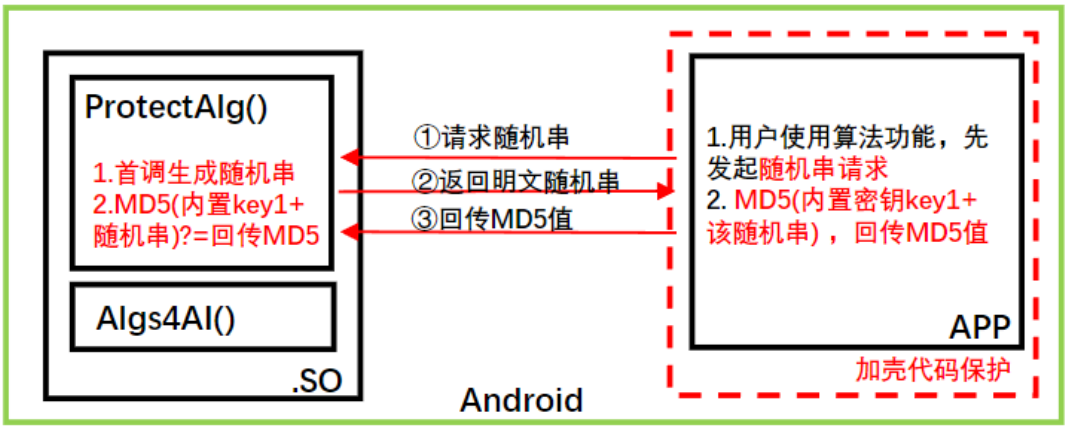
算法加解密流程说明

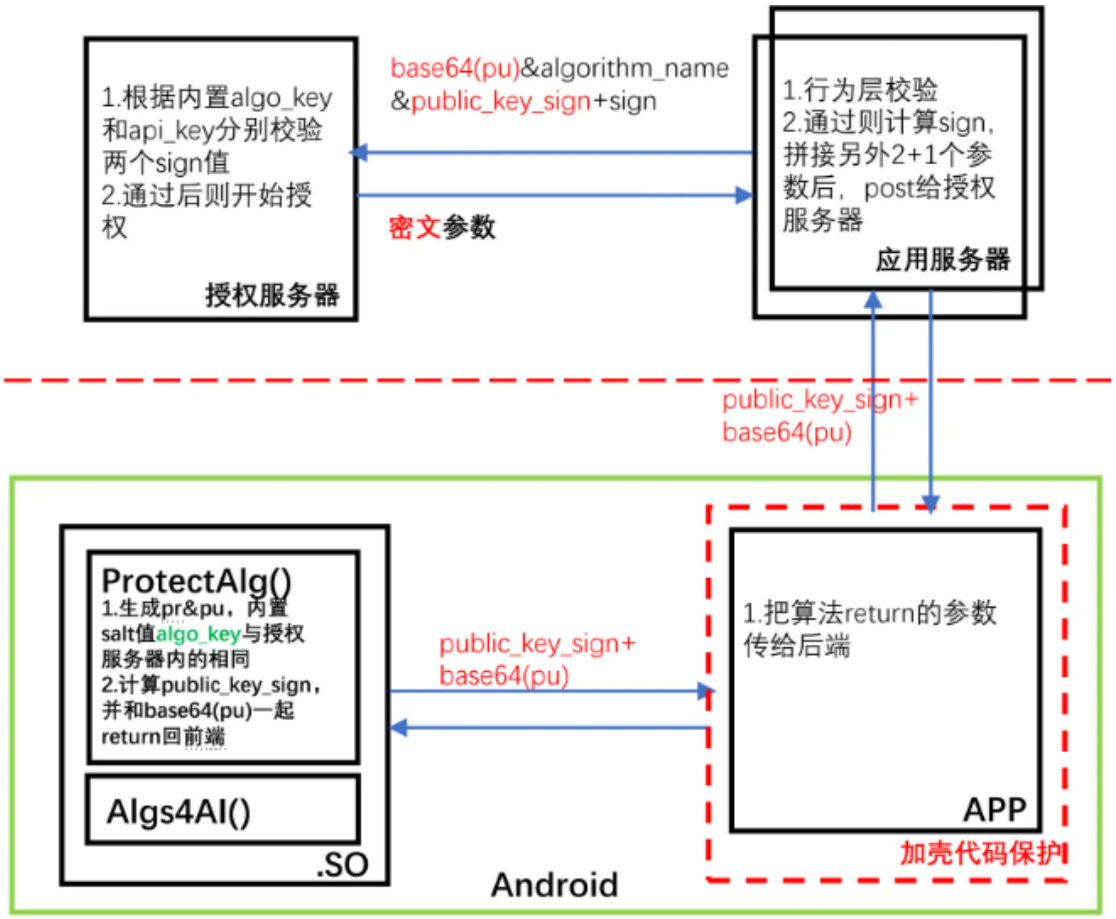
一、MD5 验证



步骤 1：安卓/ios 前端向算法端请求获得一段随机字符串；
步骤 2：算法和前端都用相同的内置 key 和步骤 1 的字符串生成 MD5（注意顺序）；
步骤 3：前端将自己的 MD5 传给算法，算法做比较，相同即通过验证。

算法端代码相关 messageDigestFive.cpp：
1、 随机字符串获取：`randStrGen()`；
2、 MD5 生成：`messageDigestFGet()`；
