# 凭据管理系统

# 最佳实践

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

# 文档目录

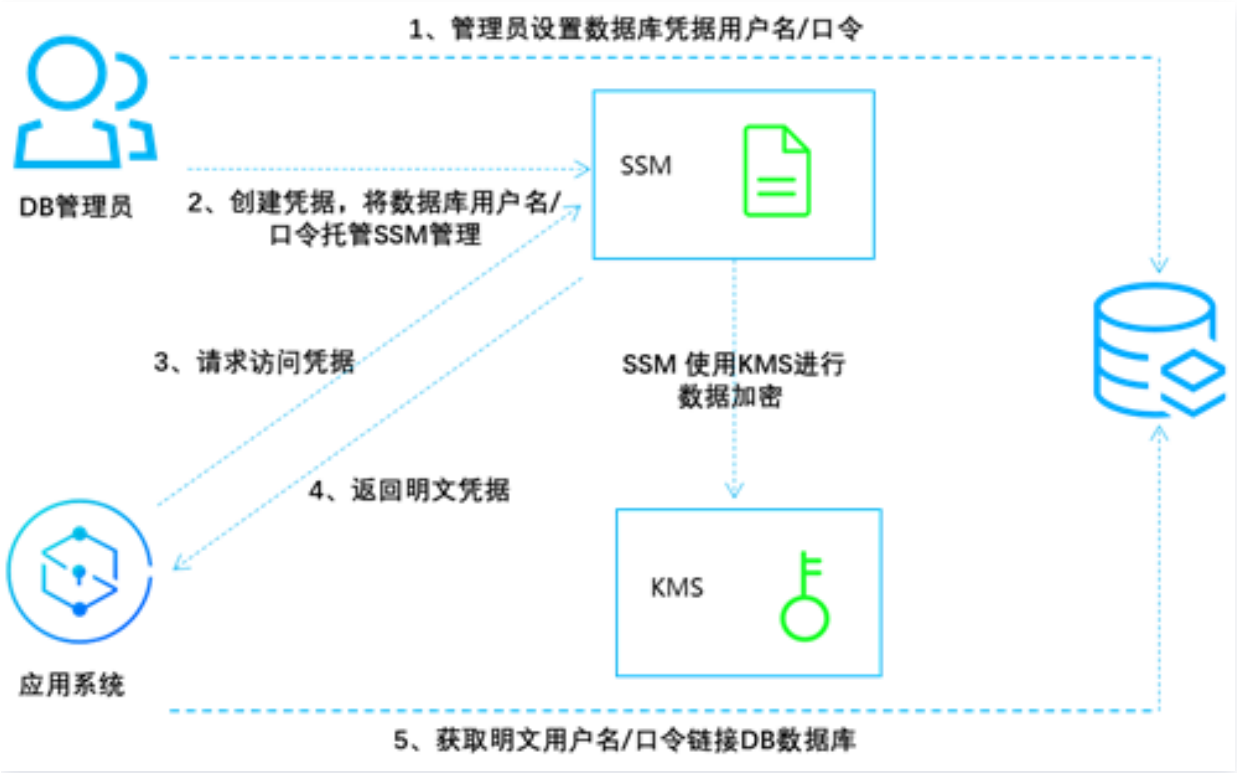
# 最佳实践
# 凭据托管和使用

最近更新时间：2023-09-26 14:21:44

应用程序或服务中，用于身份验证的各种认证信息，如口令、令牌、证书、SSH 密钥或 API 密钥等，通常情况下直接明文保存在应用程序的配置文件中，安全性较低。借助凭据管理系统将这些敏感认证信息加密存储，可有效避免敏感凭据明文编码带来的风险问题。

## 操作流程

以数据库用户名和口令的托管为例，介绍基本的凭据托管和使用场景。




1. DB 管理员设置数据库用户名和口令。
2. DB 管理员在凭据管理系统 SSM 中创建一个凭据对象，用来加密存储步骤1中的用户名和口令，关于创建凭据，详情请参见 创建凭据 。
3. 当应用系统需要访问数据库时，可向凭据管理系统 SSM 请求访问凭据，接口请求详情，请参见 获取凭据明文 。
4. 应用系统通过访问凭据接口所返回的内容，解析出明文凭据，获取到用户名和口令，从而访问该用户对应的目标数据库。
5. DB 管理员可为凭据创建多个版本内容，也可更新凭据版本内容，实现配置同步、版本管理及凭据轮换。

