is in PKCS8 format.
        final ShowPublicKeyRequest showPublicKeyRequest = new ShowPublicKeyRequest()
                .withBody(new OperateKeyRequestBody().withKeyId(keyId));
        final ShowPublicKeyResponse showPublicKeyResponse =
kmsClient.showPublicKey(showPublicKeyRequest);

        // 4. Obtain the public key string.
        final String publicKeyStr = showPublicKeyResponse.getPublicKey().replace(RSA_PUBLIC_KEY_BEGIN, "")
                .replaceAll("\n", "").replace(RSA_PUBLIC_KEY_END, "");

        // 5. Obtain the binary public key.
        final X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(publicKeyStr));
        final KeyFactory keyFactory = KeyFactory.getInstance("RSA", new BouncyCastleProvider());
        final PublicKey publicKey = keyFactory.generatePublic(keySpec);

        // 6. Encrypt the string "hello world!" offline using a public key.
        final Cipher cipher = Cipher.getInstance(RSAES_OAEP_SHA_256);
        final OAEPParameterSpec oaepParameterSpec = new OAEPParameterSpec(SHA_256, MGF1,
                new MGF1ParameterSpec(SHA_256), PSource.PSpecified.DEFAULT);
        cipher.init(Cipher.ENCRYPT_MODE, publicKey, oaepParameterSpec);
        final byte[] cipherData = cipher.doFinal(HELLO_WORLD.getBytes(StandardCharsets.UTF_8));

        // 7. Decrypt the ciphertext online using a private key.
        final DecryptDataRequest decryptDataRequest = new DecryptDataRequest()
                .withBody(new DecryptDataRequestBody().withKeyId(keyId)
                        .withEncryptionAlgorithm(DecryptDataRequestBody.EncryptionAlgorithmEnum.RSAES_OAEP
_SHA_256)
                        .withCipherText(Base64.getEncoder().encodeToString(cipherData)));

        final DecryptDataResponse decryptDataResponse = kmsClient.decryptData(decryptDataRequest);

        assert HELLO_WORLD.equals(decryptDataResponse.getPlainText());
    }

}
```

2.22 How Do I Convert an Original EC Private Key into a Private Key in PKCS8 Format?

Scenario

The EC private key is a large integer. However, in the key pair import scenario, the private key must be ASN.1-encoded and then the data must be encoded in binary mode to obtain the DER format, which cannot be obtained by running the OpenSSL command.

This section describes how to convert a 256-bit EC private key into a private key in PKCS8 format.

Environment Preparations

- Create a Java environment and import bouncy castle 1.78 or later.
- Install OpenSSL 1.1.1m or later.

Converting a Private Key to a PKCS8 Object

The following uses a secp256k1 private key as an example. The original private key in hexadecimal format is as follows:

```DC23DA6E913444ABADCE2F42A3B7DC3958569948633EE80AEC46ACCA025234 95```

NOTE

The private key is used as an example only. Do not use it in the actual environment.

Use the following code to convert the private key into a PKCS8 object:

~~~
```java
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPrivateKey;
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPublicKey;
import org.bouncycastle.jce.ECNamedCurveTable;
import org.bouncycastle.jce.interfaces.ECPrivateKey;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.jce.spec.ECNamedCurveParameterSpec;
import org.bouncycastle.jce.spec.ECPrivateKeySpec;
import org.bouncycastle.jce.spec.ECPublicKeySpec;
import org.bouncycastle.math.ec.ECPoint;

import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PublicKey;
import java.security.Security;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.InvalidParameterSpecException;
import java.util.Base64;

public class RawEcPrivateKeyToPKCS8Object {
    public static void main(String[] args)
        throws InvalidParameterSpecException, NoSuchAlgorithmException, InvalidKeySpecException {

        Security.addProvider(new BouncyCastleProvider());
~~~

~~~
        KeyFactory keyFactory = KeyFactory.getInstance("ECDSA", new BouncyCastleProvider());

        ECNamedCurveParameterSpec ecSpec = ECNamedCurveTable.getParameterSpec("secp256k1");
        BigInteger d = new
BigInteger("DC23DA6E913444ABADCE2F42A3B7DC3958569948633EE80AEC46ACCA02523495", 16);
        ECPrivateKeySpec ecPrivateKeySpec = new ECPrivateKeySpec(d, ecSpec);
        BCECPrivateKey ec = new BCECPrivateKey("EC", ecPrivateKeySpec,
BouncyCastleProvider.CONFIGURATION);

        ECPoint q = ecSpec.getG().multiply(((ECPrivateKey) ec).getD());
        ECPublicKeySpec pubSpec = new ECPublicKeySpec(q, ecSpec);
        PublicKey publicKey = keyFactory.generatePublic(pubSpec);

        BCECPrivateKey ec2 = new BCECPrivateKey("EC", ec.engineGetKeyParameters(), (BCECPublicKey)
publicKey,
            ecPrivateKeySpec.getParams(), BouncyCastleProvider.CONFIGURATION);

        System.out.println(Base64.getEncoder().encodeToString(ec2.getEncoded()));
    }
}
```
~~~

The output is as follows:

~~~
```ignorelang
MIGNAgEAMBAGByqGSM49AgEGBSuBBAAKBHYwdAIBAQQg3CPabpE0RKutzi9Co7fcOVhWmUhjPugK7Easyg
JSNJWgBwYFK4EEAAqhRANCAAQWiYvQT8cyVJx3wN85fXw0c2Ppv3SEsgnDaB96rWlz6G2bf2WhBJVD/jF5zb
+5/oxgVIOYDe8EwqYtBwhIJ3Yh
```
~~~

