数据加密服务

最佳实践

文档版本	05
发布日期	2022-03-14

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 加解密小量数据

场景说明

当有少量数据（例如：口令、证书、电话号码等）需要加解密时，用户可以通过密钥管理服务（Key Management Service，KMS）界面使用在线工具加解密数据，或者调用KMS的API接口使用指定的用户主密钥直接加密、解密数据。

约束条件

当前支持不大于4KB的小数据加解密。

在线工具加解密

- **加密数据**

步骤1 单击目标用户主密钥的别名，进入密钥详细信息在线工具加密数据页面。

步骤2 在“加密”文本框中输入待加密的数据，如图1-1所示。

图 1-1 加密数据

步骤3 单击“执行”，右侧文本框显示加密后的密文数据。

说明

- 加密数据时，使用当前指定的密钥加密数据。
- 用户可单击“清除”，清除已输入的数据。
- 用户可单击“复制到剪切板”拷贝加密后的密文数据，并保存到本地文件中。

- **解密数据**

步骤4 解密数据时，可单击任意“启用”状态的非默认主密钥别名，进入该密钥的在线工具页面。

步骤5 单击“解密”，在左侧文本框中数据待解密的密文数据，如图1-2所示。

说明

- 在线工具自动识别并使用数据被加密时使用的密钥解密数据。
- 若该密钥已被删除，会导致解密失败。

图 1-2 解密数据

步骤6 单击“执行”，右侧文本框中显示解密后的明文数据。

说明

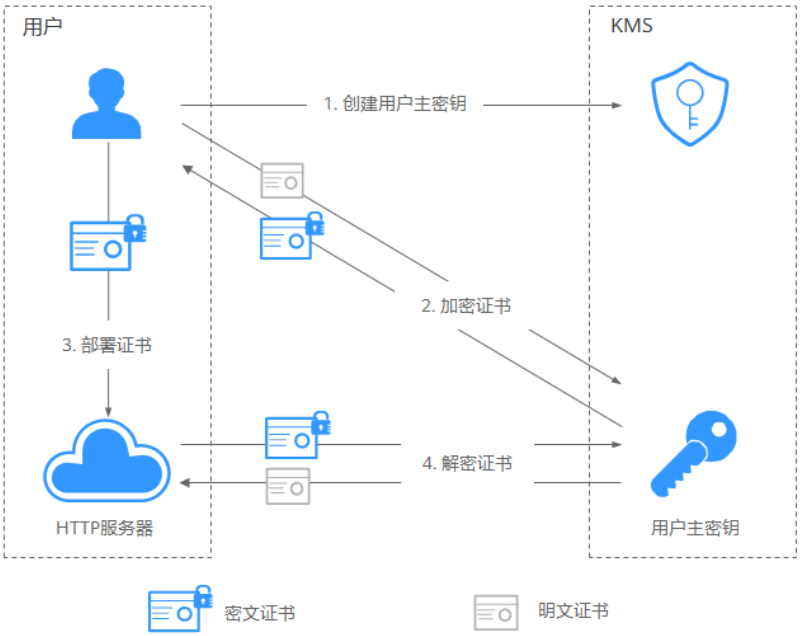
用户可直接单击“复制到剪切板”拷贝解密后的明文数据，并保存到本地文件中。

----结束

调用 API 接口加解密

以保护服务器HTTPS证书为例，采用调用KMS的API接口方式进行说明，如图1-3所示。

图 1-3 保护服务器 HTTPS 证书



流程说明如下：

1. 用户需要在KMS中创建一个用户主密钥。

2. 用户调用KMS的“encrypt-data”接口，使用指定的用户主密钥将明文证书加密为密文证书。
3. 用户在服务器上部署密文证书。
4. 当服务器需要使用证书时，调用KMS的“decrypt-data”接口，将密文证书解密为明文证书。

2 加解密大量数据

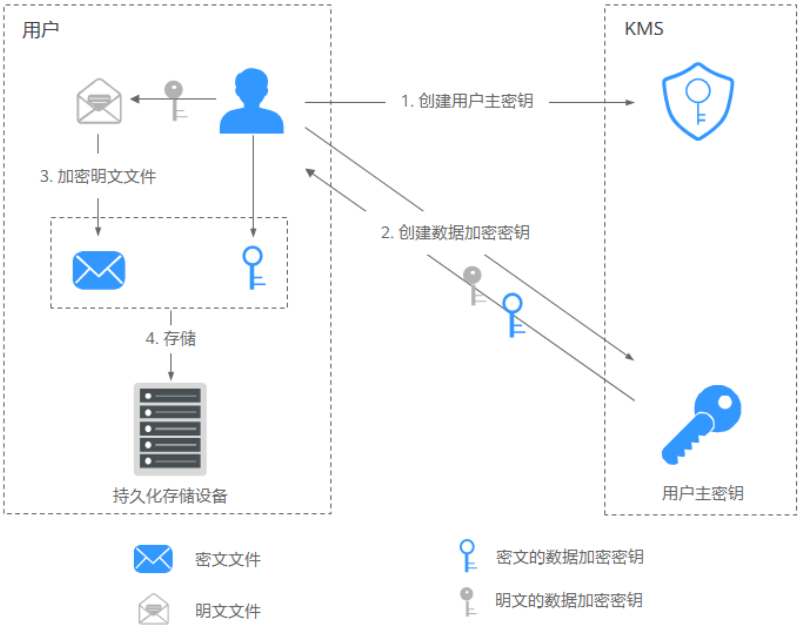
场景说明

当有大量数据（例如：照片、视频或者数据库文件等）需要加解密时，用户可采用信封加密方式加解密数据，无需通过网络传输大量数据即可完成数据加解密。

加密和解密原理

- 大量数据加密

图 2-1 加密本地文件



说明如下：

a. 用户需要在KMS中创建一个用户主密钥。

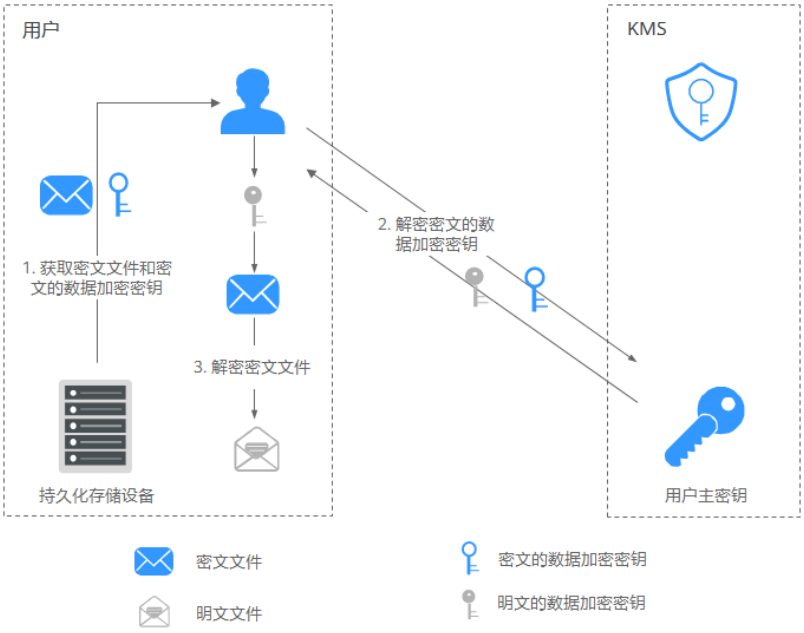
b. 用户调用KMS的“create-datakey”接口创建数据加密密钥。用户得到一个明文的数据加密密钥和一个密文的数据加密密钥。其中**密文的数据加密密钥**是由指定的**用户主密钥**加密**明文的数据加密密钥**生成的。

c. 用户使用明文的数据加密密钥来加密明文文件，生成密文文件。

d. 用户将密文的数据加密密钥和密文文件一同存储到持久化存储设备或服务中。

- 大量数据解密

图 2-2 解密本地文件



说明如下：

a. 用户从持久化存储设备或服务中读取密文的数据加密密钥和密文文件。

b. 用户调用KMS的“decrypt-datakey”接口，使用对应的用户主密钥（即生成密文的数据加密密钥时所使用的用户主密钥）来解密密文的数据加密密钥，取得明文的数据加密密钥。

