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

$$
\begin{aligned}
r &= \texttt{ephemeral\_private\_key} \\
R &= \texttt{ephemeral\_public\_key} \\
S &= \texttt{shared\_secret} \\
K_B &= \texttt{peer\_public\_key} \\
k_E &= \texttt{aes\_key} \\
k_M &= \texttt{hmac\_key}
\end{aligned}
$$

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

```
// generate ephemeral_private_key and ephemeral_public_key
ec_genkey(ephemeral_private_key, ephemeral_public_key);
```

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

```
// derive shared_secret from ephemeral_private_key and peer_public_key
ec_ecdh(shared_secret, ephemeral_private_key, peer_public_key);
```

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

```
// derive AES and HMAC keys from shared_secret
kdf(aes_key, hmac_key, shared_secret);
```

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