Use the ASN.1 decoding tool:

~~~
```
 <SEQUENCE>
  <INTEGER/>
  <SEQUENCE>
   <OBJECT_IDENTIFIER Comment="ANSI X9.62 public key type"
Description="ecPublicKey">1.2.840.10045.2.1</OBJECT_IDENTIFIER>
   <OBJECT_IDENTIFIER Comment="SECG (Certicom) named elliptic curve"
Description="secp256k1">1.3.132.0.10</OBJECT_IDENTIFIER>
  </SEQUENCE>
  <OCTET_STRING>
   <SEQUENCE>
    <INTEGER>1</INTEGER>
    <OCTET_STRING>0xDC23DA6E913444ABADCE2F42A3B7DC3958569948633EE80AEC46ACCA02523495</
OCTET_STRING>
    <NODE Sign="a0">
     <OBJECT_IDENTIFIER Comment="SECG (Certicom) named elliptic curve"
Description="secp256k1">1.3.132.0.10</OBJECT_IDENTIFIER>
    </NODE>
    <NODE Sign="a1">
     <BIT_STRING
Bits="520">0x000416898BD04FC732549C77C0DF397D7C347363E9BF7484B209C3681F7AAD6973E86D9B7F6
5A1049543FE3179CDBFB9FE8C605483980DEF04C2A62D070848277621</BIT_STRING>
    </NODE>
   </SEQUENCE>
  </OCTET_STRING>
 </SEQUENCE>
```
~~~

Add the following content to the **ec_private_key.pem** file:

~~~
```ignorelang
-----BEGIN PRIVATE KEY-----
MIGNAgEAMBAGByqGSM49AgEGBSuBBAAKBHYwdAIBAQQg3CPabpE0RKutzi9Co7fcOVhWmUhjPugK7Easyg
JSNJWgBwYFK4EEAAqhRANCAAQWiYvQT8cyVJx3wN85fXw0c2Ppv3SEsgnDaB96rWlz6G2bf2WhBJVD/jF5zb
+5/oxgVIOYDe8EwqYtBwhIJ3Yh
-----END PRIVATE KEY-----
```
~~~

Run the following commands to view the EC key information:

~~~
```shell
openssl ec -in ec_private_key.pem -text
```
```ignorelang
read EC key
Private-Key: (256 bit)
priv:
    dc:23:da:6e:91:34:44:ab:ad:ce:2f:42:a3:b7:dc:
    39:58:56:99:48:63:3e:e8:0a:ec:46:ac:ca:02:52:
    34:95
pub:
    04:16:89:8b:d0:4f:c7:32:54:9c:77:c0:df:39:7d:
    7c:34:73:63:e9:bf:74:84:b2:09:c3:68:1f:7a:ad:
    69:73:e8:6d:9b:7f:65:a1:04:95:43:fe:31:79:cd:
    bf:b9:fe:8c:60:54:83:98:0d:ef:04:c2:a6:2d:07:
    08:48:27:76:21
ASN1 OID: secp256k1
writing EC key
-----BEGIN EC PRIVATE KEY-----
MHQCAQEEINwj2m6RNESrrc4vQqO33DlYVplIYz7oCuxGrMoCUjSVoAcGBSuBBAAK
oUQDQgAEFomL0E/HMlScd8DfOX18NHNj6b90hLIJw2gfeq1pc+htm39loQSVQ/4x
ec2/uf6MYFSDmA3vBMKmLQcISCd2IQ==
-----END EC PRIVATE KEY-----

```
~~~