若对应的用户主密钥被误删除，会导致解密失败。因此，需要妥善管理好用户主密钥。

c. 用户使用明文的数据加密密钥来解密密文文件。

加密和解密的 API

您可以调用以下API，在本地对数据进行加解密。

API名称	说明
创建数据密钥	创建数据密钥。
解密数据密钥	用指定的主密钥解密数据密钥。

加密本地文件

1. 通过华为云控制台，创建用户主密钥，请参见创建密钥。

2. 请准备基础认证信息。
   - ACCESS_KEY: 华为云账号Access Key
   - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
   - PROJECT_ID: 华为云局点项目ID，请参见**华为云局点项目**。
   - KMS_ENDPOINT: 华为云KMS服务访问终端地址，请参见**终端节点**。

   说明

   华为云SDK，提供统一的SDK使用方式。通过添加依赖或下载的方式调用华为云API，访问华为云应用、资源和数据。若您需要，请参见**华为云SDK**。

3. 加密本地文件。

   示例代码中：
   - 用户主密钥：华为云控制台创建的密钥ID。
   - 明文数据文件：FirstPlainFile.jpg。
   - 输出的密文数据文件：SecondEncryptFile.jpg。

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.kms.v1.KmsClient;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequestBody;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyResponse;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequestBody;

import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.security.SecureRandom;

/**
 * 使用数据密钥（DEK）进行文件加解密
 * 激活assert语法，请在VM_OPTIONS中添加-ea
 */
public class FileStreamEncryptionExample {

    private static final String ACCESS_KEY = "<AccessKey>";
    private static final String SECRET_ACCESS_KEY = "<SecretAccessKey>";
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String KMS_ENDPOINT = "<KmsEndpoint>";

    // KMS服务接口版本信息，当前固定为v1.0
    private static final String KMS_INTERFACE_VERSION = "v1.0";

    /**
     * AES算法相关标识：
     * - AES_KEY_BIT_LENGTH: AES256密钥比特长度
     * - AES_KEY_BYTE_LENGTH: AES256密钥字节长度
     * - AES_ALG: AES256算法，本例分组模式使用GCM，填充使用PKCS5Padding
     * - AES_FLAG: AES算法标识
     * - GCM_TAG_LENGTH: GCM TAG长度
     * - GCM_IV_LENGTH: GCM 初始向量长度
     */
    private static final String AES_KEY_BIT_LENGTH = "256";
    private static final String AES_KEY_BYTE_LENGTH = "32";
    private static final String AES_ALG = "AES/GCM/PKCS5Padding";
    private static final String AES_FLAG = "AES";
    private static final int GCM_TAG_LENGTH = 16;
```



```
    private static final int GCM_IV_LENGTH = 12;

    public static void main(final String[] args) {
        // 您在华为云控制台创建的用户主密钥ID
        final String keyId = args[0];

        encryptFile(keyId);
    }

    /**
     * 使用数据密钥加解密文件实例
     *
     * @param keyId 用户主密钥ID
     */
    static void encryptFile(String keyId) {
        // 1.准备访问华为云的认证信息
        final BasicCredentials auth = new
BasicCredentials().withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);

        // 2.初始化SDK，传入认证信息及KMS访问终端地址
        final KmsClient kmsClient =
KmsClient.newBuilder().withCredential(auth).withEndpoint(KMS_ENDPOINT).build();

        // 3.组装创建数据密钥请求信息
        final CreateDatakeyRequest createDatakeyRequest = new
CreateDatakeyRequest().withVersionId(KMS_INTERFACE_VERSION)
                .withBody(new
CreateDatakeyRequestBody().withKeyId(keyId).withDatakeyLength(AES_KEY_BIT_LENGTH));

        // 4.创建数据密钥
        final CreateDatakeyResponse createDatakeyResponse =
kmsClient.createDatakey(createDatakeyRequest);

        // 5.接收创建的数据密钥信息
        // 密文密钥与KeyId建议保存在本地，方便解密数据时获取明文密钥
        // 明文密钥在创建后立即使用，使用前需要将16进制明文密钥转换成byte数组
        final String cipherText = createDatakeyResponse.getCipherText();
        final byte[] plainKey = hexToBytes(createDatakeyResponse.getPlainText());

        // 6.准备待加密的文件
        // inFile 待加密的原文件
        // outEncryptFile 加密后的文件

        final File inFile = new File("FirstPlainFile.jpg");
        final File outEncryptFile = new File("SecondEncryptFile.jpg");

        // 7.使用AES算法进行加密时，可以创建初始向量
        final byte[] iv = new byte[GCM_IV_LENGTH];
        final SecureRandom secureRandom = new SecureRandom();
        secureRandom.nextBytes(iv);

        // 8.对文件进行加密，并存储加密后的文件
        doFileFinal(Cipher.ENCRYPT_MODE, inFile, outEncryptFile, plainKey, iv);

    }

    /**
     * 对文件进行加解密
     *
     * @param cipherMode 加密模式，可选值为Cipher.ENCRYPT_MODE或者
Cipher.DECRYPT_MODE
     * @param infile     待加解密的文件
     * @param outFile    加解密后的文件
     * @param keyPlain   明文密钥
     * @param iv         初始化向量
     */
    static void doFileFinal(int cipherMode, File infile, File outFile, byte[] keyPlain, byte[] iv) {
```

```
        try (BufferedInputStream bis = new BufferedInputStream(new FileInputStream(infile));
             BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(outFile))) {
            final byte[] bytIn = new byte[(int) infile.length()];
            final int fileLength = bis.read(bytIn);

            assert fileLength > 0;

            final SecretKeySpec secretKeySpec = new SecretKeySpec(keyPlain, AES_FLAG);
            final Cipher cipher = Cipher.getInstance(AES_ALG);
            final GCMParameterSpec gcmParameterSpec = new
GCMParameterSpec(GCM_TAG_LENGTH * Byte.SIZE, iv);
            cipher.init(cipherMode, secretKeySpec, gcmParameterSpec);
            final byte[] bytOut = cipher.doFinal(bytIn);
            bos.write(bytOut);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

