НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп’ютерного практикума

Розгортання систем Etherium та криптовалют

Виконали студенти
групи ФЕ-31мп
Мятка І.І.
Кирилюк Д.В.
Столярчук Т.В.

Перевірила:
Байденко П. В.

Київ — 2024

Мета роботи: «Отримання навичок налаштування платформ виконання смарт-контрактів та криптовалют».

Для другого типу лабораторних робіт:

Провести налаштування обраної системи та виконати тестові операції в системі Ethtrium.

Теоретичні відомості:

Ethereum - це мережа з безлічі вузлів / нод, або EVM (Ethereum Virtual Machine). EVM - це програмне забезпечення, яке може розуміти написаний за певними правилами набір інструкцій (для запуску транзакції, смарт-контрактів, застосунків) і виконувати їх у мережі Ethereum у деякому логічному порядку, як звичайний комп'ютер.

Щоб запустити приватний блокчейн, комп'ютер має стати вузлом у мережі Ethereum, тобто потрібно завантажити весь блокчейн мережі та синхронізувати його з основним ланцюжком Ethereum. Для цього можна використовувати різні інструменти (клієнт), але в контексті лабораторної роботи потрібні ті, які дадуть змогу взаємодіяти з мережею блокчейну:

- ***Geth (Go Ethereum)***. Реалізація мови програмування Go (Golang) для Ethereum називається Geth. Це одна з трьох початкових реалізацій протоколу Ethereum (поряд з Python і C++). Використовується для експериментування або професійного створення інтерфейсу для dapps.
- ***Клієнт на основі Python***, що реалізує "криптоекономічний кінцевий автомат Ethereum". Потрібно обирати, для розробки dapps або академічних дослідженнях мережі Ethereum.
- ***C++ (eth)***. Імплементація Ethereum на C++. Використовується, якщо серйозно ставитись до майнінгу на GPU або потрібно підвищити безпеку, запустивши дві реалізації програми.

Geth - це інструмент інтерфейсу командного рядка (CLI), який пов'язує вашу систему та обладнання з мережею Ethereum. З Geth можливо:

- здійснювати фінансові транзакції;
- майнити ефір (утилітарна валюта мережі Ефіріума);
- створювати смарт-контракти, децентралізовані додатки;
- створювати приватні блокчейни;
- досліджувати історію блоків;
- багато іншого.

Розумні контракти - це основні цеглини для створення додатків Ethereum. Це комп'ютерні програми, що зберігаються на блокчейні, які слідують за

логікою "якщо це, тоді то" і гарантовано виконуються відповідно до правил, визначених його кодом, який не може бути змінений після створення.

Головна перевага розумного контракту в тому, що він однозначно виконує недвозначний код за дотримання певних умов. Немає потреби чекати людину для виконання будь-яких необхідних операцій. Це усуває необхідність у довірених посередниках.

Об'єкти розумного контракту:

- ***Підписанти*** - сторони розумного контракту, які приймають або відмовляються від умов з використанням електронних підписів. Прямим аналогом є підпис відправника коштів у мережі Bitcoin, який підтверджує внесення транзакції в ланцюжок блоків.
- ***Предмет договору***. Предметом договору може бути тільки об'єкт, що знаходиться всередині середовища існування самого розумного контракту, або ж повинен забезпечуватися безперешкодний, прямий доступ розумного контракту до предмета договору без участі людини.
- ***Умови***. Умови розумного контракту повинні мати повний математичний опис, який можливо запрограмувати в середовищі існування розумного контракту. Саме в умовах описується логіка виконання пунктів предмета договору.
- ***Децентралізована платформа***. Для розподіленого зберігання смарт-контракту необхідний його запис у блокчейні цієї платформи

Транзакція зазвичай складається з наступних параметрів: [nonce, gasPrice, gasLimit, to, value, data, v, r, s]. Проте, Ethereum розвивався, щоб дозволити інші стандарти транзакцій, такі як EIP-1559. Стандарт транзакцій EIP-1559 з'явився в результаті пропозиції щодо вдосконалення, щоб забезпечити більш передбачувані збори за газ і більш ефективний ринок транзакцій.