3、 字符串比较：`strCompare()`；

二、算法服务器验证



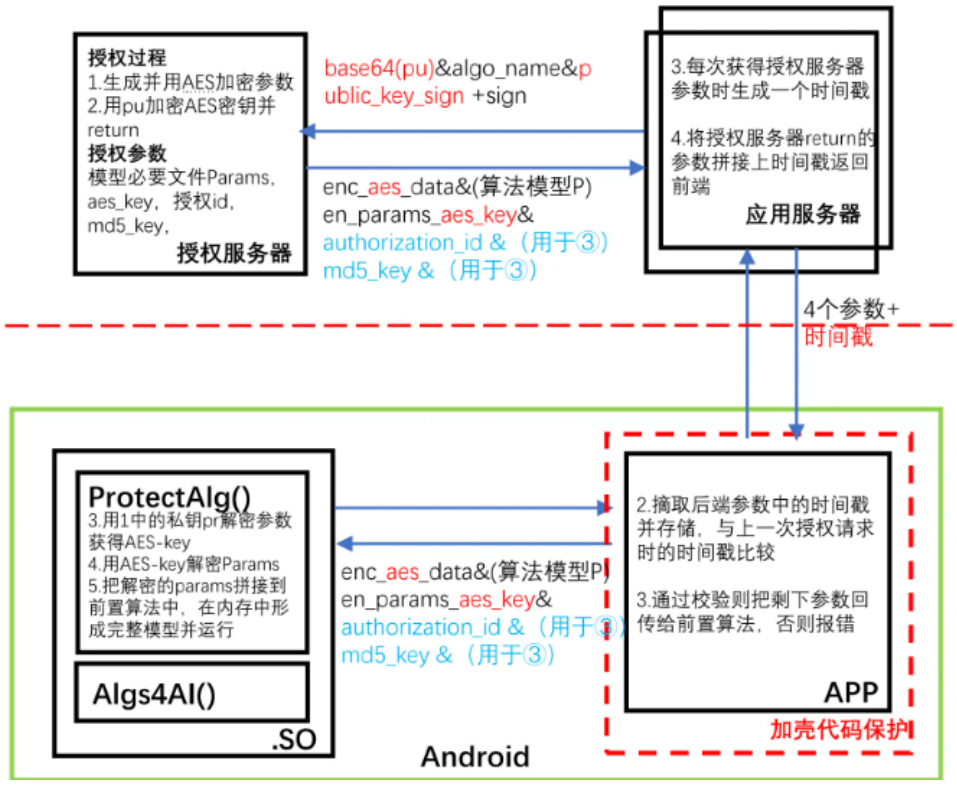
步骤 1：算法将模型扣出部分（model_p2）通过后端传给授权服务器，授权服务器会给模型对应的 salt_key；剩余的模型（model_p1）可自己选择方式加密后给安卓/ios 前端；

步骤 2：前端向算法请求获得加密的公钥，算法生成公私钥，将公钥先进行 base64 加密得到 base64_pu，将 base64_pu+“&”+步骤 1 得到的 salt_key 获得 MD5，将 base64_pu 和 MD5 返回给前端；