}
```

解密本地文件

1. 请准备基础认证信息。
   - ACCESS_KEY: 华为云账号Access Key
   - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
   - PROJECT_ID: 华为云局点项目ID，请参见**华为云局点项目**。
   - KMS_ENDPOINT: 华为云KMS服务访问终端地址，请参见**终端节点**。

   说明

   华为云SDK，提供统一的SDK使用方式。通过添加依赖或下载的方式调用华为云API，访问华为云应用、资源和数据。若您需要，请参见**华为云SDK**。

2. 解密本地文件。

   示例代码中：
   - 用户主密钥：华为云控制台创建的密钥ID。
   - 输出的密文数据文件：SecondEncryptFile.jpg。
   - 加密后再解密的数据文件：ThirdDecryptFile.jpg。

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.kms.v1.KmsClient;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyRequestBody;
import com.huaweicloud.sdk.kms.v1.model.CreateDatakeyResponse;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequest;
import com.huaweicloud.sdk.kms.v1.model.DecryptDatakeyRequestBody;

import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.security.SecureRandom;

/**
 * 使用数据密钥（DEK）进行文件加解密
 * 激活assert语法，请在VM_OPTIONS中添加-ea
```

```
*/
public class FileStreamEncryptionExample {

    private static final String ACCESS_KEY = "<AccessKey>";
    private static final String SECRET_ACCESS_KEY = "<SecretAccessKey>";
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String KMS_ENDPOINT = "<KmsEndpoint>";

    // KMS服务接口版本信息，当前固定为v1.0
    private static final String KMS_INTERFACE_VERSION = "v1.0";

    /**
     * AES算法相关标识：
     * - AES_KEY_BIT_LENGTH: AES256密钥比特长度
     * - AES_KEY_BYTE_LENGTH: AES256密钥字节长度
     * - AES_ALG: AES256算法，本例分组模式使用GCM，填充使用PKCS5Padding
     * - AES_FLAG: AES算法标识
     * - GCM_TAG_LENGTH: GCM TAG长度
     * - GCM_IV_LENGTH: GCM 初始向量长度
     */
    private static final String AES_KEY_BIT_LENGTH = "256";
    private static final String AES_KEY_BYTE_LENGTH = "32";
    private static final String AES_ALG = "AES/GCM/PKCS5Padding";
    private static final String AES_FLAG = "AES";
    private static final int GCM_TAG_LENGTH = 16;
    private static final int GCM_IV_LENGTH = 12;

    public static void main(final String[] args) {
        // 您在华为云控制台创建的用户主密钥ID
        final String keyId = args[0];
        // 创建数据密钥时，响应的密文数据密钥
        final String cipherText = args[1];

        decryptFile(keyId, cipherText);
    }

    /**
     * 使用数据密钥加解密文件实例
     *
     * @param keyId 用户主密钥ID
     * @param cipherText 密文数据密钥
     */
    static void decryptFile(String keyId，String cipherText) {
        // 1.准备访问华为云的认证信息
        final BasicCredentials auth = new
BasicCredentials().withAk(ACCESS_KEY).withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);

        // 2.初始化SDK，传入认证信息及KMS访问终端地址
        final KmsClient kmsClient =
KmsClient.newBuilder().withCredential(auth).withEndpoint(KMS_ENDPOINT).build();

        // 3.准备待加密的文件
        // inFile 待加密的文件
        // outEncryptFile 加密后的文件
        // outDecryptFile 加密后再解密的文件
        final File inFile = new File("FirstPlainFile.jpg");
        final File outEncryptFile = new File("SecondEncryptFile.jpg");
        final File outDecryptFile = new File("ThirdDecryptFile.jpg");

        // 4.使用AES算法进行解密时，初始向量需要与加密时保持一致，此处仅为占位。
        final byte[] iv = new byte[GCM_IV_LENGTH];

        // 5.组装解密数据密钥的请求，其中cipherText为创建数据密钥时返回的密文数据密钥。
        final DecryptDatakeyRequest decryptDatakeyRequest = new DecryptDatakeyRequest()
                .withVersionId(KMS_INTERFACE_VERSION).withBody(new
DecryptDatakeyRequestBody()
                        .withKeyId(keyId).withCipherText(cipherText).withDatakeyCipherLength(AES_KEY
_BYTE_LENGTH));
```

```
        // 6.解密数据密钥，并对返回的16进制明文密钥换成byte数组
        final byte[] decryptDataKey =
hexToBytes(kmsClient.decryptDatakey(decryptDatakeyRequest).getDataKey());

        // 7.对文件进行解密，并存储解密后的文件
        // 句末的iv为加密示例中创建的初始向量
        doFileFinal(Cipher.DECRYPT_MODE, outEncryptFile, outDecryptFile, decryptDataKey, iv);

        // 8.比对原文件和加密后再解密的文件
        assert getMD5(inFile).equals(getMD5(outDecryptFile));

    }

    /**
     * 对文件进行加解密
     *
     * @param cipherMode 加密模式，可选值为Cipher.ENCRYPT_MODE或者
Cipher.DECRYPT_MODE
     * @param infile     待加解密的文件
     * @param outFile    加解密后的文件
     * @param keyPlain   明文密钥
     * @param iv         初始化向量
     */
    static void doFileFinal(int cipherMode, File infile, File outFile, byte[] keyPlain, byte[] iv) {

        try (BufferedInputStream bis = new BufferedInputStream(new FileInputStream(infile));
             BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(outFile))) {
            final byte[] bytIn = new byte[(int) infile.length()];
            final int fileLength = bis.read(bytIn);

            assert fileLength > 0;

            final SecretKeySpec secretKeySpec = new SecretKeySpec(keyPlain, AES_FLAG);
            final Cipher cipher = Cipher.getInstance(AES_ALG);
            final GCMParameterSpec gcmParameterSpec = new
GCMParameterSpec(GCM_TAG_LENGTH * Byte.SIZE, iv);
            cipher.init(cipherMode, secretKeySpec, gcmParameterSpec);
            final byte[] bytOut = cipher.doFinal(bytIn);
            bos.write(bytOut);
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

    /**
     * 十六进制字符串转byte数组
     *
     * @param hexString 十六进制字符串
     * @return byte数组
     */
    static byte[] hexToBytes(String hexString) {
        final int stringLength = hexString.length();
        assert stringLength > 0;
        final byte[] result = new byte[stringLength / 2];
        int j = 0;
        for (int i = 0; i < stringLength; i += 2) {
            result[j++] = (byte) Integer.parseInt(hexString.substring(i, i + 2), 16);
        }
        return result;
    }

    /**
     * 计算文件SHA256摘要
     *
     * @param file 文件
     * @return SHA256摘要
     */
```

```
    static String getFileSha256Sum(File file) {
        int length;
        MessageDigest sha256;
        byte[] buffer = new byte[1024];
        try {
            sha256 = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e.getMessage());
        }
        try (FileInputStream inputStream = new FileInputStream(file)) {
            while ((length = inputStream.read(buffer)) != -1) {
                sha256.update(buffer, 0, length);
            }
            return new BigInteger(1, sha256.digest()).toString(16);
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage());
        }
    }
}
```

3 云服务使用 KMS 加解密数据

3.1 概述

数据加密服务中的密钥管理服务（Key Management Service，KMS），是一种安全、可靠、简单易用的密钥托管服务，帮助您轻松创建和管理密钥，保护密钥的安全。

云服务与KMS集成后，您只需在决定加密云服务数据时，选择一个KMS管理的用户主密钥，就可以轻松使用您选择的用户主密钥加解密您存储在这些云服务内的数据。

您可以选择云服务自动通过KMS创建的默认密钥，也可以选择您通过KMS自行创建或导入的自定义密钥，详细请参见**默认密钥与自定义密钥的区别**。

表 3-1 使用 KMS 加密的云服务列表

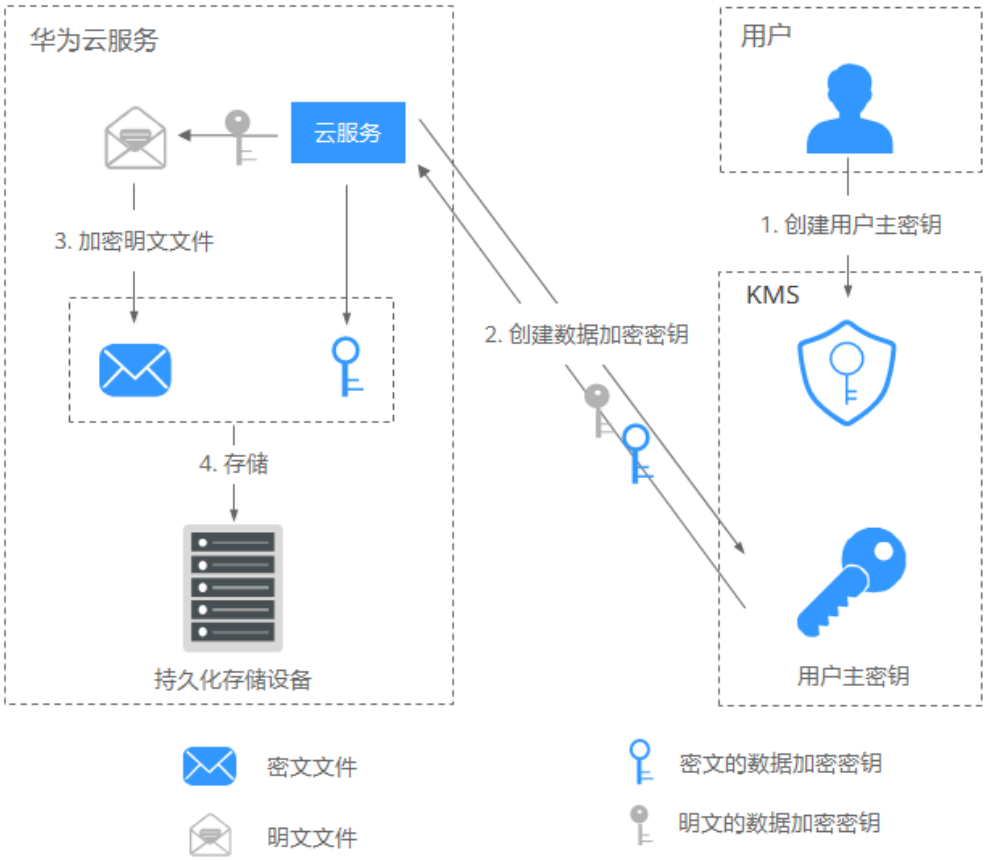
类型	服务	加密方式说明
计算	弹性云服务器ECS	弹性云服务器资源加密包括镜像加密和云硬盘加密。 • 在创建弹性云服务器时，您如果选择加密镜像，弹性云服务器的系统盘会自动开启加密功能，加密方式与镜像保持一致。 • 在创建弹性云服务器时，您也可以对添加的数据盘进行加密。
	镜像服务IMS	**IMS服务端加密**
存储	对象存储服务OBS	**OBS服务端加密**
	云硬盘EVS	**EVS服务端加密**
	云硬盘备份VBS	云硬盘备份主要对服务器中单个的云硬盘（系统盘和数据盘）创建在线备份，加密云硬盘的备份数据会以加密方式存放。