If the commands can be executed properly, the following **DER** command is generated:

~~~
```shell
openssl pkcs8 -topk8 -inform PEM -outform DER -in ec_private_key.pem -out ec_private_key.der -nocrypt```
~~~

3 CSMS Related

3.1 Why Cannot I Delete the Version Status of a Secret?

SYSCURRENT and **SYSPREVIOUS** are preconfigured statuses and cannot be deleted.

4 KPS Related

4.1 How Do I Handle an Import Failure of a Key Pair Created Using PuTTYgen?

Symptom

When a key pair created using PuTTYgen was imported to the management console, the system displayed a message indicating that importing the public key failed.

Possible Causes

The format of the public key content does not meet system requirements.

Storing a public key by clicking **Save public key** will change the format of the public key content. Importing such a public key will fail because the key does not pass the format verification by the system.

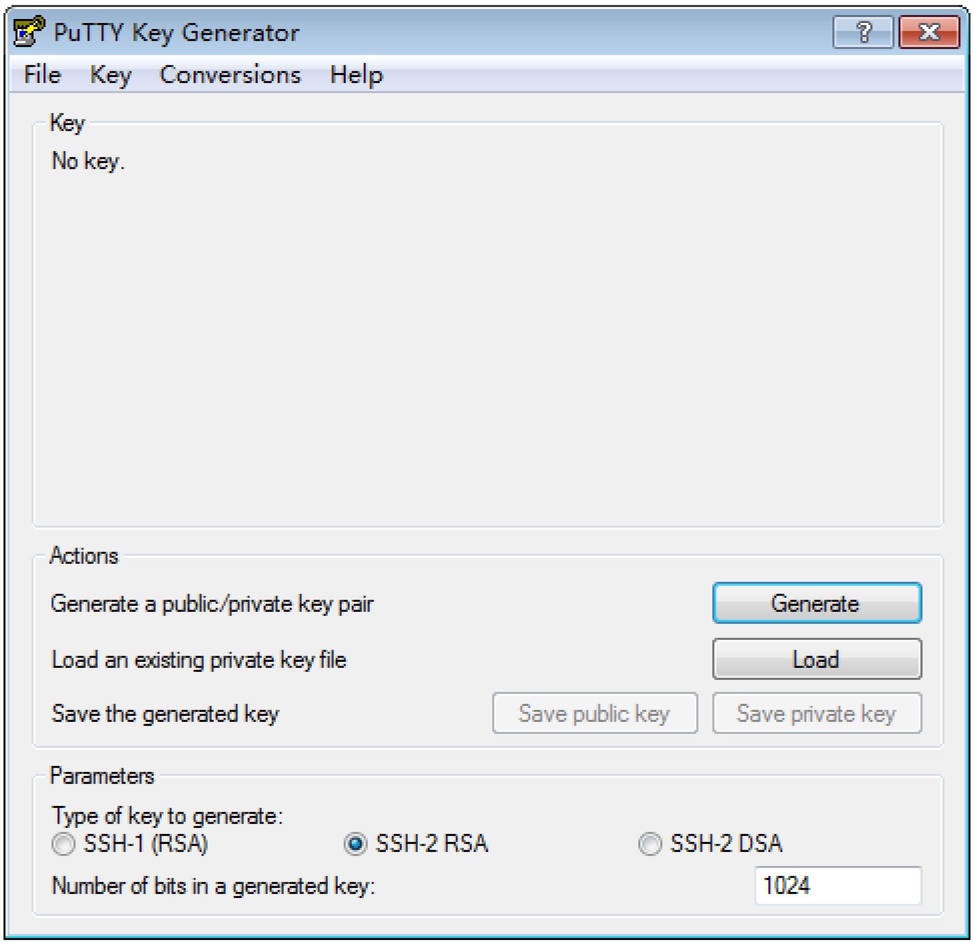
Procedure

Use the locally stored private key and **PuTTY Key Generator** to restore the format of the public key content. Then, import the public key to the management console.

Step 1 Restore the public key file in the correct format.

1. Double-click **PuTTYgen.exe**. The **PuTTY Key Generator** page is displayed, as shown in **Figure 4-1**.

Figure 4-1 Main interface of the PuTTY Key Generator



Key
No key.

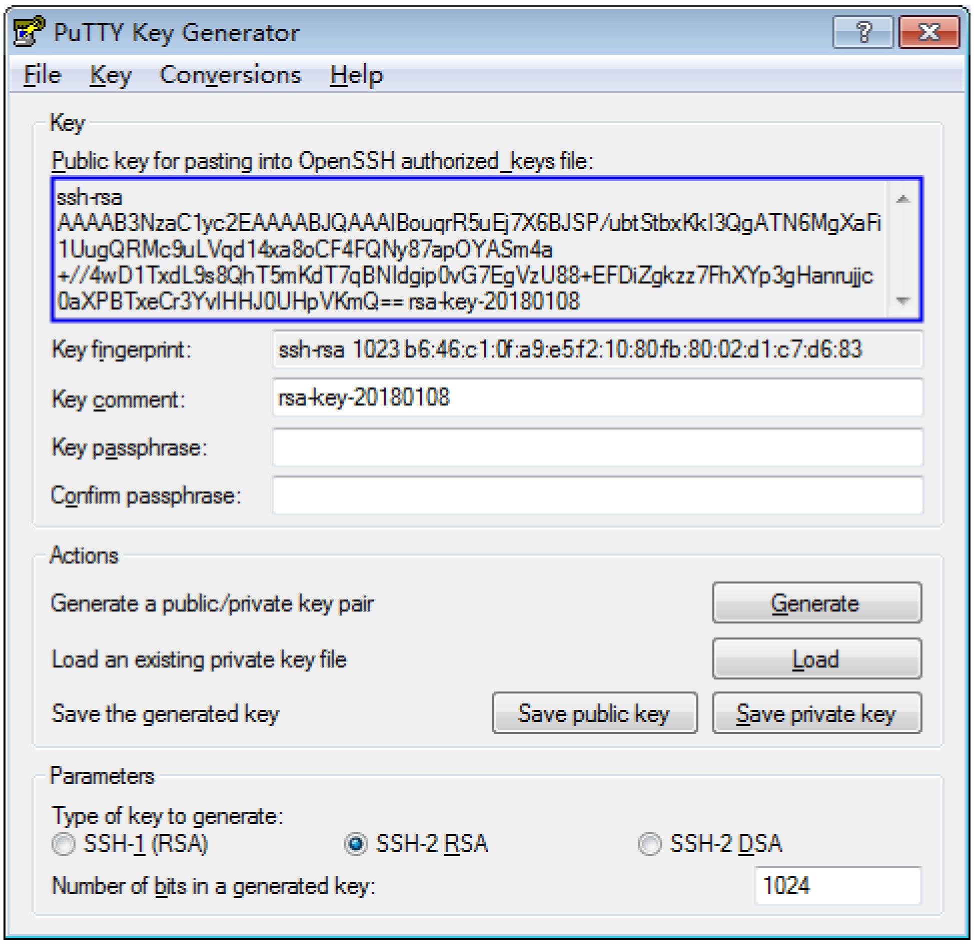
Actions
Generate a public/private key pair
Load an existing private key file
Save the generated key

Parameters
Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA
Number of bits in a generated key: 1024

2. Click **Load** and select the private key.

   The system automatically loads the private key and restores the format of the public key content in **PuTTY Key Generator**. The content in the red box in **Figure 4-2** is the public key with the format meeting system requirements.

Figure 4-2 Restoring the format of the public key content



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBouqrR5uEj7X6BJSP/ubtStbxKkI3QgATN6MgXaFi
1UugQRMc9uLVqd14xa8oCF4FQNy87apOYASm4a
+//4wD1TxdL9s8QhT5mKdT7qBNldgip0vG7EgVzU88+EFDiZgkzz7FhXYp3gHanrujjc
0aXPBTxeCr3YvlHHJ0UHpVKmQ== rsa-key-20180108
```

Key fingerprint: ssh-rsa 1023 b6:46:c1:0f:a9:e5:f2:10:80:fb:80:02:d1:c7:d6:83

Key comment: rsa-key-20180108

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

3. Copy the information in the blue square and save it in a local **.txt** file.

NOTICE

Do not save the public key by clicking **Save public key**. If you save a public key using **Save public key**, the public key format will be changed and cannot be imported to the management console directly.

Step 2 Import the public key file in the correct format to the KPS console.

1. Log in to the **DEW console**.
2. In the navigation pane, click **Key Pair Service**.
3. On the **Key Pair Service** page, click **Import Key Pair**.
4. Click **Select File**, select the .txt public key file, or copy and paste the public key content to the text box of the public key content.
5. Click **OK** to import the public key file.

----End

4.2 What Should I Do When I Fail to Import a Key Pair Using Internet Explorer 9?

Symptom

Importing a key pair may fail if Internet Explorer 9 is used.

Procedure

Step 1 Click ⚙ in the upper right corner of the browser.

Step 2 Select **Internet Options**.

