## 前端与服务端数据通讯格式

1：前端与服务端通讯采用websocket长链接。通讯数据采用小端结构。

2：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Len | CRC | Ver | Sign | MainId | SubId | EncryptType |  | requestId | realSize | Data |
| 2B | 2B | 2B | 2B | 1B | 1B | 1B | 1B | 4B | 2B | N |
| 包长度 | 校验 | 版本 | 标记 | 主命令 | 子命令 | 加密类型 | 备用 | 请求ID | 原始数据包长度 | 加密数据 |

Ver：初始版本 0x0001

Sign：标记字段， 值为:0x5F5F

EncryptType：加密类型， 0x01 不加密 Json

0x02 不加密 protobuf

0x11 位运算 Json

0x12 位运算 protobuf

0x21 RSA Json

0x22 RSA protobuf

0x31 AES Json

0x32 AES protobuf

realSize：原始数据的长度，因为有时候Data长度不为16整数倍时，会填充原始数据，解密后会有填充数据在里面，probobuf解析时长度不对会解析不成功。

3：Data数据字段格式

此结构为protobuf结构

|  |  |
| --- | --- |
| Sign | Data |
| 4B | N |
| 标记 | 原始数据(相应的指令数据) |

Sign：标记，验证解密 Sign:0xF5F5F5F5

4：CRC 的值为Ver字段到Data字段的数值和。

计算如下：

uint16\_t GetCheckSum(void\* header, int size)

{

uint16\_t sum = 0;

uint16\_t \*p = (uint16\_t\*)header;

for(int i = 0; i < size / 2; ++i)

sum += \*p++;

if(size %2 )

sum+=\*(uint8\_t\*)p

return sum;

}