类型	服务	加密方式说明
	云服务器备份CSBS	云服务器备份主要对服务器下所有云硬盘创建一致性在线备份，云服务器备份产生的备份，会显示在云硬盘备份中。加密云硬盘的备份数据会以加密方式存放。
	弹性文件服务SFS	**SFS服务端加密**
数据库	云数据库MySQL	**RDS数据库加密**
	云数据库Postgre SQL	
	云数据库SQL Server	
	文档数据库服务DDS	**DDS数据库加密**
EI企业智能	数据仓库服务DWS	**DWS数据库加密**

原理介绍

华为云服务基于信封加密技术，通过调用KMS接口来加密云服务资源。由用户管理自己的用户主密钥，华为云服务在拥有用户授权的情况下，使用用户指定的用户主密钥对数据进行加密。

图 3-1 华为云服务使用 KMS 加密原理



加密流程说明如下：

1. 用户需要在KMS中创建一个用户主密钥。
2. 华为云服务调用KMS的“create-datakey”接口创建数据加密密钥。得到一个明文的数据加密密钥和一个密文的数据加密密钥。

   说明

   密文的数据加密密钥是由指定的用户主密钥加密明文的数据加密密钥生成的。
3. 华为云服务使用明文的数据加密密钥来加密明文文件，得到密文文件。
4. 华为云服务将密文的数据加密密钥和密文文件一同存储到持久化存储设备或服务中。

说明

用户通过华为云服务下载数据时，华为云服务通过KMS指定的用户主密钥对密文的数据加密密钥进行解密，并使用解密得到的明文的数据加密密钥来解密密文数据，然后将解密后的明文数据提供给用户下载。

3.2 OBS 服务端加密

简介

当您启用服务端加密功能后，在上传对象时，数据会在服务端加密成密文后存储。在下载加密对象时，存储的密文会先在服务端解密为明文，再提供给您。

KMS通过使用硬件安全模块（HSM）保护密钥安全的托管，帮助您轻松创建和管理加密密钥。您的密钥明文不会出现在HSM之外，避免密钥泄露。KMS对密钥的所有操作都会进行访问控制及日志跟踪，提供所有密钥的使用记录，满足监督和合规性要求。

需要上传的对象可以通过KMS提供密钥的方式进行服务端加密。您首先需要在KMS中创建密钥（或者使用KMS提供的默认主密钥），当您在OBS中上传对象时使用该密钥进行服务端加密。

使用服务端加密方式上传文件（控制台）

步骤1 在OBS管理控制台桶列表中，单击待操作的桶，进入“概览”页面。

步骤2 在左侧导航栏，单击“对象”。

步骤3 单击“上传对象”，系统弹出“上传对象”对话框。

步骤4 选择待上传的文件后，单击“打开”。

步骤5 勾选“KMS加密”，在后面的选择框中选择密钥名称，并单击“上传”，如图3-2所示。

图 3-2 加密上传对象

密钥名称：用户主密钥名称，是用户在数据加密服务中创建的密钥，主要用于加密保护数据。OBS会提供一个名为“obs/default”的默认密钥，用户可以选择使用默认密钥加密上传对象，也可以通过数据加密服务页面创建的自定义密钥加密上传对象。

步骤6 对象上传成功后，可在对象列表中查看对象的加密状态。

说明

- 对象的加密状态不可以修改。
- 使用中的密钥不可以删除，如果删除将导致加密对象不能下载。

----结束

使用服务端加密方式上传文件（API）

用户也可以通过调用OBS API接口，选择服务端加密SSE-KMS方式（SSE-KMS方式是指OBS使用KMS提供的密钥进行服务端加密）上传文件，详情请参考《对象存储服务API参考》。

3.3 EVS 服务端加密

简介

当您由于业务需求需要对存储在云硬盘的数据进行加密时，EVS为您提供加密功能，可以对新创建的云硬盘进行加密。加密云硬盘使用的密钥由数据加密服务（DEW，Data Encryption Workshop）中的密钥管理（KMS，Key Management Service）功能提供，无需您自行构建和维护密钥管理基础设施，安全便捷。

磁盘加密针对数据盘加密。系统盘的加密依赖于镜像，具体请参见**IMS服务端加密**。

哪些用户有权限使用云硬盘加密

- 安全管理员（拥有“Security Administrator”权限）可以直接授权EVS访问KMS，使用加密功能。

- 普通用户（没有“Security Administrator”权限）使用加密功能时，根据该普通用户是否为当前区域或者项目内第一个使用加密特性的用户，作如下区分：
  - 是，即该普通用户是当前区域或者项目内第一个使用加密功能的，需先联系安全管理员进行授权，然后再使用加密功能。
  - 否，即区域或者项目内的其他用户已经使用过加密功能，该普通用户可以直接使用加密功能。

对于一个租户而言，同一个区域内只要安全管理员成功授权EVS访问KMS，则该区域内的普通用户都可以直接使用加密功能。

如果当前区域内存在多个项目，则每个项目下都需要安全管理员执行授权操作。

云硬盘加密的密钥

加密云硬盘使用KMS提供的密钥，包括默认主密钥和用户主密钥 (CMK，Customer Master Key)：

- 默认主密钥：由EVS通过KMS自动创建的密钥，名称为“evs/default”。

  默认主密钥不支持禁用、计划删除等操作。
- 用户主密钥：由用户自己创建的密钥，您可以选择已有的密钥或者新创建密钥，具体请参见**创建密钥**。

使用用户主密钥加密云硬盘，若对用户主密钥执行禁用、计划删除等操作，将会导致云硬盘不可读写，甚至数据永远无法恢复，具体请参见**表3-2**。

表 3-2 用户主密钥不可用对加密云硬盘的影响

用户主密钥的状态	对加密云硬盘的影响	恢复方法
禁用	• 若加密云硬盘此时挂载至云服务器，则该云硬盘仍可以正常使用，但不保证一直可以正常读写。 • 若卸载加密云硬盘后，再重新挂载至云服务器将会失败。	启用用户主密钥，具体请参见**启用密钥**。
计划删除		取消删除用户主密钥，具体请参见**取消删除密钥**。
已经被删除		云硬盘数据永远无法恢复。

须知

用户主密钥为付费使用，若为按需计费的密钥，请及时充值确保帐户余额充足，若为包年/包月的密钥，请及时续费，以避免加密云硬盘不可读写导致业务中断，甚至数据永远无法恢复。

使用 KMS 加密云硬盘（控制台）

步骤1 在EVS管理控制台，单击“购买磁盘”。

步骤2 配置“加密”参数。

1. 展开“更多”，出现“加密”勾选框。

图 3-3 展开更多

2. 创建委托。

   勾选“加密”，如果当前未授权EVS访问KMS，则会弹出“创建委托”对话框，单击“是”，授权EVS访问KMS，当授权成功后，EVS可以获取KMS密钥用来加解密云硬盘。

   说明

   当您需要使用云硬盘加密功能时，需要授权EVS访问KMS。如果您有授权资格，则可直接授权。如果权限不足，需先联系拥有“Security Administrator”权限的用户授权，然后再重新操作。

3. 设置加密参数。

   勾选“加密”，若已经授权，会弹出“加密设置”对话框。

图 3-4 加密设置

密钥名称是密钥的标识，您可以通过“密钥名称”下拉框选择需要使用的密钥。您可以选择使用的密钥如下：

- 默认主密钥：成功授权EVS访问KMS，系统会创建默认主密钥“evs/default”。
- 用户主密钥：即您已有的密钥或者新创建密钥，具体请参见**创建密钥**。

步骤3 根据界面提示，配置云硬盘的其他基本信息。详细参数说明请参见**购买云硬盘**。

----结束

使用 KMS 加密云硬盘（API）

用户也可以通过调用EVS API接口购买加密磁盘，详情请参考《云硬盘API参考》。

3.4 IMS 服务端加密

用户可以采用加密方式创建私有镜像，确保镜像数据安全性。

约束条件

- 用户已启用数据加密服务。
- 加密镜像不能共享给其他用户。
- 加密镜像不能发布到应用超市。
- 如果云服务器的系统盘已加密，那么使用该云服务器创建的私有镜像也是加密的。
- 不能修改加密镜像使用的密钥。
- 加密镜像使用的密钥为禁用状态或者被删除时，该镜像无法使用。
- 对于加密镜像创建的弹性云服务器，其系统盘只能为加密状态，且磁盘密钥与镜像密钥一致。