Step 3 Click the **Security** tab in the displayed dialog box.

Step 4 Click **Internet**.

Step 5 If the security level indicates **Custom**, click **Default Level** to restore to the default settings.

Step 6 Move the scroll bar to set the security level to **Medium** and click **Apply**.

Step 7 Click **Custom Level**.

Step 8 Set **Initialize and script ActiveX controls not marked as safe for scripting** to **Prompt**.

Step 9 Click **Yes**.

----End

4.3 How Do I Handle the Failure in Binding a Key Pair?

Symptom

Failed to bind the key pair to the ECS.

NOTE

- The **Failed Key Pair Task** dialog box only records and displays failed key pair operations on ECSs, which do not affect the ECS status and subsequent operations. You can locate the target failure record and click **Delete** in the **Operation** column, or can click **Delete All** to delete all failure records.
- Click **Learn more** to view related documents.

Typical Errors in KPS Key Pair Tasks

Category	Error Information	Error Cause	Solution
SSH connection failure	Invalid server login credential. Message: ******	Failed to log in to the ECS in SSH mode using the authentication information provided by the user. Possible causes: 1. The authentication information provided by the user is incorrect. 2. SSH configuration of the ECS has been modified. 3. The ECS is locked due to multiple authentication failures.	Use the password or private key to log in to the ECS. Check whether the password is correct.
	SSH connection failure. Check your port.	Failed to log in to the ECS in SSH mode using the authentication information provided by the user. Possible causes: 1. The SSH service is not bound to the specified port. 2. The SSH service is abnormal.	Log in to the ECS using SSH. Check the listening status of the service port.
	Timeout: socket is not established. Check your security group rules.	Failed to log in to the ECS in SSH mode using the authentication information provided by the user. Possible causes: 1. The inbound direction of the specified port of the ECS security group is not open to 100.125.0.0/16. 2. Firewall rules have been configured for the ECS.	Log in to the ECS using SSH. Check the security group and firewall rules.
ECS status check failure	ECS is being executed.	Failed to check the ECS status. Possible cause: The ECS is being created.	Wait until the ECS is created.
	Image information of the server not found.	Failed to check the ECS status. Possible cause: An unregistered image is used.	Use a registered image.
	ECS OS not supported.	Failed to check the ECS status. Possible cause: An unsupported OS is used.	Use an OS supported by KPS.

Category	Error Information	Error Cause	Solution
	Server status error.	Failed to check the ECS status. Possible causes: ● The ECS is not in the **ACTIVE** state. ● The ECS is not in the **SHUTOFF** state.	Go to the ECS console and change the ECS status.
	System volume not found.	Failed to check the ECS status. Possible cause: Failed to query the disks mounted to the ECS.	Go to the ECS console and mount a disk to the ECS.
Key pair binding failure	Too many key pairs to be bound in batches.	Failed to bind key pairs in batches. Possible cause: The number of key pairs to be bound exceeds the upper limit.	Delete some key pair to be bound.
	Key pairs to be bound in batches are inconsistent.	Failed to bind key pairs in batches. Possible cause: Different key pairs are used.	Use only the same key pairs for batch binding.
	Unavailable ECS flavor.	Failed to bind the key pair. Possible cause: Failed to obtain the flavor information.	Check the ECS flavor.
Key pair API parameter check	The imported private key does not match the public key.	Failed to check the key pairs. Possible causes: ● The format of the current public and private keys is incorrect. ● The public and private keys are not a pair.	Check the format of the public and private keys and check whether they are a pair.
	Invalid parameter.	Failed to check the key pair API parameter. Possible causes: 1. Mandatory parameters are left blank. 2. The key pair name is invalid. 3. Other parameters are incorrectly configured.	Check the invoking parameters.
	Invalid key pair type.	Failed to check the key pair API parameter. Possible cause: The key pair type is not ssh or x509 as required.	Check the invoking parameters.

Handling Procedure

You can troubleshoot the fault by performing the following steps:

Step 1 Check the ECS status.

- If it is running, go to **Step 2**.
- If it is shut down, go to **Step 5**.

Step 2 Use the password to log in to the ECS to check whether the password is correct.

- If it is correct, go to **Step 4**.
- If it is incorrect, use the correct password to bind the key pair again.

Step 3 Check whether the permission path and owner group of the **/root/.ssh/authorized_keys** file on the ECS have been modified.

- If yes, restore the permission to the following:
  - The owner group of each level has the **root:root** permission.
  - The permission for the .ssh file is 700.
  - The permission for **authorized_keys** is 600.
- If no, go to **Step 4**.

Step 4 Check whether the **/root/.ssh/authorized_keys** file of the ECS has been modified.

- If yes, restore the original content of the **/root/.ssh/authorized_keys** file based on the site requirements.
- If no, go to **Step 5**.

Step 5 Check whether the inbound direction of port 22 of the ECS security group is open to 100.125.0.0/16. That is, 100.125.0.0/16 can remotely connect to Linux ECSs through SSH.

- If yes, go to **Step 6**.
- If no, add the following security group rule and bind the key pair again. For details about how to add a security group, see **Adding a Security Group Rule**.

Direction	Protocol/ Application	Port	Source
Inbound	SSH (22)	22	100.125.0.0/16

Step 6 Check whether the ECS can be powered on, shut down, and logged in to.

- If yes, bind the key pair again.
- If no, go to **Step 7**.

Step 7 Check whether the network is faulty.

- If yes, contact technical support to check and locate the fault.
- If no, bind the key pair again.

----End

4.4 How Do I Handle the Failure in Replacing a Key Pair?

Symptom

Failed to replace the key pair on the ECS.

NOTE

The **Failed Key Pair Task** dialog box only records and displays failed key pair operations on ECSs, which do not affect the ECS status and subsequent operations. You can click **Delete** in the row of the failure record to delete it, or you can click **Delete All** to delete all failure records.

Possible Causes

- An incorrect or invalid private key has been provided.
- The inbound direction of port 22 of the ECS security group is not open to 100.125.0.0/16.
- SSH configuration of the ECS has been modified.
- The ECS has been shut down, started, or a disk has been detached during the process of replacing the key pair.
- The network connection is faulty.
- Firewall rules have been configured for the ECS.

