DH Key Negotiation

recommended p >= 2048,
min >1024

Server

Client

1) Send newSocketConnection request
acceptedKeyGroups: DH key groups
acceptedHashTypes: allowed hash functions
acceptedCrypto: allowed crypto functions

2) Create g_1, p_1, a_1
IF useDynamicSalt: Create g_2, p_2, a_2
Calculate A_1 (and A_2 if applicable)
($A = g^a \% p$)

Create "conInitId" (strLength: 16)

1

/api/v1/client/newSocketConnection
acceptedKeyGroups, acceptedHashTypes, acceptedCrypto
useDynamicSalt

g_1, p_1, A_1, conInitId, g_2, p_2, A_2

3) Create b_1 (and b_2 if useDynamicSalt)
Calculate B_1 (and B_2 if useDynamicSalt)
Calculate S_1 (and S_2 if useDynamicSalt)
($S_n = A_n^{B_n} \bmod p_n$)
Generate shared AES key and IV
(HKDF using static salt or HKDF(S_2))
Create "chkMsg" (random string length 64)
Calculate "chkMsgHash" (sha256(chkMsg))
Encrypt "chkMsg" (send this)

/api/v1/client/newSocketConnection/{conInitId}
B_1, B_2, chkMsg, chkMsgHash

4) Calculate S_1 (and S_2 if useDynamicSalt)
Generate shared AES key and IV
(HKDF using static salt or HKDF(S_2))
Decrypt "chkMsg"
Verify "chkMsgHash" == sha256(chkMsg)
Calculate "confMsg"=sha256(chkMsg + chkMsgHash)
Create unique socketId (length 16)
Create new socket endpoint

socketId, confMsg

5) Verfiy
confMsg==sha256(chkMsg+chkMsgHash)
Connect to server socket
Finalize connection setup

/api/v1/client/socket/{socketId}
clientId, pairId