## 应用效果

对应用系统而言，通过调用 SSM 凭据管理系统的 API 或 SDK 来获取敏感的凭据明文，可避免在程序或配置中，明文编码凭据带来的信息泄露风险，调用对比如下：

- 使用本地存储数据库连接信息，连接信息明文保存在本地配置或者代码文件中，敏感凭据易泄露。
  - 获取凭据明文示例代码：

```
func GetDBConfig() string {
    dbConnStr := "user:password@tcp(127.0.0.1:3306)/test"
    return dbConnStr
}
```

  - 使用凭据明文示例代码：

```
conn, err := sql.Open("mysql", GetDBConfig())
if err != nil {
    // error handler
}
```

- 使用 SSM 凭据管理系统连接数据库 DB 时，代码和本地配置中无需明文存储 DB 的连接信息。
  - 获取凭据明文示例代码：

```
func GetDBConfig(secretName, version *string) string {
    credential := common.NewCredential(
            secretId,
            secretKey,
    )
    cpf := profile.NewClientProfile()
    cpf.HttpProfile.Endpoint = endpoint
    client, _ := ssm.NewClient(credential, region, cpf)

    request := ssm.NewGetSecretValueRequest()
    request.SecretName = secretName
    request.VersionId = version

    resp, err := client.GetSecretValue(request)
    if err != nil {
            // error handler
    }
    return *resp.Response.SecretString
}
```

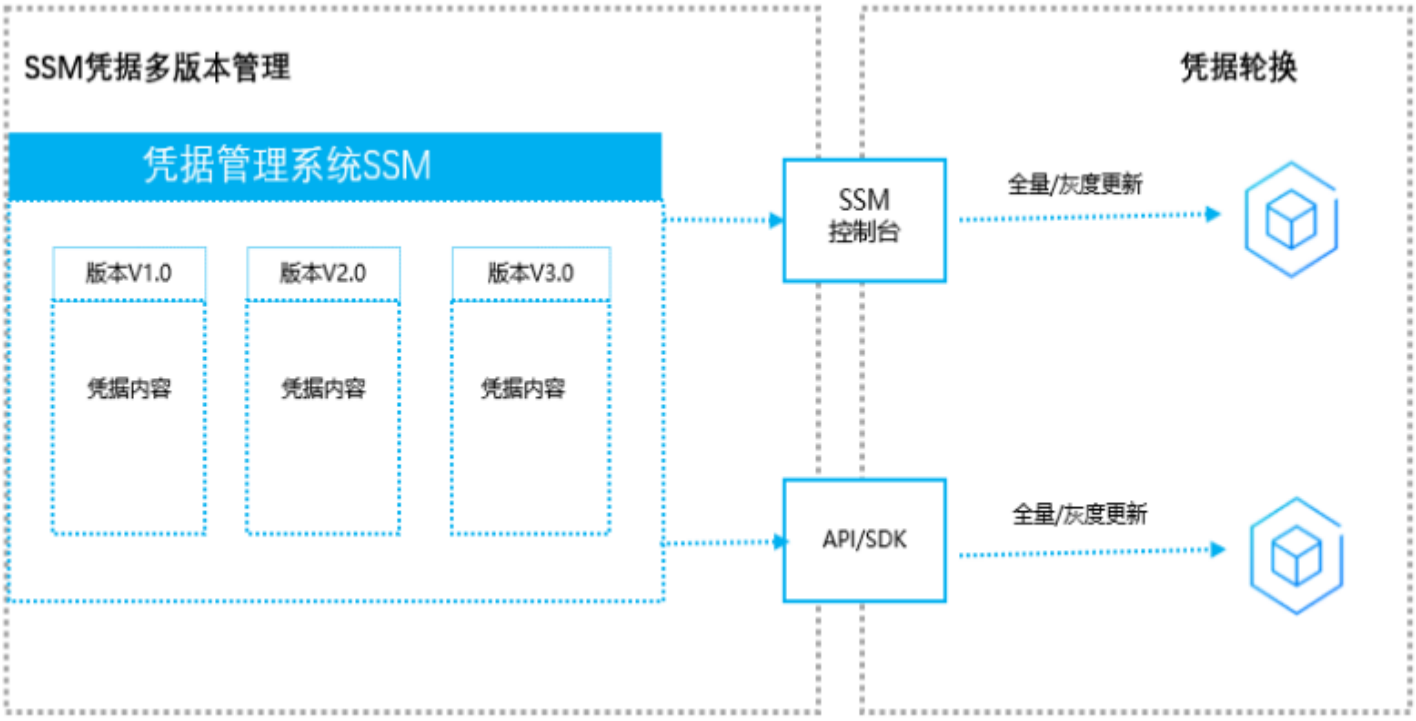
  - 使用凭据明文示例代码：

```
secretName := "MySecret1"
version := "MyVersion1"
conn, err := sql.Open("mysql", GetDBConfig(&secretName, &version))
if err != nil {
    // error handler
}
```

