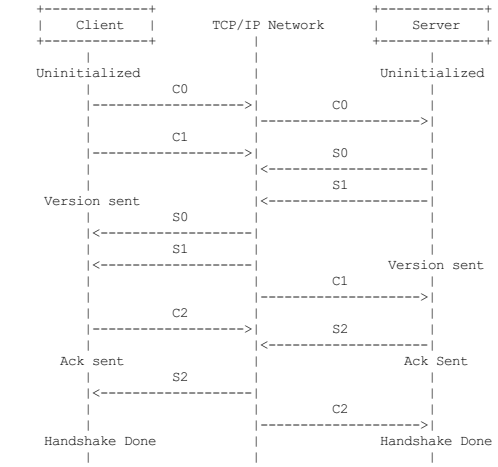
rtmp协议简介：简单握手和复杂握手

rtmp的底层连接（Tcp连接）连接上后，需要进行握手，握手成功后才能建立起rtmp连接。
rtmp客户端发送C0，C1，C2；rtmp服务端发送S0，S1，S2。
C0和S0为1个字节，剩下的C1，C2，S1，S2都是1536个字节。

握手消息的交互要遵循几个规则：

- 握手由客户端发送C0，C1消息发起
- 客户端必须收到S1后，才能发送C2
- 客户端收到S2后，才能发送其他数据
- 服务端必须收到C0或者C0，C1后，发送S0，S1
- 服务端发送完S0，S1后，可以接着发送S2
- 服务端必须收到C2后，才能发送其他数据

客户端与服务端的交互如下图：



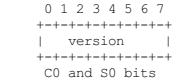
rtmp的握手协议有两种：简单握手和复杂握手。

不同的握手协议，握手消息的结构不一样。

1.简单握手

1.1 C0和S0包格式

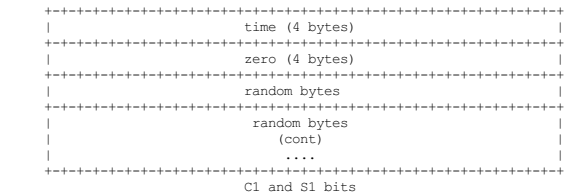
C0和S0 由一个字节组成，表示rtmp的版本号。目前使用的版本号为3。



- version: rtmp版本号。

1.2 C1和S1包格式

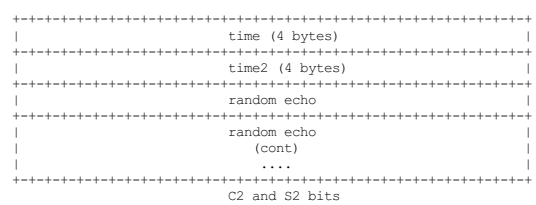
C1 和 S1 包长度为 1536 字节，结构如下：



- time（4 bytes）：时间戳，取值可以为零或其他任意值。
- zero（4 bytes）：本字段必须为零。
- random（1528 bytes）：本字段可以包含任意数据。由于握手的双方需要区分另一端，此字段填充的数据必须足够随机以防止与其他握手端混淆）。

1.3 C2和S2包格式

C2 和 S2 包长度为 1536 字节，分别回应S1和C1。结构如下：

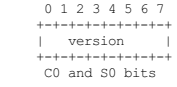


- time（4 bytes）：本字段必须包含发送时刻的时间戳。
- time2（4 bytes）：本字段必须包含时间戳，取值为接收对端发送过来的握手包的时刻。
- random（1528 bytes）：本字段必须包含对端发送过来的随机数据

2.复杂握手

2.1 C0和S0包格式

C0和S0 由一个字节组成，表示rtmp的版本号。目前使用的版本号为3。



- version: rtmp版本号。

2.2 C1和S1包格式

C1 和 S1 包长度为 1536 字节，除了4字节的时间戳和4字节版本号外，还有764字节的key和764字节的digest。有两种结构，如下：

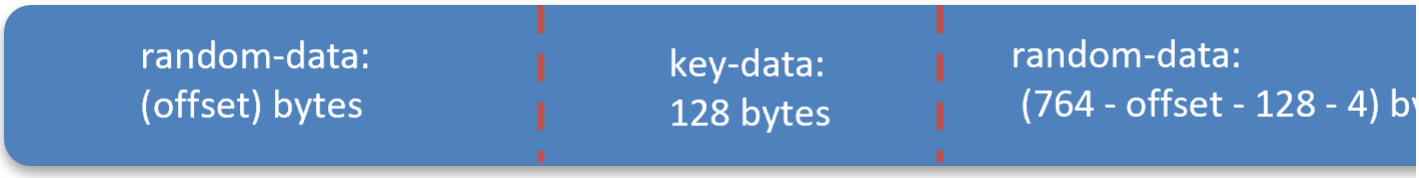
```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        time (4 bytes)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      version (4 bytes)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        key (764 bytes)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      digest (764 bytes)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                        C1 and S1 bits
```