使用 KMS 加密私有镜像（控制台）

创建加密镜像分为通过加密弹性云服务器创建加密镜像和通过外部镜像文件创建加密镜像。

- 通过加密弹性云服务器创建加密镜像

  用户选择弹性云服务器创建私有镜像时，如果该云服务器的系统盘已加密，那么使用该云服务器创建的私有镜像也是加密的。镜像加密使用的密钥为创建该系统盘时使用的密钥。

- 通过外部镜像文件创建加密镜像

  用户使用OBS桶中已上传的外部镜像文件创建私有镜像过程中，可以在注册镜像时勾选KMS加密完成镜像加密。

  用户上传镜像文件时，可以选择“KMS加密”，使用KMS提供的密钥来加密上传的文件，如图3-5所示。

  a. 在IMS管理控制台，单击“创建私有镜像”。
  b. “创建方式”选择“系统盘镜像”。
  c. “选择镜像源”为“镜像文件”。
  d. 勾选“KMS加密”。

  图 3-5 IMS 服务端加密

密钥名称是密钥的标识，您可以通过“密钥名称”下拉框选择需要使用的密钥。您可以选择使用的密钥如下：

- 默认主密钥：KMS为使用IMS（Image Management Service，IMS）的用户创建一个默认主密钥“ims/default”。
- 用户主密钥：即您已有的密钥或者新创建密钥，具体请参见**创建密钥**。

e. 根据界面提示，配置其他信息。详细参数说明请参见**注册镜像**。

使用 KMS 加密私有镜像（API）

用户也可以通过调用IMS API接口创建加密镜像，详情请参考《镜像服务API参考》。

3.5 SFS 服务端加密

简介

当您由于业务需求需要对存储在文件系统的数据进行加密时，弹性文件服务为您提供加密功能，可以对新创建的文件系统进行加密。

加密文件系统使用的是密钥管理服务（KMS）提供的密钥，无需您自行构建和维护密钥管理基础设施，安全便捷。当用户希望使用自己的密钥材料时，可通过KMS管理控制台的导入密钥功能创建密钥材料为空的用户主密钥，并将自己的密钥材料导入该用户主密钥中。具体操作请参见**导入密钥材料**。

当您需要使用文件系统加密功能时，创建SFS文件系统需要授权SFS访问KMS。若创建SFS Turbo文件系统，则不需要授权。

哪些用户有权限使用文件系统加密

- 安全管理员（拥有“Security Administrator”权限）可以直接授权SFS访问KMS，使用加密功能。
- 普通用户（没有“Security Administrator”权限）使用加密功能时，需要联系系统管理员获取安全管理员权限。

同一个区域内只要安全管理员成功授权SFS访问KMS，则该区域内的普通用户都可以直接使用加密功能。

如果当前区域内存在多个项目，则每个项目下都需要安全管理员执行授权操作。

文件系统加密的密钥

SFS加密文件系统使用KMS提供的密钥，包括默认主密钥和用户主密钥 (CMK，Customer Master Key)：

- 默认主密钥：系统会为您创建默认主密钥，名称为“sfs/default”。

  默认主密钥不支持禁用、计划删除等操作。
- 用户主密钥：即您已有的密钥或者新创建密钥，具体请参见**创建密钥**。

  如果加密文件系统使用的用户主密钥被执行禁用或计划删除操作，当操作生效后，使用该用户主密钥加密的文件系统仅可以在一段时间内（默认为60s）正常使用。请谨慎操作。

SFS Turbo文件系统无默认主密钥，可以使用您已有的密钥或者创建新的密钥，具体请参见**创建密钥**。

使用 KMS 加密文件系统（控制台）

用户通过弹性文件服务（Scalable File Service，SFS）创建文件系统时，可以选择“启用静态数据加密”，使用KMS提供的密钥来加密文件系统。

步骤1 在SFS管理控制台，单击“创建文件系统”。

步骤2 配置“加密”参数。

1. 创建委托。

   勾选“启用静态数据加密”，如果当前未授权SFS访问KMS，则会弹出“创建委托”对话框，单击“是”，授权SFS访问KMS，当授权成功后，SFS可以获取KMS密钥用来加解密文件系统。

   说明

   当您需要使用文件系统加密功能时，需要授权SFS访问KMS。如果您有授权资格，则可直接授权。如果权限不足，需先联系拥有“Security Administrator”权限的用户授权，然后再重新操作。

2. 设置加密参数。

   勾选“加密”，若已经授权，会弹出“加密设置”对话框。

   图 3-6 加密设置

   密钥名称是密钥的标识，您可以通过“密钥名称”下拉框选择需要使用的密钥。您可以选择使用的密钥如下：

   - 默认主密钥：成功授权SFS访问KMS，系统会创建默认主密钥“sfs/default”。
   - 自定义密钥：即您已有的密钥或者新创建密钥，具体请参见**创建密钥**。

步骤3 根据界面提示，配置云硬盘的其他基本信息。详细参数说明请参见**创建文件系统**。

----结束

使用 KMS 加密文件系统（API）

用户也可以通过调用SFS API接口创建加密的文件系统，详情请参考《弹性文件服务API参考》。

3.6 RDS 数据库加密

简介

关系型数据库支持MySQL、PostgreSQL、SQL Server引擎。

当启用加密功能后，用户创建数据库实例和扩容磁盘时，磁盘数据会在服务端加密成密文后存储。用户下载加密对象时，存储的密文会先在服务端解密为明文，再提供给用户。

约束条件

- 当前登录用户已通过统一身份认证服务添加华为云关系型数据库所在区域的KMS Administrator权限。权限添加方法请参见《统一身份认证服务用户指南》的“如何管理用户组并授权？”章节。
- 如果用户需要使用自定义密钥加密上传对象，则需要先通过数据加密服务创建密钥。使用数据加密服务创建密钥详情请参见**创建密钥**。
- 实例创建成功后，不可修改磁盘加密状态，且无法更改密钥。存放在对象存储服务上的备份数据不会被加密。
- 华为云关系型数据库实例创建成功后，请勿禁用或删除正在使用的密钥，否则会导致服务不可用，数据无法恢复。
- 选择磁盘加密的实例，新扩容的磁盘空间依然会使用原加密密钥进行加密。

使用 KMS 加密数据库实例（控制台）

用户在通过关系型数据库（Relational Database Service，RDS）购买数据库实例时，可以选择“磁盘加密”，使用KMS提供的密钥来加密数据库实例的磁盘，更多信息请参见**购买MySQL实例**、**购买PostgreSQL实例**、**购买SQL Server实例**。

图 3-7 RDS 服务端加密

使用 KMS 加密数据库实例（API）

用户也可以通过调用RDS API接口购买加密数据库实例，详情请参考《关系型数据库API参考》。

3.7 DDS 数据库加密

简介

当启用加密功能后，用户创建数据库实例和扩容磁盘时，磁盘数据会在服务端加密成密文后存储。用户下载加密对象时，存储的密文会先在服务端解密为明文，再提供给用户。

约束条件

- 已通过统一身份认证服务添加华为云文档数据库服务所在区域的KMS Administrator权限。权限添加方法请参见《统一身份认证服务用户指南》的“如何管理用户组并授权？”章节。
- 如果用户需要使用自定义密钥加密上传对象，则需要先通过数据加密服务创建密钥。使用数据加密服务创建密钥详情请参见**创建密钥**。
- 实例创建成功后，不可修改磁盘加密状态，且无法更改密钥。存放在对象存储服务上的备份数据不会被加密。
- 华为云文档数据库服务实例创建成功后，请勿禁用或删除正在使用的密钥，否则会导致数据库不可用，数据无法恢复。
- 选择磁盘加密的实例，新扩容的磁盘空间依然会使用原加密密钥进行加密。

使用 KMS 加密数据库实例（控制台）

用户在通过文档数据库服务（Document Database Service，DDS）购买数据库实例时，可以选择“磁盘加密”，使用KMS提供的密钥来加密数据库实例的磁盘，更多信息请参见**购买实例**。

图 3-8 DDS 服务端加密

使用 KMS 加密数据库实例（API）

用户也可以通过调用DDS API接口购买加密数据库实例，详情请参考《文档数据库服务API参考》。

3.8 DWS 数据库加密

简介

在DWS中，您可以为集群启用数据库加密，以保护静态数据。当您为集群启用加密时，该集群及其快照的数据都会得到加密处理。您可以在创建集群时启用加密。加密是集群的一项可选且不可变的设置。要从未加密的集群更改为加密集群(或反之)，必须从现有集群导出数据，然后在已启用数据库加密的新集群中重新导入这些数据。