Handling Procedure

Step 1 Use the SSH key pair to log in to the ECS and check whether the private key is correct.

- If it is correct, go to **Step 2**.
- If it is incorrect, use the correct private key to replace the key pair again.

Step 2 Check whether the **/root/.ssh/authorized_keys** file of the ECS has been modified.

- If yes, restore the original content of the **/root/.ssh/authorized_keys** file based on the site requirements.
- If no, go to **Step 3**.

Step 3 Check whether the inbound direction of port 22 of the ECS security group is open to 100.125.0.0/16. That is, 100.125.0.0/16 can remotely connect to Linux ECSs through SSH.

- If yes, go to **Step 4**.
- If no, add the following security group rule and replace the key pair again.

Direction	Protocol/ Application	Port	Source
Inbound	SSH (22)	22	100.125.0.0/16

Step 4 Check whether the ECS can be powered on, shut down, and logged in to.

- If yes, replace the key pair again.
- If no, go to **Step 5**.

Step 5 Check whether the network is faulty.

- If yes, contact technical support to check and locate the fault.
- If no, replace the key pair again.

----End

4.5 How Do I Handle the Failure in Resetting a Key Pair?

Symptom

Failed to reset the key pair on the ECS.

NOTE

The **Failed Key Pair Task** dialog box only records and displays failed key pair operations on ECSs, which do not affect the ECS status and subsequent operations. You can click **Delete** in the row of the failure record to delete it, or you can click **Delete All** to delete all failure records.

Possible Causes

- The inbound direction of port 22 of the ECS security group is not open to 100.125.0.0/16.
- The ECS has been shut down, started, or a disk has been detached during the process of resetting the key pair.
- The network connection is faulty.
- Firewall rules have been configured for the ECS.

Handling Procedure

Step 1 Check whether the inbound direction of port 22 of the ECS security group is open to 100.125.0.0/16. That is, 100.125.0.0/16 can remotely connect to Linux ECSs through SSH.

- If yes, go to **Step 2**.
- If no, add the following security group rule and reset the key pair again.

Direction	Protocol/ Application	Port	Source
Inbound	SSH (22)	22	100.125.0.0/16

Step 2 Check whether the ECS can be powered on, shut down, and logged in to.

- If yes, reset the key pair again.

- If no, go to **Step 3**.

Step 3 Check whether the network is faulty.

- If yes, contact technical support to check and locate the fault.
- If no, reset the key pair again.

----End

4.6 How Do I Handle the Failure in Unbinding a Key Pair?

Symptom

Failed to unbind the key pair from the ECS.

NOTE

The **Failed Key Pair Task** dialog box only records and displays failed key pair operations on ECSs, which do not affect the ECS status and subsequent operations. You can locate the target failure record and click **Delete** in the **Operation** column, or can click **Delete All** to delete all failure records.

Possible Causes

- An incorrect or invalid private key has been provided.
- The inbound direction of port 22 of the ECS security group is not open to 100.125.0.0/16.
- SSH configuration of the ECS has been modified.
- The ECS has been shut down, started, or a disk has been detached during the process of unbinding the key pair from the ECS.
- The network connection is faulty.
- A firewall rule is configured when an ECS is started and unbound.
- Unbinding a key pair on KPS is not supported by the current OS.

  You can unbind an ECS on the KPS console for the following OSs: EulerOS, CentOS, RedHat, SUSE, Debian, OpenSUSE, Oracle Linux, Fedora, Ubuntu, Huawei Cloud EulerOS, AlmaLinux, Rocky Linux, CentOS Stream, and OpenEuler.

Handling Procedure

Step 1 Check the ECS status.

- If it is running, go to **Step 2**.
- If it is shut down, go to **Step 4**.

Step 2 Use the SSH key pair to log in to the ECS and check whether the private key is correct.

- If it is correct, go to **Step 4**.
- If it is incorrect, use the correct private key to unbind the key pair again.

Step 3 Check whether the **/root/.ssh/authorized_keys** file of the ECS has been modified.

- If yes, restore the original content of the **/root/.ssh/authorized_keys** file.
- If no, go to **Step 4**.

Step 4 Check whether the inbound direction of port 22 of the ECS security group is open to 100.125.0.0/16. That is, 100.125.0.0/16 can remotely connect to Linux ECSs through SSH.

- If yes, go to **Step 5**.
- If no, add the following security group rule and unbind the key pair again.

Direction	Protocol/ Application	Port	Source
Inbound	SSH (22)	22	100.125.0.0/16

Step 5 Check whether the ECS can be powered on, shut down, and logged in to.

- If yes, unbind the key pair again.
- If no, go to **Step 6**.

Step 6 Check whether the network is faulty.

- If yes, contact technical support to check and locate the fault.
- If no, unbind the key pair again.

----End

4.7 Do I Need to Restart Servers After Replacing Its Key Pair?

No. Key pair replacement does not affect services.

4.8 How Do I Enable the Password Login Mode for an ECS?

If you disable the password login mode when binding a key pair to an ECS, you can enable the password login mode again later when you need to.

Procedure

The following example describes how to log in to the ECS using PuTTY and enable the password login mode.

Step 1 Double-click **PuTTY.EXE**. The **PuTTY Configuration** page is displayed.

Step 2 Choose **Connection** > **Data**. Enter the image username in **Auto-login username**.

NOTE

- If the public image of the **CoreOS** is used, the username of the image is **core**.
- For a **non-CoreOS** public image, the username of the image is **root**.