# 自定义凭据的轮换

最近更新时间：2023-09-26 14:21:44

为提升系统安全性，要求对目标凭据具备依赖性的应用配置同步更新。当多种应用系统在本地存储凭据内容时，在凭据更新时容易遗漏，从而带来应用中断风险。使用凭据管理系统，可以实现凭据管理系统内一处凭据更新，处处生效。此外，还可以为 凭据创建配置多个版本 ，实现凭据的灰度更新及轮换。




可以使用以下两种方式进行凭据轮换：

- **方式一**：增加新的凭据版本，业务侧通过更新获取凭据的版本号实现灰度轮换，详情可参见 多版本管理 。


- **方式二**：直接修改当前使用凭据的内容，业务侧下次调用接口获取凭据时，会自动更新凭据内容，详情可以参见 凭据相关调用示例 。

# 数据库凭据的应用

最近更新时间：2023-09-26 14:21:44

## 使用场景

在应用程序的开发过程中，数据库是常用于存储业务数据的基础服务。

- 为了能够对数据库的访问权限进行有效的控制，只有当应用程序在获得合适权限的账号和密码时，才能够创建数据库连接。
- 同时为了降低账号和密码泄露的风险，应用程序需要定期更换使用的账号和密码信息。

凭据管理系统（以下简称为 SSM）能够完美的适用于上述场景，避免了应用程序频繁更换账号密码的麻烦，并保证业务数据的安全。

## SSM SDK 功能特性

基于 SSM 提供的数据库凭据功能，以及数据库账号安全管理的最佳实践，腾讯云团队开发了一套应用程序集成的 SSM SDK。
应用程序通过调用 SSM SDK，传入数据库连接所需的参数和凭据名称，可以获取到一个可用的数据库连接，不必关心账号密码的获取和轮转的实现细节。

## 支持语言

- Golang 1.13 及以上版本。
- Python 2.7 及以上版本, Python 3 及以上版本。

## 风险提示

SSM 在对数据库凭据进行周期性轮转的时候，会更新账号和密码。
请严格按照 **SDK 使用说明**来使用 SSM SDK。**不要在除了SSM SDK内部逻辑之外的任何地方缓存获取到的数据库连接，也不要缓存获取到的凭据中的任何信息**，以避免账号密码失效导致的数据库连接失败情况的发生。

## 前提条件

1. 确认已开通 KMS 服务，SSM 基于密钥管理系统 KMS 托管的密钥进行加密。
2. 在腾讯云平台上已购买至少一台云数据库实例（目前只支持 MySQL 实例），并完成数据库的初始化，同时创建了至少一个database。如未创建 MySQL 实例，请参见 创建MySQL实例 。
3. 在 SSM 控制台中已创建一个数据库凭据，并已关联指定数据库。如未创建数据库凭据，请参见 创建数据库凭据 。
4. 在 访问管理（CAM）控制台 中，需完成以下两点：
   - 已创建能够访问 SSM 凭据资源和 MySQL 实例资源的子账号。
   - 给子账号分配了 API 密钥，为了便于获取 SecretId 和 SecretKey 用于 API 的访问。

# SDK使用说明

> **说明**
> 可直接下载 SDK 源码，根据源码中提供的 demo 代码来使用。

## Golang-sdk

1. 获取 go 模块：

（1）运行命令：go get github.com/tencentcloud/ssm-rotation-sdk-golang/lib
（2）在golang项目中引入：

```
import (
  "github.com/tencentcloud/ssm-rotation-sdk-golang/lib/db"
  "github.com/tencentcloud/ssm-rotation-sdk-golang/lib/ssm"
)
```

2. 初始化 DynamicSecretRotationDb 对象

```
dbConn = &db.DynamicSecretRotationDb{}
err := dbConn.Init(&db.Config{
DbConfig: &db.DbConfig{
MaxOpenConns:       100,
MaxIdleConns:       50,
IdleTimeoutSeconds:  100,
ReadTimeoutSeconds:  5,
WriteTimeoutSeconds: 5,
SecretName:         "test",         // 凭据名
IpAddress:          "127.0.0.1",    // 数据库实例的访问地址
Port:               58366,          // 数据库实例的访问端口
DbName:             "database_name", // 指定具体的数据库名，如果为空，则只连接到数据库实例，不连接具体的数据库
ParamStr:           "charset=utf8&loc=Local", // 数据库连接相关的配置参数
},
SsmServiceConfig: &ssm.SsmAccount{
SecretId:  os.Getenv("secret_id"),     // 子账号的SecretId
SecretKey: os.Getenv("secret_key"),    // 子账号的SecretKey
Region:    "ap-guangzhou", // 选择凭据实际所存储的地域，请根据实际情况填写，示例中给出的是广州地区。
},
WatchChangeInterval: time.Second * 10, // 监控凭据内容发生变化的间隔，一般间隔时间设置为10秒~60秒之间为宜
})
```