步骤 3：前端将 base64_pu 和 MD5 发送给授权服务器，授权服务器解密后认证。

算法端代码相关 rsa_ed.cpp base64_ed.cpp encode_decode.cpp

1、 获取加密后的公钥信息：_getPubkey()



步骤 4：如步骤 3 中认证通过，服务器会生成 aes 密钥加密 model_p2 和其他信息并用 base_64 再次加密生成 enc_aes_data，用步骤 2 中的公钥加密 aes 密钥并用 base_64 再次加密获得 en_params_aes_key，由前端传给算法端；

步骤 5：算法将 en_params_aes_key 进行 base_64 解密，用步骤 2 中的私钥再解密，获得 aes 密钥；

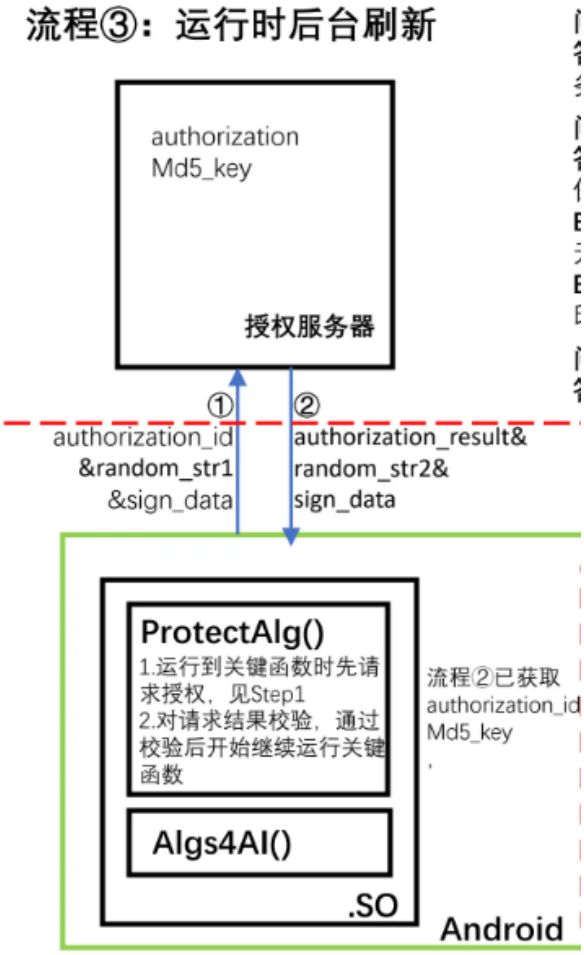
步骤 6：将 enc_aes_data 进行 base_64 解密，用步骤 5 中的 aes 密钥解密获得 model_p2，前端会传进 model_p1，按自己先前加密方式解密 model_p1 后与 model_p2 在内存中拼接在一起后使用；

算法端代码相关 encode_decode.cpp

1、 解密授权数据：decode_authorization();

2、 内存拼接模型：modelCombine();

三、后台 http 联网验证



步骤 1：在流程二的步骤 6 中，解密后的 enc_aes_data 还包含了两个信息：md5_key 和 authorization_id，算法生成随机字符串 random_str，加上字符“&“后同 md5_key 算 MD5 数值：sign_data，通过网络 http 链接将 random_str 和 sign_data 发送给授权服务器；

步骤 2：授权服务器会返回授权结果，包含了服务器生成的第二个随机字符串 random_str_2、授权结果 authorization_result（1 为通过 0 为不通过）、以及 random_str2&authorization_result&md5_key 算出来的第二个 sign_data2，算法端用自己的 random_str+”&”+authorization_result+”&”+md5_key 算出自己的 sign_data3，与 sign_data2 比较，如果 authorization_result 为 1 且两个 sign_data 相同则通过验证。同时算法保存 random_str2，每次授权时比较新 random_str2 与上一次的 random_str2，不同则通过验证。

算法端代码相关 myhttp.cpp（有用到 curl 库，详情参见链接 https://github.com/curl/curl）

1、 http 链接、发送及验证：`httpcheck()`；