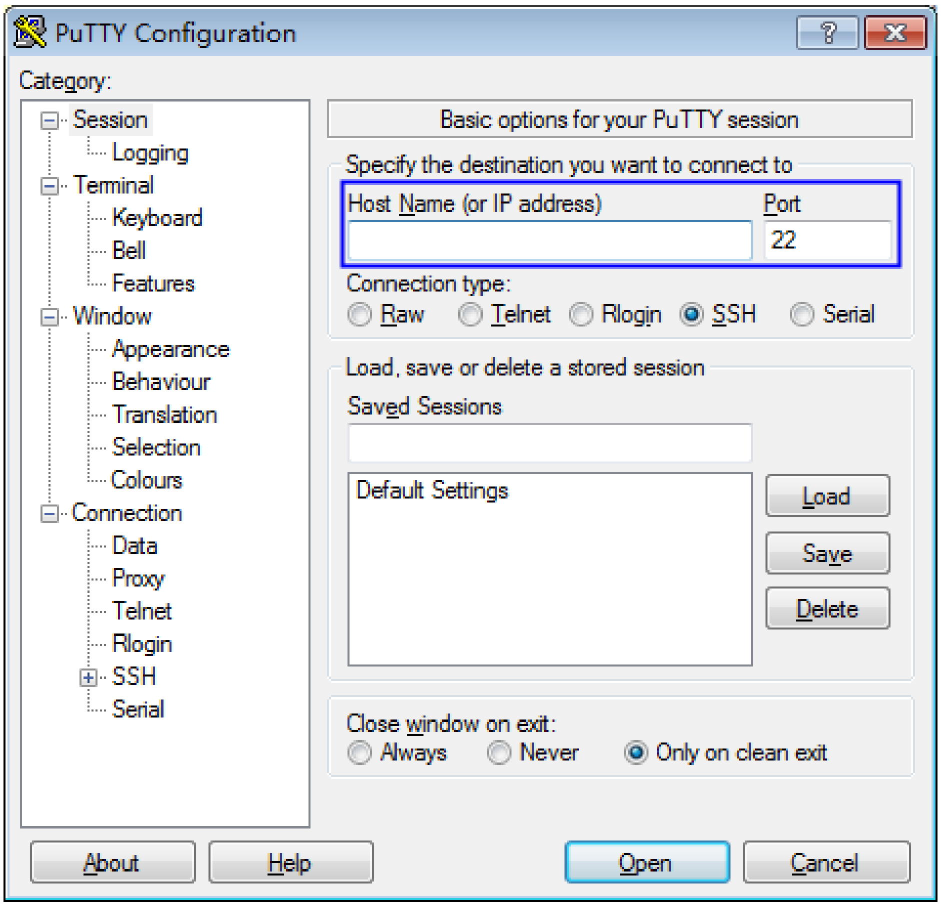
Step 3 Choose **Connection** > **SSH** > **Auth**. In **Private key file for authentication**, click **Browse** and select a private key file (in the **.ppk** format).

NOTE

If the file is in the .pem format, convert it by referring to **Converting the Private Key File in the .pem Format to the .ppk Format**.

Step 4 Click **Session** and enter the EIP of the ECS under **Host Name (or IP address)**.

Figure 4-3 Configuring the EIP



Step 5 Click **Open** to log in to the ECS.

Step 6 Run the following command to open the **/etc/ssh/sshd_config** file:

vi /etc/ssh/sshd_config

Step 7 Press **i** to enter the editing mode and enable the password login mode.

- For a non-SUSE operating system, change the value of **PasswordAuthentication** to **yes**.
```
PasswordAuthentication yes
```
- For a SUSE operating system, change the values of **PasswordAuthentication** and **UsePAM** to **yes**.
```
PasswordAuthentication yes
UsePAM yes
```

NOTE

- Non-SUSE OS

  To disable password login, change the value of **PasswordAuthentication** to **no**. If the **PasswordAuthentication** parameter is not contained in the **/etc/ssh/sshd_config** file, add it and set it to **no**.
- SUSE OS

  To disable password login, change the values of **PasswordAuthentication** and **UsePAM** to **no**. If the file does not contain the **PasswordAuthentication** and **UsePAM** parameters, add the parameters and set the values to **no**.

Step 8 Press **Esc** to exit the editing mode.

Step 9 Enter **:wq** and press **Enter** to save and exit.

Step 10 Run the following command to restart the SSH service for the configuration to take effect:

- Non-Ubuntu14.xx OS

  service sshd restart
- Ubuntu14.xx OS

  service ssh restart

----End

4.9 How Do I Handle the Failure in Logging In to ECS After Unbinding the Key Pair?

Symptom

- If the login mode is set to **Key Pair** when purchasing an ECS, after I unbind the initial password, I do not have the password or key pair to log in to the ECS. How can I solve this problem?
- When I bind a key pair to the ECS on the KPS management console, I disabled the password login mode. After the key pair is unbound, I have no password and key pair to log in to the ECS. How can I solve this problem?

Procedure

Method 1: resetting the password

Reset the password on the ECS console and log in to the ECS using the password. For details, see *Elastic Cloud Server User Guide*.

Method 2: resetting the key pair

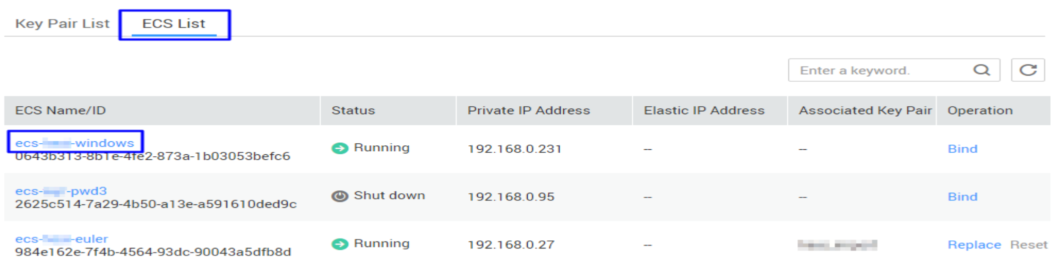
Shut down the ECS, bind the key pair to the ECS on the KPS console, and use the key pair to log in to the ECS. The procedure is as follows:

Step 1 Log in to the **DEW console**.

Step 2 Click in the upper left corner of the console and select a region or project.

Step 3 Click **ECS List** to view ECSs, as shown in **Figure 4-4**.