如果希望加密，可以在集群创建时启用加密。虽然加密是DWS集群中的一项可选设置，但我们建议您为包含敏感数据的集群启用该设置。

哪些用户有权限使用 DWS 数据库加密

- 安全管理员（拥有“Security Administrator”权限）可以直接授权DWS访问KMS，使用加密功能。
- 普通用户（没有“Security Administrator”权限）使用加密功能时，根据该普通用户是否为当前区域或者项目内第一个使用加密特性的用户，作如下区分：
  - 是，即该普通用户是当前区域或者项目内第一个使用加密功能的，需先联系安全管理员进行授权，然后再使用加密功能。
  - 否，即区域或者项目内的其他用户已经使用过加密功能，该普通用户可以直接使用加密功能。

对于一个租户而言，同一个区域内只要安全管理员成功授权DWS访问KMS，则该区域内的普通用户都可以直接使用加密功能。

如果当前区域内存在多个项目，则每个项目下都需要安全管理员执行授权操作。

使用 KMS 加密 DWS 数据库流程

当选择KMS对DWS进行密钥管理时，加密密钥层次结构有三层。按层次结构顺序排列，这些密钥为主密钥（CMK）、集群加密密钥 (CEK)、数据库加密密钥 (DEK)。

主密钥用于给CEK加密，保存在KMS中。

CEK用于加密DEK，CEK明文保存在DWS集群内存中，密文保存在DWS服务中。

DEK用于加密数据库中的数据，DEK明文保存在DWS集群内存中，密文保存在DWS服务中。

密钥使用流程如下：

1. 用户选择主密钥。
2. DWS随机生成CEK和DEK明文。
3. KMS使用用户所选的主密钥加密CEK明文并将加密后的CEK密文导入到DWS服务中。
4. DWS使用CEK明文加密DEK明文并将加密后的DEK密文保存到DWS服务中。
5. DWS将DEK明文传递到集群中并加载到集群内存中。

当该集群重启时，集群会自动通过API向DWS请求DEK明文，DWS将CEK、DEK密文加载到集群内存中，再调用KMS使用主密钥CMK来解密CEK，并加载到集群内存中，最后用CEK明文解密DEK，并加载到集群内存中，返回给集群。

使用 KMS 加密 DWS 数据库（控制台）

步骤1 在DWS管理控制台，单击“购买数据仓库集群”。

步骤2 打开“加密数据库”开关。

1. 在“高级配置”中选择“自定义”，出现“加密数据库”开关。

图 3-9 加密数据库

 表示打开“加密数据库”开关，启用数据库加密。每个区域的每个项目首次启用数据库加密时，系统会弹出一个“创建委托”的对话框，单击“是”创建委托以授权DWS访问KMS，若单击“否”将不会启用加密功能。然后在“密钥名称”的下拉列表中选择已创建的密钥。如果没有密钥，可以登录KMS服务进行创建，详细操作请参见《数据加密服务用户指南》。

 表示关闭“加密数据库”开关，不启用数据库加密。

2. 创建委托。

   打开“加密数据库”开关，如果当前未授权DWS访问KMS，则会弹出“创建委托”对话框，单击“是”，授权DWS访问KMS，当授权成功后，DWS可以获取KMS密钥用来加解密云硬盘。

   说明

   当您需要使用数据库加密功能时，需要授权DWS访问KMS。如果您有授权资格，则可直接授权。如果权限不足，需先联系拥有“Security Administrator”权限的用户授权，然后再重新操作。

3. 加密设置。

   打开“加密数据库”开关，若已经授权，会弹出“加密设置”对话框。

图 3-10 加密设置

密钥名称是密钥的标识，您可以通过“密钥名称”下拉框选择需要使用的密钥。

步骤3 根据界面提示，配置其他基本信息。详细参数说明请参见**创建集群**。

----结束

4 应用程序使用凭据管理服务登录数据库

您在日常访问应用程序的过程中，通常会嵌入凭据直接访问程序。在需要更新凭据时，您除了创建新的凭据以外，还需要执行一些其他操作。您还需要花费一些时间更新应用程序以使用新的凭据。如果您有多个应用程序同一凭据，而您错过更新其中一个，则该应用程序就无法使用凭据登录。

因此使用方便有效且安全性高的凭据管理工具十分关键。

华为云凭据管理服务（Cloud Secret Management Service，CSMS）可以帮助您拥有以下优势：

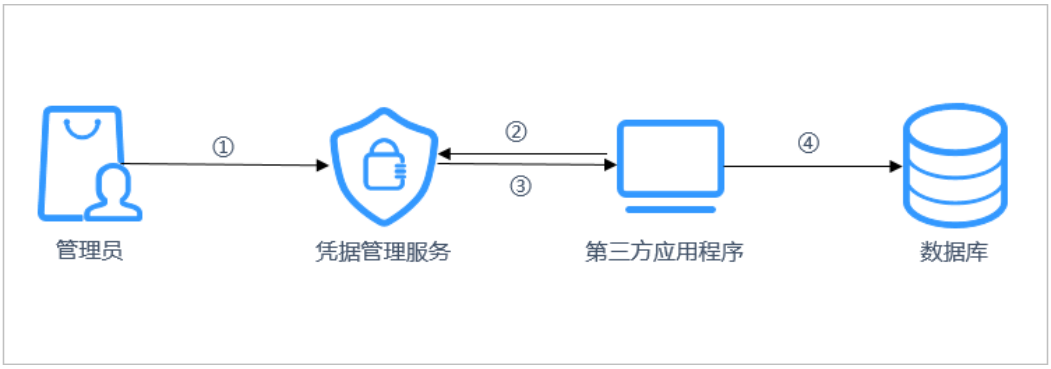
- 通过凭据托管功能，降低通过硬编码方式带来的凭据泄露风险，提高数据及资产的安全性。
- 减少对业务的影响：当您进行**人工轮换**的方式更新凭据时，您的业务不会受到影响。
- 提供安全的SDK接入方式，动态调用您的凭据。
- 凭据存储类型多样化。您不仅可以存储业务相关的账号密码，也可以存储业务数据库详细信息，包括但不限于：数据库名称、IP地址和端口等。

使用凭据登录数据库

下文为您介绍如何创建凭据，并且通过API调用凭据来登录到您的数据库。

您首先需要确保您的账号拥有KMS Administrator或者KMS CMKFullAccess权限，详情见**DEW权限管理**。

图 4-1 凭据登录流程



流程说明如下：

步骤1 您首先需要在凭据管理服务中使用**控制台**或者**API**创建一个凭据，用来存储数据库的相关信息（例如：数据库地址、端口、密码）。

步骤2 当您使用应用程序访问数据库时，凭据管理服务会去查询**步骤1**所创建的凭据存储的内容。

步骤3 凭据管理服务检索并解密凭据密文，将凭据中保存的信息通过凭据管理API安全地返回到应用程序中。

步骤4 应用程序获取到解密后的凭据明文信息，使用这些安全的信息访问数据库。

----结束

创建凭据和查询凭据 API

您可以调用以下API，通过API创建凭据保存相关内容并且查询相关凭据信息。

API名称	说明
创建凭据	创建新的凭据，并且将凭据值存入凭据的初始版本
查询凭据	查询指定凭据的信息

通过 API 接口创建凭据和查询凭据

1. 请准备基础认证信息：
   - ACCESS_KEY: 华为云账号Access Key
   - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
   - PROJECT_ID: 华为云局点项目ID，请参见**华为云局点项目**。
   - CSMS_ENDPOINT: 华为云CSMS服务访问终端地址，请参见**终端节点**。
2. 创建和查询凭据信息：
   凭据名称：secretName
   凭据值：secretString
   凭据版本值：LATEST_SECRET
   凭据版本：versionId

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

public class CsmsCreateSecretExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - PROJECT_ID: 华为云局点项目ID 详情见https://support.huaweicloud.com/productdesc-iam/
iam_01_0023.html
     * - CSMS_ENDPOINT: 华为云CSMS服务访问终端地址 详情见https://support.huaweicloud.com/api-dew/
```



```
dew_02_0052.html
     */
    private static final String ACCESS_KEY = "<AccessKey>";
    private static final String SECRET_ACCESS_KEY = "<SecretAccessKey>";
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";

    // 用来查询凭据最新版本详情的版本Id
    private static final String LATEST_SECRET = "latest";

    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];

        // 创建凭据
        createSecret(secretName, secretString);

        // 通过凭据版本值“latest”或者版本值“v1"查询到新创建的凭据内容
        ShowSecretVersionResponse latestVersion = showSecretVersion(secretName, LATEST_SECRET);
        ShowSecretVersionResponse firstVersion = showSecretVersion(secretName, "v1");

        assert latestVersion.equals(firstVersion);
        assert latestVersion.getVersion().getSecretString().equalsIgnoreCase(secretString);
    }

