# 应用层加密解密方案

## 加密解密流程

### 客户端与服务端握手

1. 客户端向服务端申请公钥。
2. 服务端生成非对称加密密钥对，生成一个唯一标识作为Client Token，服务端缓存Client Token和密钥对。服务端向客户端返回公钥和Client Token。
3. 客户端生成对称加密密钥，密钥作为客户端的唯一标识（Client Identity），客户端存储Client Identity和Client Token。Client Identity使用公钥加密，客户端向服务端发送加密的Client Identity和Client Token
4. 服务端通过Client token获取密钥对，使用密钥解密Client Identity，服务端缓存Client Token和Client Identity的关系，服务端销毁Client Token和密钥对关系，销毁密钥对。
5. 客户端与服务端所有通信可以使用对称加密，密钥为Client Identity

### 用户与服务端握手

1. 用户登陆成功后，服务端生成对称加密密钥，密钥作为用户唯一标识（User Identity），服务端缓存Client Token和User Identity的关系。服务端使用Client Identity加密User Identity，服务端发送加密的User Identity到客户端。
2. 客户端解密得到User Identity，客户端存储User Identity
3. 客户端与服务端所有通信可以使用对称加密，密钥为User Identity

#### 名词解释

1. Client Token：客户端与服务端相处传递时的标识，用于隐藏真正的密钥，以及标识请求。
2. Client Identity：客户端标识，实质为对称加密密钥，由客户端生成，用于客户端与服务端请求的加密解密。
3. User Identity：用户标识，实质为对称加密密钥，由服务端生成，用于指定客户从客户端与服务端请求的加密解密；同时可以标识用户的登陆状态。

## 优缺点

#### 缺点

* 应用层做加密解密，对请求速度有影响，延迟程度取决于需要加密请求的数据大小（待测试）
* 框架复杂度变高，客户端框架需要实现对应的加密解密算法和流程
* 测试复杂度变高，服务端接收的时加密数据，解密需要对应的Client Identity / User Identity，测试时需要构建这些前提条件

#### 优点

* 更加安全，在每个客户端一个密钥的基础上可以做到每个用户一个密钥，所以即使某个客户端或者某个客户被攻破，不会影响到其他客户端或者用户。
* 算法替换更加灵活，加密解密由框架控制，不受证书影响

## 时序图

1. 客户端与服务端握手



1. 用户与服务端握手



## 默认加密解密算法

##### 对称加密

AES-256

##### 非对称加密

RSA-512

##### 签名算法

HMAC-SHA256 加密

RSA-512 数字签名