hack-sdk-python

Python version SDK for hackathon.

Requirements

Python 3 or higher.

Installation

Directly import the *.py files into your workspace folder
You may find IntelliJ PyCharm extremely helpful.

To install the dependencies, use pip:

```
pip install coincurve pycryptodome kafka-python requests
```

Usage

Initialization

All APIs are provided in og_solver.py and can be called once the OGSolver is inited.

```
og = OGSolver(API_URL, TOKEN)
my_account = Account(private_key)
```

Here you need to put your token assigned by committee in order to successfully call the APIs.

Basic Queries

do_og_solver.py includes all usages of APIs provided. For example, you may call query_all_txs_in_pool() to get all transactions that is currently in the pool and to be picked up as parents.

```
og = OGSolver(API_URL, TOKEN)
print(og.query_all_txs_in_pool())
```

The results will be in dict format

```
{
    "data": {
        "sequencer": {
            "type": 1,
            "hash":
"0x8de248720f97d949e7de227505384ca9fce4c59c785a2dab13e316157b5ebd15",
            "parents": [

"0x63bb2fb37fb6aeecb23ea55c1ccd76a61a1588c409b1b62d4d3d51049a3519db"
            ],
            "from": "0x7349f7a6f622378d5fb0e2c16b9d4a3e5237c187",
            "nonce": 2276,
            "treasure": "1000",
            "height": 2276,
            "weight": 0
        },
        "transactions": [
            {
                "type": 0,
                "hash":
"0x86f4ef1c9b01e727767fb78ac13b29217eb99165b14d4adb8b1292d8573fa225",
                "parents": [

"0x8de248720f97d949e7de227505384ca9fce4c59c785a2dab13e316157b5ebd15"
                ],
                "from": "0xc4321fee1e29b13b042feab06dea55e7caf85948",
                "to": "0x0000000000000000000000000000000000000000",
                "nonce": 597,
                "guarantee": "1",
                "value": "0",
                "weight": 0
            }
        ]
    },
    "err": ""
}
```

To send a transaction, call send_tx():

```
og.send_tx(self, account, parents, nonce, sender, guarantee, pubkey, to=None,
value=0)
```

Message Queue Events

To get the latest transaction (including those txs that have not been confirmed) and sequencers, we provide a message queue to annoucne the new tx events. Once a new Tx/Seq is broadcasted, you will be notified if you subscribe the Kafka topic:

```
# this is the callback for receiving a new tx:
consumer = KafkaConsumer(KAFKA_TOPIC, bootstrap_servers=KAFKA_SERVER)
for message in consumer:
    tx = json.loads(message.value.decode('utf-8'))
    on_new_tx(tx)
```

A further processing function should be there to handle this tx:

```
def on_new_tx(tx):
    if tx['type'] == 0:
        print('Received Tx: ', tx)
        # Maybe we need to follow it?
    elif tx['type'] == 1:
        print('Received Seq: ', tx)
        # Maybe we need to follow it?
```

You may find the code in strategy.py