This is a C code project for learning about Ethereum communication.

For example, consider the following documentation from the Ethereum project website.[1]

> Alice wants to send an encrypted message that can be decrypted by Bob's static private key k_B. Alice knows about Bob's static public key K_B.
>
> To encrypt the message m, Alice generates a random number r and corresponding elliptic curve public key $R = r * G$ and computes the shared secret $S = P_x$ where $(P_x, P_y) = r * K_B$. She derives key material for encryption and authentication as $k_E \,\|\, k_M = \texttt{KDF}(S, 32)$ as well as a random initialization vector `iv`. Alice sends the encrypted message $R \,\|\, \texttt{iv} \,\|\, c \,\|\, d$ where $c = \texttt{AES}(k_E, \texttt{iv}, m)$ and $d = \texttt{MAC}\big(\texttt{sha256}(k_M), \texttt{iv} \,\|\, c\big)$ to Bob.

Let

r =	`ephemeral_private_key`	32 bytes
R =	`ephemeral_public_key`	64 bytes
S =	`shared_secret`	32 bytes
K_B =	`peer_public_key`	64 bytes
k_E =	`aes_key`	16 bytes
k_M =	`hmac_key`	32 bytes

Then this is the code for r and $R = r * G$.

```
ec_genkey(ephemeral_private_key, ephemeral_public_key);
```

This is the code for $S = P_x$ where $(P_x, P_y) = r * K_B$.

```
ec_ecdh(shared_secret, ephemeral_private_key, peer_public_key);
```

And this is the code for $k_E \,\|\, k_M = \texttt{KDF}(S, 32)$.

```
kdf(aes_key, hmac_key, shared_secret);
```

[1] `https://github.com/ethereum/devp2p/blob/master/rlpx.md`

Review of shared secrets

Let k_a and k_b be private keys and let K_a and K_b be public keys such that

$$K_a = k_a\, G, \quad K_b = k_b\, G$$

where G is the generator point.

It follows that

$$\frac{K_a}{k_a} = G, \quad \frac{K_b}{k_b} = G$$

Hence

$$\frac{K_a}{k_a} = \frac{K_b}{k_b}$$

and

$$k_b\, K_a = k_a\, K_b$$

After A and B exchange public keys, both can compute shared secret S.

$$k_a\, K_b = S, \quad k_b\, K_a = S$$

Since S is a point, by convention S_x is used for the actual shared secret.