加密通讯设计说明

|  |  |
| --- | --- |
| 版本编号 | 0.1.0 |
| 文档编号 |  |
| 发布者 | 系统组 |
| 发布日期 | 2016-4 |

# 概述

## 编制目的

当汇购访问外部系统时，如果对方没有提供安全通讯机制，请参照本手册使用。如对方已提供如https 等安全通讯，请按对方要求通讯。

## 内容概述

* 我方加密过程如下：

1. 随机生成一个AES加密的的Key (byte[])。
2. 将AES-key编码，encryptKey= base64（（使用渠道的公钥RSA算法，加密AES-Key））,为String
3. 做签名,sign=md5（appId+data+time+encryptKey+ 渠道的公钥）
4. 将业务对象变为json字符串，
5. 加密数据，data= （使用AES加密json字符串 ，再进行base64编码）

* 渠道接收请求后，进行上述步骤的逆过程：

1. 通过base64解码参数encryptKey，使用渠道的私钥RSA算法解密，得到AES-KEY
2. 验证签名，md5（appId+data+time+encryptKey +渠道公钥）的值是否和sign相等，不等则拒绝。
3. 对data进行BASE64解码，使用AES-KEY 对 其做AES解密，再得到业务数据的json字符串
4. 进行业务逻辑处理
5. 返回数据用AES-KEY 做AES加密，使用base64序列化成字符串

## 名称解释

## 参考文献

# 类说明



## 对请求加密

**类**： CommunicateUtil

**方法**： EncodedCommunicateData encryptRequest(CommunicateData data，String 公钥)

请求对象(未加密)

CommunicateData：

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **说明** |
| appId | string | 分配给汇积分应用的标识。  应以区分不同的应用 |
| data | string | 业务数据 |
| time | string | 时间的毫秒 |
| aesKey | string | aes 加密用到的key |

**返回**: 签名并加密后要发送的对象EncodedCommunicateData.

请求对象(加密)

EncodedCommunicateData：

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **说明** |
| appId | string | 分配给汇积分应用的标识。  应以区分不同的应用 |
| data | string | 加密后的业务数据 |
| time | string | 时间的毫秒 |
| sign | string | 签名项 |
| encryptKey | string | 加密后的aesKey |

## 对请求解密

**方法：**CommunicateData decyrptRequest(EncodedCommunicateData data，String 私钥)

**返回**: 解密后的对象CommunicateData.

## 验证签名

**方法：**boolean checkSign(EncodedCommunicateData data，String 公钥)

**返回**: 通过校验返回true

## 响应内容加密

**方法：String** encryptResponse(String aesKey，String content)

**返回**: 序列化成base64字串的加密结果；

注：这里的content是json字串形式响应内容，加密使用AES，密钥是请求传过来解密后的密钥

## 示例:

//加密请求

EncodedCommunicateData enCode = CommunicateUtil.encryptRequest(data, Base64Utils.Base64Encode(keyPairs.getPublicKey()));

//合作放校验签名

String deContent="";

if(CommunicateUtil.checkSign(enCode,

Base64Utils.Base64Encode(keyPairs.getPublicKey()))){

//校验成功 解密

CommunicateData deCode = CommunicateUtil.decyrptRequest(enCode, Base64Utils.Base64Encode(keyPairs.getPrivateKey()));

System.out.println("解密后的数据"+deCode);

String content = "{'status':'true','text':'处理成功'}";

System.out.println("处理业务，得到返回数据"+content);

//加密响应数据

deContent = CommunicateUtil.encryptResponse(deCode.aesKey, content);

}else{

System.out.println("校验签名失败");

}

//接受到响应数据，解密

try {

String enContent = new String(AESUtil.decryptAES(Base64Utils.Base64Decode(deContent), data.aesKey.getBytes("utf-8")));

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}