    /**
     * 创建凭据
     * @param secretName
     * @param secretString
     */
    private static void createSecret(String secretName, String secretString) {
        CreateSecretRequest secret = new CreateSecretRequest().withBody(
                new CreateSecretRequestBody().withName(secretName).withSecretString(secretString));

        CsmsClient csmsClient = getCsmsClient();

        CreateSecretResponse createdSecret = csmsClient.createSecret(secret);

        System.out.printf("Created secret success, secret detail:%s", createdSecret);
    }
    /**
     * 根据凭据版本id查询凭据版本详情
     * @param secretName
     * @param versionId
     * @return
     */
    private static ShowSecretVersionResponse showSecretVersion(String secretName, String versionId) {
        ShowSecretVersionRequest showSecretVersionRequest = new
ShowSecretVersionRequest().withSecretName(secretName)
                .withVersionId(versionId);

        CsmsClient csmsClient = getCsmsClient();

        ShowSecretVersionResponse version = csmsClient.showSecretVersion(showSecretVersionRequest);

        System.out.printf("Query secret success. version id:%s",
version.getVersion().getVersionMetadata().getId());

        return version;
    }

    /**
     * 获取CSMS服务客户端
     * @return
     */
    private static CsmsClient getCsmsClient() {
        BasicCredentials auth = new BasicCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
```



```
            .withProjectId(PROJECT_ID);

        return CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).build();
    }
}
```

使用应用程序获取数据库账号和密码

1. 获取凭据管理服务的CSMS SDK的依赖声明。

   示例如下：

   ```
   <dependency>
               <groupId>mysql</groupId>
               <artifactId>mysql-connector-java</artifactId>
               <version>XXX</version>
           </dependency>
           <dependency>
               <groupId>com.google.code.gson</groupId>
               <artifactId>gson</artifactId>
               <version>2.8.9</version>
           </dependency>
           <dependency>
               <groupId>com.huaweicloud.sdk</groupId>
               <artifactId>huaweicloud-sdk-csms</artifactId>
               <version>3.0.79</version>
           </dependency>
   ```

2. 建立数据库链接并且获取账号密码：

   示例代码如下

   ```
   import com.google.gson.Gson;
   import com.google.gson.JsonObject;
   import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
   import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

   import java.sql.Connection;
   import java.sql.DriverManager;
   import java.sql.SQLException;

    // 通过凭据信息获取指定的数据库账号和口令
       public static Connection getMySQLConnectionBySecret(String secretName, String jdbcUrl) throws
   ClassNotFoundException, SQLException{
           Class.forName(MYSQL_JDBC_DRIVER);
           ShowSecretVersionResponse latestVersionValue = getCsmsClient().showSecretVersion(new
   ShowSecretVersionRequest().withSecretName(secretName).withVersionId("latest"));
           String secretString = latestVersionValue.getVersion().getSecretString();
           JsonObject jsonObject = new Gson().fromJson(secretString, JsonObject.class);
           return DriverManager.getConnection(jdbcUrl, jsonObject.get("username").getAsString(),
   jsonObject.get("password").getAsString());
       }
   ```

5 如何轮换凭据

5.1 单用户凭据轮换策略

简介

单用户凭据轮换是指一个凭据中更新一个用户所保存的信息。

这是最基础的凭据轮换策略，适用于大多数日常使用场景。

例如：

- 访问数据库。凭据轮换时不会删除数据库连接，轮换后的新连接使用新凭据。
- 访问允许用户创建一个用户帐户的服务，例如以电子邮件地址作为用户名。服务通常允许用户根据需要经常更改密码，但用户无法创建其他用户或更改用户名。
- 根据需要创建的用户，称为临时用户。
- 以交互方式输入密码的用户，而不是让应用程序以编程方式从凭据管理服务中检索密码。这种类型的用户不需要更改他们的用户名和密码。

约束条件

您首先需要确保您的账号拥有KMS Administrator或者KMS CMKFullAccess权限，详情见**DEW权限管理**。

轮换凭据的 API

您可以调用以下API，在本地进行凭据轮换。

API名称	说明
创建凭据版本	创建新的凭据版本。
查询凭据版本与凭据值	查询指定凭据版本的信息和版本中的明文凭据值。

单账号凭据轮换代码示例

1. 通过华为云控制台，创建一个凭据，详情见**创建凭据**。
2. 请准备基础认证信息。
   - ACCESS_KEY: 华为云账号Access Key
   - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
   - PROJECT_ID: 华为云局点项目ID，请参见**华为云局点项目**。
   - CSMS_ENDPOINT: 华为云CSMS服务访问终端地址，请参见**终端节点**。
3. 进行单用户凭据轮换。

   示例代码中：
   - 凭据名称：华为云控制台创建的凭据名称。
   - 凭据内容：华为云控制台创建的凭据中所保存的值。
   - 凭据版本Id：华为云控制台创建凭据后自行生成凭据ID。
   - 凭据版本：LATEST_VERSION

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;

public class CsmsSingleAccountExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - PROJECT_ID: 华为云局点项目ID 详情见https://support.huaweicloud.com/productdesc-iam/
iam_01_0023.html
     * - CSMS_ENDPOINT: 华为云KMS服务访问终端地址 详情见https://support.huaweicloud.com/
api-dew/dew_02_0052.html
     */
    private static final String ACCESS_KEY = "<AccessKey>";
    private static final String SECRET_ACCESS_KEY = "<SecretAccessKey>";
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";
    // 用来查询凭据最新版本详情的版本Id
    private static final String LATEST_VERSION = "latest";
    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];
        singleAccountRotation(secretName, secretString);
    }

    /**
     * 凭据轮换-单账号模式代码示例
     *
     * @param secretName   凭据名称
     * @param secretString 凭据内容
     */
    private static void singleAccountRotation(String secretName, String secretString) {
        // 将新版凭据托管到凭据管理服务
        createNewSecretVersion(secretName, secretString);
        // 通过凭据版本“latest”查询到最新版本凭据
        ShowSecretVersionResponse secretResponseByLatest =
showSecretVersionDetail(secretName, LATEST_VERSION);
        assert secretResponseByLatest.getVersion().getSecretString().equals(secretString);
    }

    /**
     * 创建凭据示例
```



```
    * 使用不附带版本状态的方式添加凭据版本，则程序默认将SYSCURRENT版本状态指向当前新创建的版本
    *
    * @param secretName   凭据名称
    * @param secretString 凭据内容
    * @return
    */
   private static CreateSecretVersionResponse createNewSecretVersion(String secretName, String secretString) {
       CsmsClient csmsClient = getCsmsClient();

       CreateSecretVersionRequest createSecretVersionRequest = new CreateSecretVersionRequest()
               .withSecretName(secretName)
               .withBody(new CreateSecretVersionRequestBody().withSecretString(secretString));

       CreateSecretVersionResponse secretVersion = csmsClient.createSecretVersion(createSecretVersionRequest);

       System.out.printf("Created new version success, version id:%s", secretVersion.getVersionMetadata().getId());
       return secretVersion;
   }
   /**
    * 查询指定版本凭据
    *
    * @param secretName 查询凭据名称
    * @param versionId  查询凭据版本Id
    * @return
    */
   private static ShowSecretVersionResponse showSecretVersionDetail(String secretName, String versionId) {
       ShowSecretVersionRequest showSecretVersionRequest = new ShowSecretVersionRequest().withSecretName(secretName)
               .withVersionId(versionId);
       CsmsClient csmsClient = getCsmsClient();
       ShowSecretVersionResponse secretDetail = csmsClient.showSecretVersion(showSecretVersionRequest);
       System.out.printf("Query latest version success. version id:%s", secretDetail.getVersion().getVersionMetadata().getId());
       return secretDetail;
   }
   /**
    * 获取CSMS服务客户端
    *
    * @return
    */
   private static CsmsClient getCsmsClient() {
       BasicCredentials auth = new BasicCredentials()
               .withAk(ACCESS_KEY)
               .withSk(SECRET_ACCESS_KEY)
               .withProjectId(PROJECT_ID);
       return CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).build();
   }
}
```

5.2 双用户凭据轮换策略

简介

双用户轮换策略是指在一个凭据中更新两个用户所保存的信息。

例如：您的应用程序需要保持高可用性，如果您使用单用户轮换，那么您在修改用户密码和更新凭据内容的过程中会出现访问应用失败的情况，这样您就需要使用双用户凭据轮换策略。

