Gossip-6 LAYER4 Secure Communication Protocol

1. Need secure communication:
   - Establish shared AES256 key: Diffie Hellman key exchange,
   - Need to trust the source (signature),
   - Since there will be no CA, we need Proof of Work for the identity,
   - Proof of work should not be used for more than 1 communication,
   - We are already given an out-of-band public key sharing mechanism.

2. For Alice (client):
   - Let handshake message $\boldsymbol{m} = \boldsymbol{DHE_{pub}^{Alice}} \mid \boldsymbol{RSA_{pub}^{Alice}} \mid \boldsymbol{T} \mid (\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Bob}} \mid \boldsymbol{nonce}$ such that $\boldsymbol{T}$ is the Unix[(1)] timestamp and $\boldsymbol{scrypt_C}(\boldsymbol{m}) < \boldsymbol{k}$ for some pre-determined $\boldsymbol{k} \in \mathbb{Z}^+$ and Scrypt hash function with configuration $\boldsymbol{C}$,
   - Sign the digest $\mathbf{SHA3\_256}(\boldsymbol{m})$ with $\boldsymbol{RSA_{priv}^{Alice}}$ as $\boldsymbol{s} = \boldsymbol{Sign}_{\boldsymbol{RSA_{priv}^{Alice}}}(\mathbf{SHA3\_256}(\boldsymbol{m}))$,
   - Add the signature to the message as $\boldsymbol{m^{Alice}} = \boldsymbol{m} \mid \boldsymbol{s}$ and send $\boldsymbol{m^{Alice}}$ to Bob (server).
3. For Bob (server):
   - Upon receiving $\boldsymbol{m^{Alice}}$, get the fields of the message as $\boldsymbol{m} \mid \boldsymbol{s}$,
   - Check validity first as $\boldsymbol{scrypt_C}(\boldsymbol{m}) < \boldsymbol{k}$, if not valid then discard connection,
   - From $\boldsymbol{m}$, get the fields $\boldsymbol{DHE_{pub}^{Alice}} \mid \boldsymbol{RSA_{pub}^{Alice}} \mid \boldsymbol{T} \mid (\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Bob}} \mid \boldsymbol{nonce}$,
   - Make sure $\boldsymbol{RSA_{pub}^{Alice}}$ is a known and trusted public key*****,
   - Verify signature as $\boldsymbol{Verify}_{\boldsymbol{RSA_{pub}^{Alice}}}(\boldsymbol{m}, \boldsymbol{s})$, if not valid then discard connection,
   - Check if $\boldsymbol{Now} - \boldsymbol{T} \leq \boldsymbol{t_{max}}$ for some pre-determined $\boldsymbol{t_{max}} \in \mathbb{R}^+$ and if not then discard connection,
   - Check if $(\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Bob}}$ is Bob's P2P listening socket and if not then discard connection,
   - Let handshake message $\boldsymbol{m'} = \boldsymbol{DHE_{pub}^{Bob}} \mid \boldsymbol{RSA_{pub}^{Bob}} \mid \boldsymbol{T'} \mid (\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Alice}} \mid \boldsymbol{nonce'}$ such that $\boldsymbol{T'}$ is the Unix[(1)] timestamp and $\boldsymbol{scrypt_C}(\boldsymbol{m'}) < \boldsymbol{k}$ for some pre-determined $\boldsymbol{k} \in \mathbb{Z}^+$ and Scrypt hash function with configuration $\boldsymbol{C}$,
   - Sign the digest $\mathbf{SHA3\_256}(\boldsymbol{m'})$ with $\boldsymbol{RSA_{priv}^{Bob}}$ as $\boldsymbol{s'} = \boldsymbol{Sign}_{\boldsymbol{RSA_{priv}^{Bob}}}(\mathbf{SHA3\_256}(\boldsymbol{m'}))$,
   - Add the signature to the message as $\boldsymbol{m^{Bob}} = \boldsymbol{m'} \mid \boldsymbol{s'}$ and send $\boldsymbol{m^{Bob}}$ to Alice (client).
4. For Alice again (client):
   - Upon receiving $\boldsymbol{m^{Bob}}$, get the fields of the message as $\boldsymbol{m'} \mid \boldsymbol{s'}$,
   - Check validity first as $\boldsymbol{scrypt_C}(\boldsymbol{m'}) < \boldsymbol{k}$, if not valid then discard connection,
   - From $\boldsymbol{m'}$, get the fields $\boldsymbol{DHE_{pub}^{Bob}} \mid \boldsymbol{RSA_{pub}^{Bob}} \mid \boldsymbol{T'} \mid (\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Alice}} \mid \boldsymbol{nonce'}$,
   - Make sure $\boldsymbol{RSA_{pub}^{Bob}}$ is a known and trusted public key*****,
   - Verify signature as $\boldsymbol{Verify}_{\boldsymbol{RSA_{pub}^{Bob}}}(\boldsymbol{m'}, \boldsymbol{s'})$, if not valid then discard connection,
   - Check if $\boldsymbol{Now} - \boldsymbol{T'} \leq \boldsymbol{t_{max}}$ for some pre-determined $\boldsymbol{t_{max}} \in \mathbb{R}^+$ and if not then discard connection,
   - Check if $(\boldsymbol{ip}:\boldsymbol{port})_{\boldsymbol{Alice}}$ is Alice's P2P client socket[(2)] and if not then discard connection.

5. Now that both Alice and Bob have $\boldsymbol{DHE_{pub}^{Alice}}$ and $\boldsymbol{DHE_{pub}^{Bob}}$, they can both calculate the shared secret as $\boldsymbol{AES256_{key}} = \boldsymbol{DHE}(\boldsymbol{DHE_{priv}^{Alice}}, \boldsymbol{DHE_{pub}^{Bob}}) = \boldsymbol{DHE}(\boldsymbol{DHE_{pub}^{Alice}}, \boldsymbol{DHE_{priv}^{Bob}})$. After 1 round trip, the secure communication has been established. Each message following the handshakes will be encrypted with the $\boldsymbol{AES256_{key}}$.

*****: We can check if a public key is known due to the out-of-band hostkey sharing mechanism.
(1): Number of seconds passed since January 1, 1970 UTC time zone.
(2): The (IP, port) that Alice used as a client to connect to the server Bob.