Figure 4-4 ECS list



ECS Name/ID	Status	Private IP Address	Elastic IP Address	Associated Key Pair	Operation
ecs-[redacted]-windows 0643b313-8b1e-4fe2-873a-1b03053befc6	Running	192.168.0.231	--	--	Bind
ecs-[redacted]-pwd3 2625c514-7a29-4b50-a13e-a591610ded9c	Shut down	192.168.0.95	--	--	Bind
ecs-[redacted]-euler 984e162e-7f4b-4564-93dc-90043a5dfb8d	Running	192.168.0.27	--	[redacted]	Replace Reset

Step 4 Click the target ECS name to access its details page.

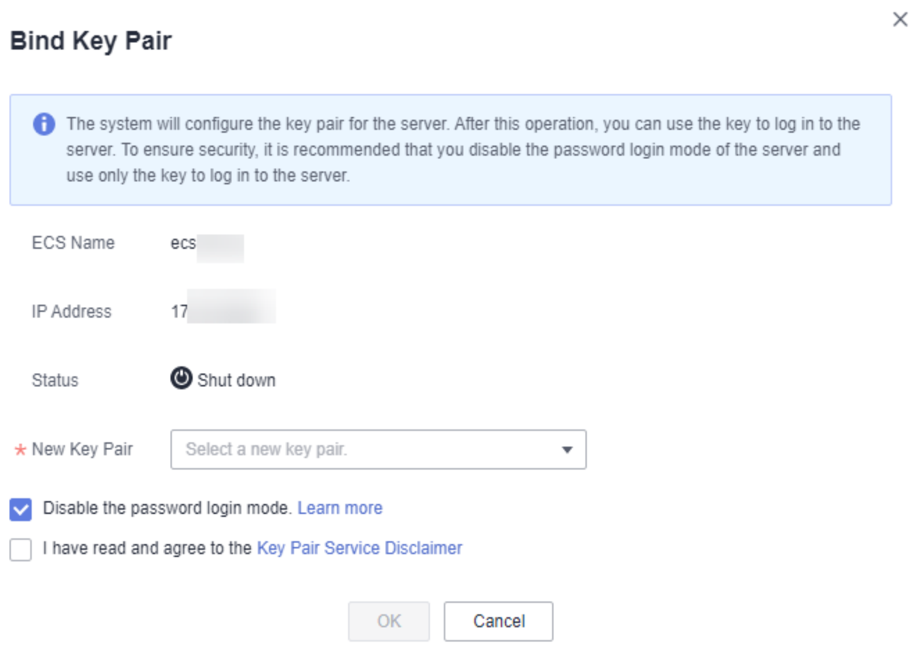
Step 5 Click **Shut Down** in the upper right corner of the page.

Step 6 Return to the ECS list page by referring to step **Step 4**.

Step 7 Locate the target ECS and click **Bind**.

Step 8 Select a new key pair from the drop-down list box of **New Key Pair**.

Figure 4-5 Binding a key pair



Bind Key Pair

The system will configure the key pair for the server. After this operation, you can use the key to log in to the server. To ensure security, it is recommended that you disable the password login mode of the server and use only the key to log in to the server.

ECS Name	ecs[redacted]
IP Address	17[redacted]
Status	Shut down
* New Key Pair	Select a new key pair.

[✓] Disable the password login mode. Learn more

☐ I have read and agree to the Key Pair Service Disclaimer

Step 9 You can choose whether to disable the password login mode as necessary. By default, the password login mode is disabled.

NOTE

- If you do not disable the password login mode, you can use the password or the key pair to log in to the ECS.
- If the password login mode is disabled, you can use only the key pair to log in to the ECS. If you need to use the password login mode later, you can enable the password login mode again. For details, see **4.8 How Do I Enable the Password Login Mode for an ECS?**.

Step 10 Read and select **I have read and agree to the Key Pair Service Disclaimer**.

Step 11 Click **OK**. The key pair is bound. You can use the key pair to log in to the ECS.

----End

4.10 What Should I Do If My Private Key Is Lost?

For Private Key Managed in KPS

You can export the private key from KPS again.

For Private Key Not Managed in KPS

The private key cannot be retrieved.

You can bind a key pair to the ECS again by resetting the password or key pair. For details, see "How Do I Handle the Failure in Logging In to ECS After Unbinding the Key Pair?".

4.11 How Do I Convert the Format of a Private Key File?

Converting the Private Key File in the .ppk Format to the .pem Format

The private key to be uploaded or copied to the text box must be in the .pem format. If the file is in the .ppk format, perform the following steps:

Step 1 Visit the following website and download PuTTY and PuTTYgen:

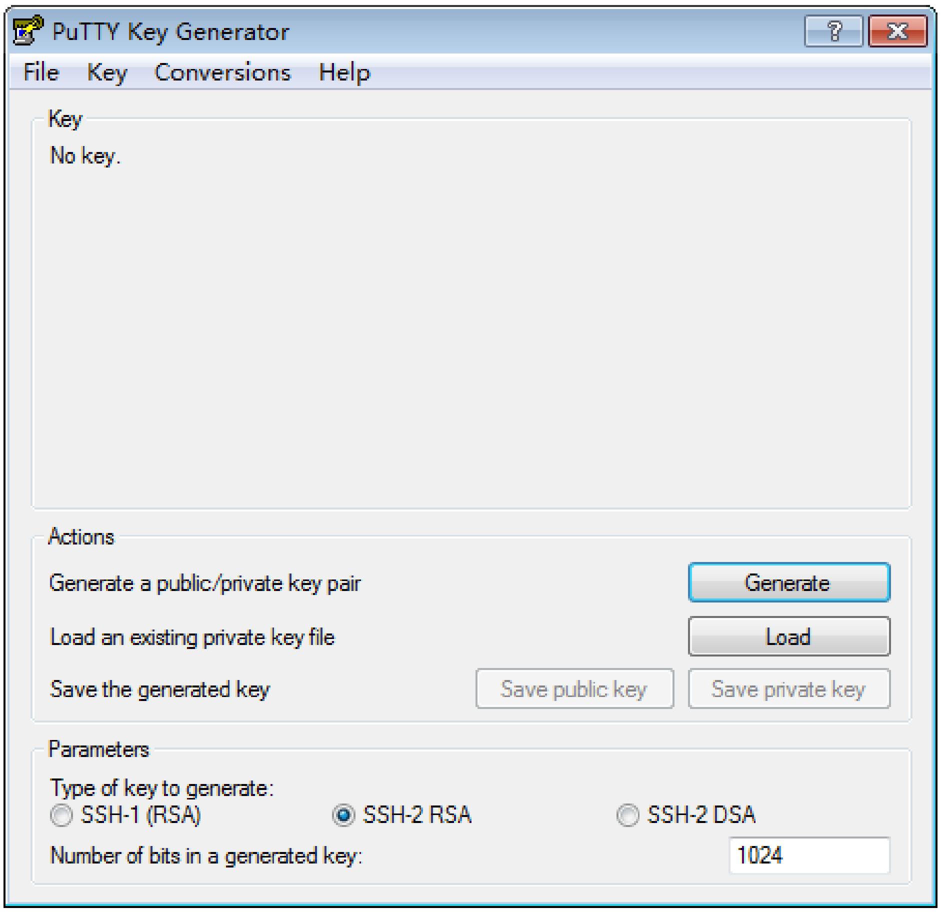
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