3. 获取数据库连接

```golang
c := dbConn.GetConn() // 警告：每次需要访问数据库时，都需要调用GetConn()来获取最新的DB连接，请不要在业务代码中缓存此对象，以免DB访问失败！
if err := c.Ping(); err != nil { // 示例中只是调用ping()方法测试账号密码的可用性。实际业务中，这里就可以执行具体的db操作了。
    log.Fatal("failed to access db with err: ", err)
    return
}
```

4. 关闭数据库连接

```golang
c.Close() // 当应用程序退出时，可主动关闭数据库连接。这是个通用的操作，和数据库凭据没有直接关系
```

### [Python-sdk](https://github.com/TencentCloud/ssm-rotation-sdk-python)

1. 安装依赖包

```shell
$ pip install -r requirements.txt
```

2. 配置信息

```python
WATCH_FREQ = 10  # 监控凭据内容发生变化的间隔，一般间隔时间设置为10秒~60秒之间为宜
db_config = DbConfig(
params={
  'secret_name': "test",  # 凭据名
  'ip_address': "127.0.0.1",  # 数据库地址
  'port': 58366,  # 数据库端口
  'db_name': "database_name",  # 指定具体的数据库名，如果为空，则只连接到数据库实例，不连接具体的数据库
  'param_str': "charset=utf8&loc=Local",
})
ssm_service_config = SsmAccount(
```

```
params={
  'secret_id': os.getenv('SECRET_ID'),  # 需填写实际可用的SecretId
  'secret_key': os.getenv('SECRET_KEY'),  # 需填写实际可用的SecretKey
  'url': test_url,
  'region': "ap-guangzhou"  # 选择凭据所存储的地域
})
config = Config(
params={
  'db_config': db_config,
  'ssm_service_config': ssm_service_config,
  'WATCH_FREQ': WATCH_FREQ
})
```

3. 初始化数据库对象

```python
db_conn = DynamicSecretRotationDb()
err = db_conn.init(config)
```

4. 获取数据库连接

```python
c = db_conn.get_conn()  # 警告：每次需要访问数据库时，都需要调用GetConn()来获取最新的DB连接，请不要在业务代码中缓存此对象，以免DB访问失败！
try:
    c.ping()  # 示例中只是调用ping()方法测试账号密码的可用性。实际业务中，这里就可以执行具体的db操作了。
except TencentCloudSDKException as e:
    logging.error("failed to access db with err: {0}".format(str(
        e.args[0])).encode("utf-8"))
```