Існує два типи облікових записів: *облікові записи смарт-контрактів* і *зовнішні облікові записи (EOA)*:

- ***Зовнішні облікові записи (EOA)*** - це облікові записи, якими керує людина, наприклад, особистий гаманець Metamask або Coinbase. Цей обліковий запис ідентифікується за допомогою відкритого ключа (також відомого як адреса облікового запису) і контролюється за допомогою закритого ключа. Публічний ключ отримується з приватного за допомогою криптографічного алгоритму. Важливо зазначити, що ці акаунти не можуть зберігати іншу інформацію, окрім балансу та nonce [це номер транзакції адреси відправника].
- ***Акаунти смарт-контрактів*** (також відомі як контрактні акаунти) також містять адресу для відображення балансу, але відрізняються тим,

що можуть містити EVM-код і сховище. Контрактні акаунти контролюють себе за допомогою логіки в EVM-коді, що зберігається в акаунті.

Ethereum використовує алгоритм цифрового підпису еліптичної кривої (ECDSA) для підтвердження автентичності (тобто, підтвердження того, що у нас є приватний ключ для нашої публічної адреси) і перевірки того, що наша транзакція походить від акаунта, який підписав транзакцію, і не є шахрайською.

Типи транзакцій

- ***Транзакція виклику повідомлень***: Обмін повідомленнями відбувається від зовнішнього облікового запису, який хоче взаємодіяти з іншим обліковим записом EOA або контрактним обліковим записом. Прикладом виклику повідомлення може бути відправка Ефіру з одного облікового запису на інший або взаємодія зі смарт-контрактом (наприклад, обмін токенів на Uniswap).
- ***Транзакція створення контракту***: Транзакція створення контракту походить від EOA для створення облікового запису смарт-контракту (як правило, для зберігання коду і сховища). Прикладом такого типу транзакції може бути розгортання смарт-контракту для зберігання даних.

Хід виконання:

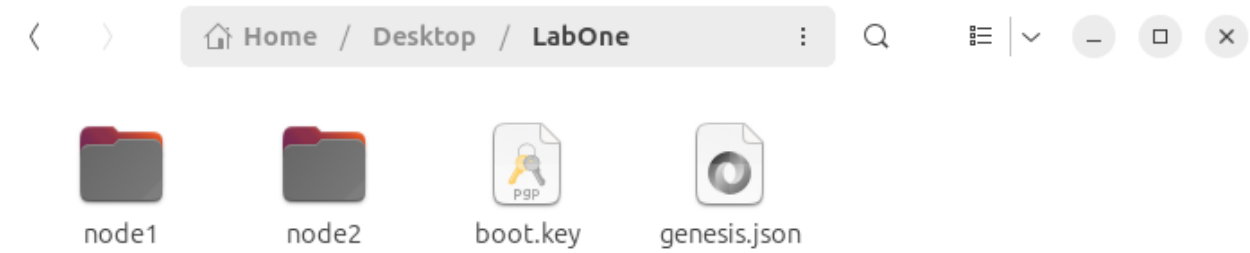
1. **Встановлення geth**

На OS Ubuntu, виконуючи вказівки з офіційної документації (https://geth.ethereum.org/docs/getting-started/installing-geth) за допомогою PPAs було встановлено geth та встановлено останню стабільну версію.

```
ubuntu@ubuntu:~/Desktop$ geth version
Geth
Version: 1.13.15-stable
Git Commit: c5ba367eb6232e3eddd7d6226bfd374449c63164
Architecture: amd64
Go Version: go1.21.6
Operating System: linux
GOPATH=
GOROOT=
```

2. **Створення нових акаунтів**

Виконуючи інструкції (https://geth.ethereum.org/docs/fundamentals/private-network) була створена проста приватна мережа на 2 ноди. Відповідно було створено 2 папки «`node1`» та «`node2`» в папці «`LabOne`»:



