CypherRock Assignment

To implement the Multiplicative-to-Additive (MtA) protocol using Correlated Oblivious Transfer (COT) for generating additive shares of the multiplication of two random 32-byte numbers, we will use the secp256k1 elliptic curve. The implementation will involve the following steps:

- Elliptic Curve Operations: We will use Trezor's ECDSA library for point multiplication and point addition.
- Correlated Oblivious Transfer (COT): We will implement COT to generate additive shares.
- Finite Field Arithmetic: All operations will be performed under the finite field of the secp256k1 curve.
- Hashing and Encryption: We will use SHA-256 for hashing and XOR for encryption.

Explanation
Elliptic Curve Initialization: The secp256k1 curve is initialized using OpenSSL's EC_GROUP functions.

Random Number Generation: Random values a and b are generated within the finite field of the curve.

Additive Shares Generation: The generate_additive_shares function computes the additive shares c and d using COT.

Verification: The program verifies that a * b = c + d under the finite field of the curve.

Output: The values of a, b, c, and d are printed in hexadecimal format.

Download the folder and open it in the VSCode.

Open **Git Bash** or VS Code's terminal and run:
git clone https://github.com/trezor/trezor-firmware.git
cd trezor-firmware/crypto
cp -r C:\Users\YourUsername\Documents\trezor-firmware\crypto .\trezor-crypto

Make sure MinGW or MSYS2. Installed and in VScode install c/c++ extension
Also python compiler

Open the Terminal
Write these commands

gcc main.c -L./trezor-crypto -lcrypto -o MtA_COT.exe

./MtA_COT

Outputs of a,b,c,d will be different every time

```
PS C:\Users\PRATHAM\Desktop\MtA_COT> gcc main.c  -L./trezor-crypto -lcrypto -o MtA_COT.exe
main.c: In function 'sha256':
main.c:12:5: warning: 'SHA256_Init' is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   12 |     SHA256_Init(&sha256);
      |     ^~~~~~~~~~~
In file included from main.c:4:
C:/msys64/ucrt64/include/openssl/sha.h:73:27: note: declared here
   73 | OSSL_DEPRECATEDIN_3_0 int SHA256_Init(SHA256_CTX *c);
      |                           ^~~~~~~~~~~
main.c:13:5: warning: 'SHA256_Update' is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   13 |     SHA256_Update(&sha256, data, len);
      |     ^~~~~~~~~~~~~
C:/msys64/ucrt64/include/openssl/sha.h:74:27: note: declared here
   74 | OSSL_DEPRECATEDIN_3_0 int SHA256_Update(SHA256_CTX *c,
      |                           ^~~~~~~~~~~~~
main.c:14:5: warning: 'SHA256_Final' is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   14 |     SHA256_Final(hash, &sha256);
      |     ^~~~~~~~~~~~
C:/msys64/ucrt64/include/openssl/sha.h:76:27: note: declared here
   76 | OSSL_DEPRECATEDIN_3_0 int SHA256_Final(unsigned char *md, SHA256_CTX *c);
      |                           ^~~~~~~~~~~~
PS C:\Users\PRATHAM\Desktop\MtA_COT> ./MtA_COT
Verification successful: a * b = c + d mod n
a: 9072ABC1DA24A69575C4F0D934CBB9FCA0562BF991E96D2FE68E2F4AB8F76559
b: C8BA6E803B1DB5D1918232DC141708BD706C8E576A831711F916EDDB457ED593
c: C366EA534AFCF67E141E4E20A4836D3C56A1D69FA8B7B327EC57E7758B150377
d: 88DE432008FC14F7BE4A066E4ED98719BC48866E20FB42796968206D4081F306
```

To check the given output, copy the outputs of **a,b,c, and d** into Python file **verify_mta.py Respectively.**

Run the command
python verify_mta.py

```
PS C:\Users\PRATHAM\Desktop\MtA_COT> python verify_mta.py
Verification successful: a * b = c + d mod n
a * b mod n: 0x4c452d7353f90b75d268548ef35cf457583b80271a6a556595eda955fb60b53c
c + d mod n: 0x4c452d7353f90b75d268548ef35cf457583b80271a6a556595eda955fb60b53c
PS C:\Users\PRATHAM\Desktop\MtA_COT>
```

When you run the C code you will get different values od a,b,c,d so paste them in python code for verification

```
PS C:\Users\PRATHAM\Desktop\MtA_COT> ./MtA_COT
Verification successful: a * b = c + d mod n
a: 8187E101E8413652EF8965BE3A60C82924C98E77D9453572BBA3564B8BF7E54D
b: 245338869658D0CC100D35AFAC6E5FF49891BB5086F93876ADC2619ABC828CF2
c: 9BB2CA836A5451CDC8BE22EE4023041836179FD3AADB1368AF2D3799BA5B5464
d: C23E61818458773EBB725AFD58F311E7717E0B122CCC58C66AAFF938507A68CC
PS C:\Users\PRATHAM\Desktop\MtA_COT> python verify_mta.py
Verification successful: a * b = c + d mod n
a * b mod n: 0x5df12c04eeacc90c84307deb99161600ece6cdff285ecbf35a0ad2453a9f7bef
c + d mod n: 0x5df12c04eeacc90c84307deb99161600ece6cdff285ecbf35a0ad2453a9f7bef
```