您需要有一个账号例如Admin有权限创建并修改user1、user2用户的账号密码，首先您创建一个凭据创建并保存user1的账号与密码，记为user1/password1。通过新增凭据版本v2创建user2，并且保存user2的账号密码，记为user2/password2。修改user1的密码后，您会需要新增凭据版本v3，记为user1/password3。比如在您轮换创建新的凭据版本v3时，应用程序会继续使用现有的凭据版本v2获取信息。一旦凭据版本v3准备就绪，轮换会切换暂存标签，以便应用程序使用凭据版本v3获取信息。

约束条件

您首先需要确保您的账号拥有KMS Administrator或者KMS CMKFullAccess权限，详情见**DEW权限管理**。

轮换凭据的 API

您可以调用以下API，在本地进行凭据轮换。

API名称	说明
创建凭据版本	创建新的凭据版本。
查询凭据版本与凭据值	查询指定凭据版本的信息和版本中的明文凭据值。
更新凭据版本状态	更新凭据的版本状态。
查询凭据版本状态	查询指定的凭据版本状态标记的版本信息。

双账号凭据轮换代码示例

1. 通过华为云控制台，使用管理员账号创建一个凭据，详情见**创建凭据**。
2. 请准备基础认证信息。
   - ACCESS_KEY: 华为云账号Access Key
   - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
   - PROJECT_ID: 华为云局点项目ID，请参见**华为云局点项目**。
   - CSMS_ENDPOINT: 华为云CSMS服务访问终端地址，请参见**终端节点**。
3. 进行单用户凭据轮换。

   示例代码中：
   - 凭据名称：华为云控制台创建的凭据名称。
   - 凭据内容：华为云控制台创建的凭据中所保存的值。
   - 凭据版本Id：华为云控制台创建凭据后自行生成凭据ID。
   - 凭据版本：LATEST_VERSION

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.csms.v1.CsmsClient;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionRequestBody;
import com.huaweicloud.sdk.csms.v1.model.CreateSecretVersionResponse;
import com.huaweicloud.sdk.csms.v1.model.ListSecretStageRequest;
import com.huaweicloud.sdk.csms.v1.model.ListSecretStageResponse;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionRequest;
import com.huaweicloud.sdk.csms.v1.model.ShowSecretVersionResponse;
```

```
import com.huaweicloud.sdk.csms.v1.model.UpdateSecretStageRequest;
import com.huaweicloud.sdk.csms.v1.model.UpdateSecretStageRequestBody;

import java.util.Collections;
import java.util.List;

public class CsmsDualAccountExample {
    /**
     * 基础认证信息：
     * - ACCESS_KEY: 华为云账号Access Key
     * - SECRET_ACCESS_KEY: 华为云账号Secret Access Key
     * - PROJECT_ID: 华为云局点项目ID 详情见https://support.huaweicloud.com/productdesc-iam/
iam_01_0023.html
     * - CSMS_ENDPOINT: 华为云KMS服务访问终端地址 详情见https://support.huaweicloud.com/
api-dew/dew_02_0052.html
     */
    private static final String ACCESS_KEY = "<AccessKey>";
    private static final String SECRET_ACCESS_KEY = "<SecretAccessKey>";
    private static final String PROJECT_ID = "<ProjectID>";
    private static final String CSMS_ENDPOINT = "<CsmsEndpoint>";

    // 用来查询凭据最新版本详情的版本Id
    private static final String LATEST_VERSION = "latest";
    private static final String HW_CURRENT_STAGE = "SYSCURRENT";
    private static final String HW_PENDING_STAGE = "HW_PENDING";

    public static void main(String[] args) {
        String secretName = args[0];
        String secretString = args[1];

        dualAccountRotation(secretName, secretString);
    }

    /**
     * 凭据轮换-双账号模式代码示例
     *
     * @param secretName
     * @param secretString
     */
    private static void dualAccountRotation(String secretName, String secretString) {
        // 创建带自定义版本状态的新版本凭据，假如凭据内容为服务A的账号密码。
        CreateSecretVersionResponse newSecretVersion =
createNewSecretVersionWithStage(secretName,
                secretString, Collections.singletonList(HW_PENDING_STAGE));
        String versionId = newSecretVersion.getVersionMetadata().getId();

        //  轮换前检查pending是否被修改
        ListSecretStageResponse listSecretStageResponse = showSecretStage(secretName,
HW_PENDING_STAGE);
        assert listSecretStageResponse.getStage().getVersionId().equals(versionId);

        // 在更新服务A的账号密码后，同步将新版本凭据的版本状态更新为SYSCURRENT，之前旧的凭
据仍存储在服务中，已无法通过latest访问。
        updateSecretStage(secretName, versionId, HW_CURRENT_STAGE);

        // 此时通过"latest"版本状态查询最新版本凭据，完成双账号凭据轮转。
        ShowSecretVersionResponse secretResponse = showVersionDetail(secretName,
LATEST_VERSION);

        assert secretResponse.getVersion().getSecretString().equals(secretString);
    }

    /**
     * 创建凭据示例
     * 使用自定义版本状态的方式添加凭据版本，程序不会将SYSCURRENT版本状态指向新版本凭据。
     *
     * @param secretName
     * @param newSecretVersionText
     * @param stageList
```



```
     * @return
     */
    private static CreateSecretVersionResponse createNewSecretVersionWithStage(String
secretName,
                                                                                String newSecretVersionText,
                                                                                List<String> stageList) {
        CsmsClient csmsClient = getCsmsClient();

        // 将新版凭据托管到凭据管理服务
        CreateSecretVersionRequest createSecretVersionRequest = new
CreateSecretVersionRequest()
                .withSecretName(secretName)
                .withBody(new CreateSecretVersionRequestBody()
                        .withSecretString(newSecretVersionText)
                        .withVersionStages(stageList));

        CreateSecretVersionResponse secretVersion =
csmsClient.createSecretVersion(createSecretVersionRequest);

        System.out.printf("Created new version success, version id: %s",
secretVersion.getVersionMetadata().getId());

        return secretVersion;
    }

    /**
     * 将传入的版本状态指向指定凭据版本
     * @param secretName
     * @param versionId
     * @param newStageName
     */
    private static void updateSecretStage(String secretName, String versionId, String
newStageName) {
        UpdateSecretStageRequest updateSecretStageRequest = new
UpdateSecretStageRequest().withSecretName(secretName)
                .withStageName(newStageName).withBody(new
UpdateSecretStageRequestBody().withVersionId(versionId));

        CsmsClient csmsClient = getCsmsClient();

        csmsClient.updateSecretStage(updateSecretStageRequest);

        System.out.printf("Version stage update success. version id:%s, new stage name:%s",
versionId, newStageName);
    }

    /**
     * 查询指定版本凭据
     * @param secretName
     * @param versionId
     * @return
     */
    private static ShowSecretVersionResponse showVersionDetail(String secretName, String
versionId) {
        ShowSecretVersionRequest showSecretVersionRequest = new
ShowSecretVersionRequest().withSecretName(secretName)
                .withVersionId(versionId);

        CsmsClient csmsClient = getCsmsClient();
        ShowSecretVersionResponse secretDetail =
csmsClient.showSecretVersion(showSecretVersionRequest);

        System.out.printf("Query latest version success. version id:%s",
 secretDetail.getVersion().getVersionMetadata().getId());
        return secretDetail;
    }

    /**
     * 查询指定版本状态详情
```



```
     * @param secretName
     * @param stageName
     * @return
     */
    private static ListSecretStageResponse showSecretStage(String secretName, String
stageName) {
        ShowSecretStageRequest showSecretStageRequest = new ShowSecretStageRequest()
                .withSecretName(secretName).withStageName(stageName);
        CsmsClient csmsClient = getCsmsClient();
        return csmsClient.showSecretStage(showSecretStageRequest);
    }

    /**
     * 获取CSMS服务客户端
     *
     * @return
     */
    private static CsmsClient getCsmsClient() {
        BasicCredentials auth = new BasicCredentials()
                .withAk(ACCESS_KEY)
                .withSk(SECRET_ACCESS_KEY)
                .withProjectId(PROJECT_ID);
        return
CsmsClient.newBuilder().withCredential(auth).withEndpoint(CSMS_ENDPOINT).build();
    }
}
```

A 修订记录

发布日期	修改说明
2022-03-11	第五次正式发布。 新增“应用程序使用凭据管理服务登录数据库”和“如何轮换凭据”。
2021-09-30	第四次正式发布。 新增“加解密大量数据”。
2020-04-03	第三次正式发布。 更新界面截图。
2019-07-30	第二次正式发布。 新增DWS数据库加密。
2019-07-02	第一次正式发布。

