

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 $k_E \parallel k_M = \text{KDF}(S, 32)$ as well as a random initialization vector iv . Alice sends the encrypted message $R \parallel \text{iv} \parallel c \parallel d$ where $c = \text{AES}(k_E, \text{iv}, m)$ and $d = \text{MAC}(\text{sha256}(k_M), \text{iv} \parallel c)$ to Bob.

Let

$r = \text{ephemeral_private_key}$	32 bytes
$R = \text{ephemeral_public_key}$	64 bytes
$S = \text{shared_secret}$	32 bytes
$K_B = \text{peer_public_key}$	64 bytes
$k_E = \text{aes_key}$	16 bytes
$k_M = \text{hmac_key}$	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 $k_E \parallel k_M = \text{KDF}(S, 32)$.

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

¹<https://github.com/ethereum/devp2p/blob/master/rlpx.md>