【详细设计】RTC鉴权机制设计

1 总体设计

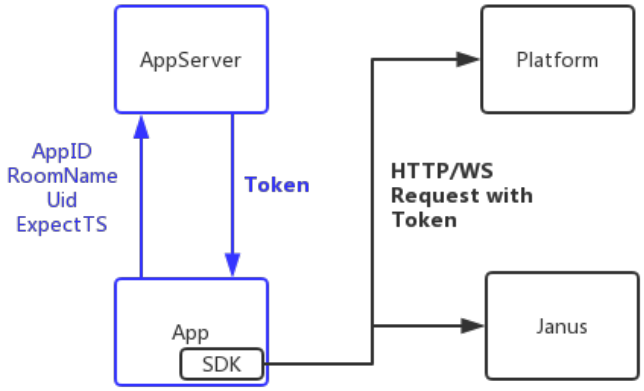
百度RTC设计中，包括两类用户和服务端的交互方式。

一类是用户的服务端直接调用RTC提供的OpenAPI服务，其服务一般包括房间管理，日志服务，统计服务等。此类交互鉴权使用基于AK/SK的百度云标准鉴权服务，具体鉴权方式可参考相应文档。

另外一类是，集成了百度RTC SDK的用户客户端与百度RTC服务端的交互，出于安全考虑，此类交互需要使用特殊的鉴权机制。

本文档主要包括第二种鉴权机制的设计。

1.1 鉴权架构



- AppID：由baiduRTC提供，全局唯一，用于识别用户；
- AppKey: 由baiduRTC提供，每个AppID拥有一个AppKey，并且可更新；
- Token生成算法: 由baiduRTC提供，用户需集成到自己的AppServer中；
- RoomName: 房间名称，由用户指定并维护；
- Uid: 用户id，由用户指定并维护，用户保证唯一性；
- ExpectTS: 过期时间，如果为0，则默认24小时；
- AppServer: 用户的app后台，用于根据app客户端请求，AppKey以及baiduRTC提供的Token生成算法，生成相应的Token，并下发到app中；

1.2 鉴权流程

图中，蓝色部分为需要用户实现部分，黑色部分为baiduRTC提供。

- 当App需要使用RTC服务时，向AppServer请求Token；
- AppServer根据token生成算法以及相应的AppKey生成相应的Token，并下发App；
- App在调用RTC SDK时，提供Token；
- SDK在于RTC后端服务器发起http/ws连接时提供Token参数；
- RTC后端对token进行验证，验证不通过时会拒绝访问；

2 详细设计

2.1 Token格式

Token的具体格式如下:

字段	说明	长度	示例
version	鉴权版本	3字符	004
signature	签名	40字符	8f50a1f280e69f4581dd8bf8b3b9cc9d277cd3a6
ts	token生成时的时间戳	10字符，不足补0	1544766061

randomString	salt string，int转16进制字符串	8字符，不足补0	dabdd97c
expectTs	过期时间，单位秒	10字符，不足在前面补0	0000000000

2.2 签名算法

使用HmacSHA1算法加密，key为AppKey，签名内容为:

signature= HmacSHA1(AppKey, "ACS" + appID + ts + randomString + roomName + uid + expectTs);

2.3 签名计算示例

appID:app-jcagj2g5ecrqv7bn

version: 004

ts:1553144847

randomString:dabdd97c

expectTs:0000000000

roomName：aaa

uid: 54321

AppKey:s-jcagj2g5eewai6u3p3

计算得signature: HamcSHA1(s-jcagj2g5eewai6u3p3,ACSapp-jcagj2g5ecrqv7bn1553144847dabdd97caaa5432100000000000)
= 07e27ce059e51501873b038f53af3e65bbbee23c

构造token为：00407e27ce059e51501873b038f53af3e65bbbee23c1553144847dabdd97c0000000000

token生成python示例：signature.py

```
import binascii
from optparse import OptionParser

def calculate_signature(app_key, app_id, ts, random_str, room_name, uid,
expect_ts):
    data = "ACS{}{}{}{}{}{}{}".format(app_id, ts, random_str, room_name,
uid, expect_ts)
    print "data to calculate signature:{}".format(data)
    signature = hmac.new(app_key, data, sha1).digest()
    signature = binascii.b2a_hex(signature)
    print "signature:{}".format(signature)
    return signature

def calculate_token(version, signature, ts, random_str, expect_ts):
    token = "{}{}{}{}{}{}".format(version, signature, ts, random_str,
expect_ts)
    print "token:{}".format(token)

if __name__ == "__main__":
    usage = " -- for example: ./signature.py -s app_key -a app_id -t ts -r
random_str -o room -u uid -e expect_ts"
    parser = OptionParser(usage)
    parser.add_option("-s", "--sk", dest="app_key", help="app key", action
= "store", type="string")
    parser.add_option("-a", "--aid", dest="app_id", help="app id", action
= "store", type="string")
    parser.add_option("-t", "--ts", dest="ts", help="ts", action =
"store", type="string")
    parser.add_option("-r", "--rs", dest="random_str", help="random_str",
action = "store", type="string")
    parser.add_option("-o", "--room", dest="room", help="random_str",
action = "store", type="string")
    parser.add_option("-u", "--uid", dest="uid", help="uid", action =
"store", type="string")
    parser.add_option("-e", "--ets", dest="expect_ts", help="expect_ts",
action = "store", type="string")
    (opt, args) = parser.parse_args()

    sig = calculate_signature(opt.app_key, opt.app_id, opt.ts, opt.
random_str, opt.room, opt.uid, opt.expect_ts)
    calculate_token("004", sig, opt.ts, opt.random_str, opt.expect_ts)
```

BRTC_TOKEN_signature.py

用法：

python signature.py −s s−jcagj2g5eewai6u3p3 −a app−jcagj2g5ecrqv7bn −t 1553144847 −r dabdd97c −o aaa −u 54321 −e 0000000000

输出：

data to calculate signature:ACSapp−jcagj2g5ecrqv7bn1553144847dabdd97caaa543210000000000
signature:07e27ce059e51501873b038f53af3e65bbbee23c
token:00407e27ce059e51501873b038f53af3e65bbbee23c1553144847dabdd97c0000000000

2.4 客户端请求示例

客户端通过websocket连接mediaserver时，需要提供**token**和**appid，roomname和uid四**个参数(均为小写)，mediaserver校验token中的签名，若验证失败，则返回401.

示例：

wss://10.145.80.147:8188/janus?appid=app-jcagj2g5ecrqv7bn&roomname=aaa&uid=54321&token=00407e27ce059e51501873b038f53af3e65bbbee23c1553144847dabdd97c0000000000

platform可以配置appid是否开启权限校验，如果不开启权限校验，token可以不传。appid和roomname以及uid是必须传入的参数。

3. token生成Java示例代码

需要引入The Apache Commons Codec package，maven：

```
<!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->

<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.12</version>
</dependency>
```

示例代码：

TokenUtil.java