NOTE

PuTTYgen is a private key generator, which is used to create a key pair that consists of a public key and a private key for PuTTY.

Step 2 Double-click **PuTTYGEN.exe**. The **PuTTY Key Generator** page is displayed, as shown in **Figure 4-6**.

Figure 4-6 PuTTY Key Generator



No key.

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

Step 3 Choose **Conversions** > **Import Key** to import the private key file in the **.ppk** format.

Step 4 Choose **Conversions** > **Export OpenSSH Key**, the **PuTTYgen Warning** dialog box is displayed.

Step 5 Click **Yes** to save the file in the **.pem** format.

----End

Converting the Private Key File in the .pem Format to the .ppk Format

When you use PuTTY to log in to a Linux ECS, the private key must be in the .ppk format. If the file is in .pem format, perform the following steps to covert its format:

Step 1 Visit the following website and download PuTTY and PuTTYgen:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

NOTE

PuTTYgen is a private key generator, which is used to create a key pair that consists of a public key and a private key for PuTTY.

Step 2 Double-click **PuTTYgen.exe**. The **PuTTY Key Generator** window is displayed.

Step 3 In the **Actions** area, click **Load** and import the private key file that you stored when purchasing the ECS.

Ensure that the private key file format is included in **All files (*.*)**.

Step 4 Click **Save private key**.

Step 5 Save the converted private key, for example, **kp-123.ppk**, to a local directory.

----End

4.12 Can I Change the Key Pair of a Server?

Yes.

You can unbind, reset, and replace the key pair bound to the ECS. For details, see **Managing Key Pairs**.

4.13 Can a Key Pair Be Shared by Multiple Users?

Key pairs cannot be shared across accounts, but can be shared by the IAM users under the same account in either of the following ways:

- Import a key pair. To let multiple IAM users use the same key pair, you can create a key pair (by using PuTTYgen or other tools) and import it as an IAM user resource. For details, see **Importing a Key Pair**.
- Upgrade a private key pair to an account key pair. You can upgrade a key pair created on the management console by referring to **Creating a Key Pair Using the Management Console**. Or you can upgrade a created key pair by referring to **Upgrading a Key Pair**.

4.14 How Do I Obtain the Public or Private Key File of a Key Pair?

Obtaining a Private Key File

When you **create a key pair**, your private key file will be automatically downloaded.

- If the private key is not managed, it cannot be downloaded later. Keep it properly.
- If you have authorized Huawei Cloud to manage the private key, you can export the private key anytime as required.

Obtaining a Public Key File

- If a key pair was created on the management console, the public key is automatically stored in Huawei Cloud. To obtain the public key file, which is in TXT format, locate the key pair in the list and click **Download Public Key** in the **Operation** column.
- If a key pair was created using PuTTYgen, you can find its public key in the storage path on your local PC.

4.15 What Can I Do If an Error Is Reported When an Account Key Is Created or Upgraded for the First Time?

Creating an Account Key Pair for the First Time

When creating an account key pair for the first time, you need to use a user with the Tenant Administrator system role.

Upgrading an Account Key Pair for the First Time

After you upgrade a key pair to an account key pair, all users under your account can view and use the key pair. If a key pair name is the same as the private key pair name of another sub-user, the upgrade cannot be performed. To upgrade key pairs, users with the Tenant Administrator system role must perform the upgrade at least once. The number of key pairs to be upgraded is not limited.

4.16 Will the Account Key Pair Quota Be Occupied After a Private Key Pair Is Upgraded to an Account Key Pair?

Yes.

If a private key pair is upgraded to an account key pair, the account key pair quota is occupied.

5 Dedicated HSM Related

5.1 How Does Dedicated HSM Ensure the Security for Key Generation?

- A key is created by the user remotely. During the creation, only the UKey owned by the user is involved in the authentication.
- The HSM configuration and preparation of internal keys can be performed only after being authenticated by using the UKey as the credential.

The user has full control over the generation, storage, and access of keys. Dedicated HSM is only responsible for monitoring and managing HSMs and related network facilities.

5.2 Does an Equipment Room Administrator Have the Super Management Permission to Insert Privileged Ukeys to Steal Information?

UKeys are owned only by users who purchased Dedicated HSM instances. Equipment room personnel do not have the super administrator role.

Sensitive data (keys) is stored in chips. Even HSM vendor cannot access the internal key information.

5.3 What HSMs Are Used for Dedicated HSM?

Dedicated HSM uses HSMs that have earned China State Cryptography Administration (CSCA) certification and FIPS 140-2 level 3 certification, achieving high security.

5.4 What APIs Does Dedicated HSM Support?

Dedicated HSM provides the same functions and interfaces as physical cryptographic devices, helping you easily migrate services to the cloud. Supported APIs include PKCS#11 and CSP.

For details, see **Editions**.