SM2加密说明

算法参数说明

1. 密钥对生成

- 使用SM2默认曲线 sm2p256v1
- SM2推荐曲线参数（来自https://github.com/ZZMarquis/gmhelper）
- 对私钥进行压缩，公钥不压缩

2. 加密模式

- 使用C1C2C3模式
- 加密结果byte[]转Hex字符串

公钥配置说明

前往新版管理后台配置、查看SM2公钥

请求参数加密说明

注：使用保利威SM2公钥加密

请求参数加密开启

请求头携带参数**x-e-type**:2，值2表示SM2加密

GET加密请求参数规则

暂不支持GET请求参数加密

POST JSON请求参数加密规则

1. 将body中JSON报文整体使用保利威公钥SM2加密
2. POST请求body为加密后的密文

POST FORM表单请求参数加密规则

注：签名相关参数appId,timestamp,sign不参与加密

1. 原始参数生成签名sign
2. 将签名检验参数appId,sign,timestamp除外的其他业务参数按照JSON格式封装

```
{
    "channelId": 2376543,
    "pageNumber": 1,
    "pageSize": 10
}
```

3. 将JSON报文使用保利威公钥SM2加密的到密文ABCDEF

```
SM2Util.encrypt(respEncryptKey, SM2Engine.Mode.C1C2C3, xparam)
```

4. 剔除加密前业务参数，表单增加加密字段xparam=ABCDEF

```
http://api.polyv.net/xxx?appId=xxx&sign=xxx&timestamp=xxx&xparam=ABCDEF
```

加密参数请求响应说明

1. 请求参数解密成功时，正常响应接口对应业务码
2. 请求参数解密失败时，http响应码为400，错误业务码10001

```
{
    "code": 400,
    "status": "error",
    "error": {
        "code": 10001,
        "desc": "参数错误"
    },
    "success": false
}
```

响应报文加密说明

注：用户生成SM2公私钥对（自行保管好私钥），在保利威平台配置的SM2公钥，开启响应加密后，会对响应报文中data字段进行加密

响应参数加密开启

请求参数携带参数**encryptResponseType**:2，值2表示SM2加密

响应报文加密规则

1. 未配置密钥或配置错误密钥，响应结果为明文不加密
2. 只对响应码200，业务成功响应时，JSON中data字段进行加密
3. 响应码非200，业务失败时，error为明文不加密
4. 用户使用对应SM2密钥对响应报文进行解密
5. 加密报文响应示例

```
{
    "code": 200,
    "status": "success",
    "data": "043D1BF16CB1C2629348A7B18F9C51036C9466F3E198F75C6ABDE83A428E2893",
    "success": true
}
```

SM2加解密算法示例代码

```
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>5.5.2</version>
</dependency>
<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcprov-jdk15on</artifactId>
    <version>1.70</version>
</dependency>
```

```
import java.nio.charset.StandardCharsets;
import org.bouncycastle.crypto.engines.SM2Engine;
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPublicKey;
import cn.hutool.crypto.BCUtil;
import cn.hutool.crypto.ECKeyUtil;
import cn.hutool.crypto.SmUtil;
import cn.hutool.crypto.asymmetric.KeyType;
import cn.hutool.crypto.asymmetric.SM2;

public class SM2Util {

    /**
     * 生成密钥对
     * @return
     */
    public static SM2KeyPair generateKeyPair() {
        SM2 sm2 = SmUtil.sm2();
        //这里会自动生成对应的随机秘钥对 ，注意！ 这里一定要强转，才能得到对应有效的秘钥信息
        byte[] privateKey = BCUtil.encodeECPrivateKey(sm2.getPrivateKey());
        //这里公钥不压缩  公钥的第一个字节用于表示是否压缩  可以不要
        byte[] publicKey = ((BCECPublicKey) sm2.getPublicKey()).getQ().getEncoded(false);
        SM2KeyPair sm2KeyPair = new SM2KeyPair();
        sm2KeyPair.setPrivateKey(Util.byteToHex(privateKey));
        sm2KeyPair.setPublicKey(Util.byteToHex(publicKey));
        return sm2KeyPair;
    }

    /**
     * 加密
     * @param publicKey
     * @param mode
     * @param text
     * @return byte2Hex
     */
    public static String encrypt(String publicKey, SM2Engine.Mode mode, String text) {
        SM2 sm2 = SmUtil.sm2();
        sm2.setPublicKeyParams(ECKeyUtil.toSm2PublicParams(publicKey));
        sm2.setMode(mode);
        byte[] encrypt = sm2.encrypt(text.getBytes(StandardCharsets.UTF_8), KeyType.PublicKey);
        return Util.byteToHex(encrypt);
    }

    /**
     * 解密
     * @param privateKey
     * @param mode SM2Engine.Mode
     * @param text hexString
     * @return
     */
    public static String decrypt(String privateKey, SM2Engine.Mode mode, String text) {
        SM2 sm2 = SmUtil.sm2();
        sm2.setPrivateKeyParams(ECKeyUtil.toSm2PrivateParams(privateKey));
        sm2.setMode(mode);
        byte[] encrypt = sm2.decrypt(Util.hexToByte(text), KeyType.PrivateKey);
        return new String(encrypt);
```

```
    }


}
```