5. 关闭数据库连接

```python
c.close()  # 当应用程序退出时，可主动关闭数据库连接。这是个通用的操作，和数据库凭据没有直接关系
```

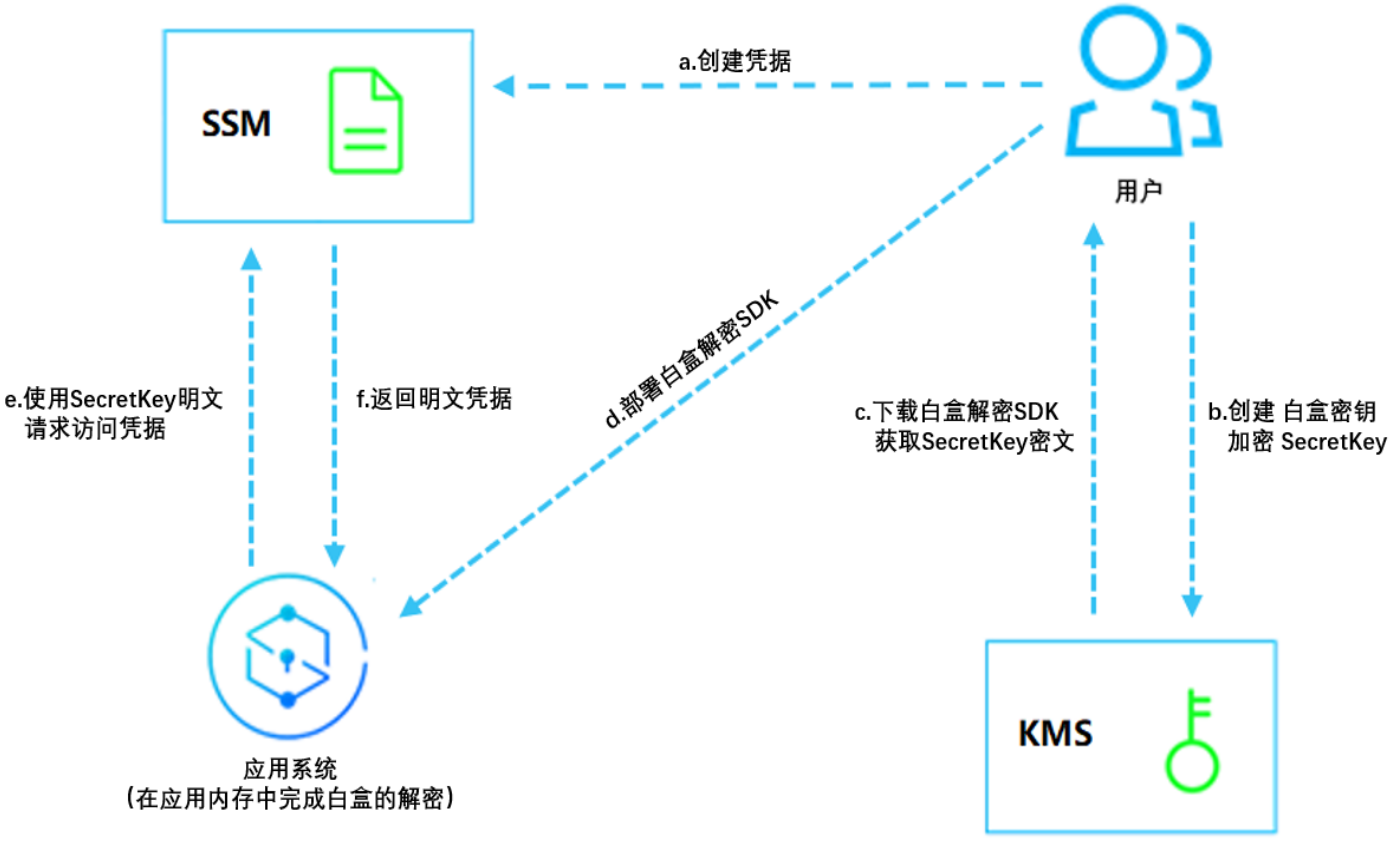
# 白盒密钥保护 SSM SecretKey

最近更新时间：2023-09-26 14:21:44

## 背景

调用凭据管理系统（SSM）服务时，将会使用到腾讯云账号的 SecretKey 作为身份的认证，其中大部分用户直接以 SecretKey 的明文形式存储，例如明文暴露在代码中、放置在配置中心、上传至 Git 等方式，风险系数极大。为了能保障端到端的全链路数据安全，防止 SecretKey 的明文泄露，可以结合密钥管理系统（KMS）的白盒密钥解决方案，以密文的形式来存储 SecretKey。
在白盒密钥管理解决方案中，将密钥和算法进行融合，有效的将密钥隐藏起来，大大增加密钥被嗅探、被破解的难度，从而保护密钥这一极其敏感的信息，提升端到端全链路的安全性。

## 操作流程




### 步骤1：创建 SSM 凭据对象

a. 用户在凭据管理系统 SSM 中创建一个凭据对象，关于创建凭据，详情请参见 创建凭据 。

### 步骤2：白盒密钥加密

b. 用户使用密钥管理系统 KMS 的白盒密钥对 SecretKey 进行加密，具体操作步骤如下所示。

i. 创建白盒密钥 。

ii. 控制台获取 API SecretKey 。
iii. 使用白盒密钥加密 API SecretKey 。

c. 加密完成后，获取对应的密文和初始化向量，并下载解密密钥和解密 SDK 文件。

> **说明**
> 白盒加密的整体操作，详情请参见 使用 KMS 白盒密钥保护 SecretKey 最佳实践 。

d. 部署白盒解密 SDK。

### 步骤3：应用系统需要访问 SSM，获取凭据明文

e. 首先在业务逻辑中调用白盒 SDK 的解密函数，获取到 SecretKey 的明文，接口详情请参见 白盒密钥解密代码示例 。
f. 应用系统通过白盒解密接口所返回的 SecretKey 明文，再向凭据管理系统 SSM 请求访问凭据，接口详情请参见 获取凭据明文 。

> **注意**
> 白盒解密的整个过程是在本地内存中运行的，不依赖于网络和外部的服务。