或者

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        time (4 bytes)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      version (4 bytes)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       digest (764 bytes)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      key (764 bytes)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                        C1 and S1 bits
```

- time（4 bytes）：时间戳，取值可以为零或其他任意值。
- version（4 bytes）：客户端为0x0C, 0x00, 0x0D, 0x0E。服务端为0x0D, 0x0E, 0x0A, 0x0D。
- digest（764 bytes）：密文。
- key（764 bytes）：密钥。

764 bytes key 结构：



- random-data: (offset) bytes,随机数据
- key-data: 128 bytes，密钥
- random-data: (764 - offset - 128 - 4) bytes，随机数据
- offset: 4 bytes，密钥的位置

764 bytes digest 结构：

- offset: 4 bytes，digest的位置信息，offset=offset[0]+offset[1]+offset[2]+offset[3]
- random-data: (offset) bytes，随机数据
- digest-data: 32 bytes，计算出来的摘要
- random-data: (764 - 4 - offset - 32) bytes，随机数据
- digest的计算方法：digest-data左边部分+digest-data右边部分的内容，以固定的key,做hmac-sha256计算得到digest。其中，客户端的key:

```
static const uint8_t rtmp_player_key[] = {
    'G', 'e', 'n', 'u', 'i', 'n', 'e', ' ', 'A', 'd', 'o', 'b', 'e', ' ',
    'F', 'l', 'a', 's', 'h', ' ', 'P', 'l', 'a', 'y', 'e', 'r', ' ', '0', '0', '1',

    0xF0, 0xEE, 0xC2, 0x4A, 0x80, 0x68, 0xBE, 0xE8, 0x2E, 0x00, 0xD0, 0xD1, 0x02,
    0x9E, 0x7E, 0x57, 0x6E, 0xEC, 0x5D, 0x2D, 0x29, 0x80, 0x6F, 0xAB, 0x93, 0xB8,
    0xE6, 0x36, 0xCF, 0xEB, 0x31, 0xAE
};
```

服务端的key:

```
static const uint8_t rtmp_server_key[] = {
    'G', 'e', 'n', 'u', 'i', 'n', 'e', ' ', 'A', 'd', 'o', 'b', 'e', ' ',
    'F', 'l', 'a', 's', 'h', ' ', 'M', 'e', 'd', 'i', 'a', ' ',
    'S', 'e', 'r', 'v', 'e', 'r', ' ', '0', '0', '1',

    0xF0, 0xEE, 0xC2, 0x4A, 0x80, 0x68, 0xBE, 0xE8, 0x2E, 0x00, 0xD0, 0xD1, 0x02,
    0x9E, 0x7E, 0x57, 0x6E, 0xEC, 0x5D, 0x2D, 0x29, 0x80, 0x6F, 0xAB, 0x93, 0xB8,
    0xE6, 0x36, 0xCF, 0xEB, 0x31, 0xAE
};
```

2.3 C2和S2包格式

C2 和 S2 包长度为 1536 字节。结构如下：

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    random-data (1504 bytes)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    digest-data (32 bytes)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                        C2 and S2 bits
```

- random-data (1504 bytes)：随机数据
- digest-data (32 bytes)：random-data的摘要。
- digest-data计算方法：
  S2：先通过C1的digest，计算出key，再用这个key计算random-data的digest。
  C2：先通过S1的digest，计算出key，再用这个key计算random-data的digest。

3.实现

- C1和S1的第5到8字节为全0，则使用简单握手模式
- 简单握手不用验证数据，只要按规则交互消息就行
- 复杂握手的S1和C1结构，发送方只需要选择其中一种；而接收方先验证其中一种结构，失败的话再继续验证另一种结构
- digest验证失败，会退到简单握手模式