Після цього, за допомогою наведених внизу команд в CLI було створено 2 нових акаунти.

```
geth --datadir node1 account new
```

```
geth --datadir node2 account new
```

Відповідно до одержаного вікна повідомлення було зазначено паролі до цих акаунтів «123456».

```
ubuntu@ubuntu:~/Desktop/LabOne$ geth --datadir node1 account new
INFO [04-23|13:06:59.270] Maximum peer count                       ETH=50 total=50
INFO [04-23|13:06:59.272] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or dir
ectory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0xD475d859660C75107a001D6BB69adC996C27Ba98
Path of the secret key file: node1/keystore/UTC--2024-04-23T13-07-03.749747315Z--d475d859660c75107a001d6bb69adc996c27ba
98

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

```
ubuntu@ubuntu:~/Desktop/LabOne$ geth --datadir node2 account new
INFO [04-23|13:07:09.386] Maximum peer count                       ETH=50 total=50
INFO [04-23|13:07:09.390] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or dir
ectory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A
Path of the secret key file: node2/keystore/UTC--2024-04-23T13-07-13.508494239Z--6eb25208b5dfb95654c1a86c70d80c840c8dab
0a

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

3. Створення `genesis` файлу

Було створено файл `genesis.json` з використання команди `touch genesis.json` в директорії папки `LabOne`. Вміст файлу показаний на рисунку нижче.

Тут:

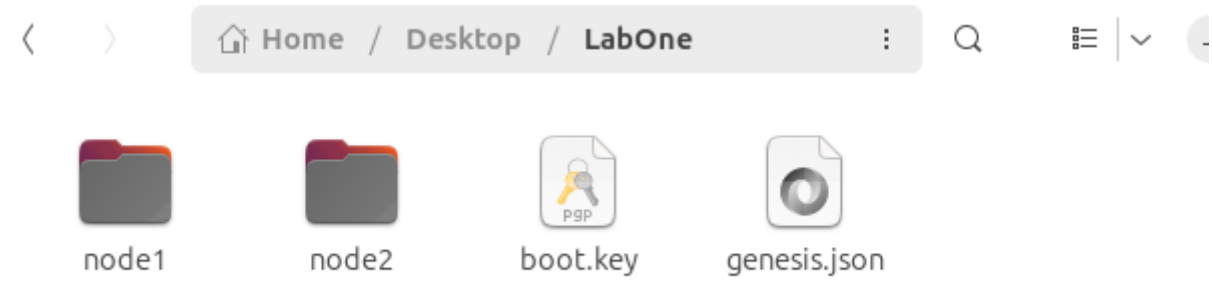
1) Встановлено `chainId` приватної мережі, яка є «вільною», тобто не занятою.
2) `homesteadBlock`: Homestead - це перший виробничий реліз Ethereum, і оскільки розробники вже використовують цю версію, значення цього параметра можна залишити рівним '0'.
3) `eip150Block/eip155Block/eip158Block`: EIP розшифровується як "Пропозиції щодо покращення Ethereum", вони були впроваджені для випуску Homestead. При розробці приватного блокчейну хардфорки не потрібні, тому значення параметра слід залишити рівним "0".
4) `difficulty`: Контролює складність головоломки майнінгу, і чим нижче значення, тим швидше відбувається майнінг.
5) `gasLimit`: встановлює верхню межу для виконання смарт-контрактів.
6) `alloc`: Дозволяє розподілити Ефір за певною адресою.

Для економії часу та наглядності початкове значення балансу для 2х адрес було встановлено 10 ETH.

4. Запуск бут-ноди

Наступним кроком буде налаштування завантажувального вузла. Це може бути будь-яка нода, але відповідно до прикладу з документації буде використано інструмент розробника bootnode для швидкого і простого налаштування виділеної ноди. Спочатку завантажувальному вузлу потрібен ключ, який можна створити за допомогою наступної команди, яка збереже ключ у файлі boot.key:

```
bootnode -genkey boot.key
```



Запускаємо ноду за допомогою команди:

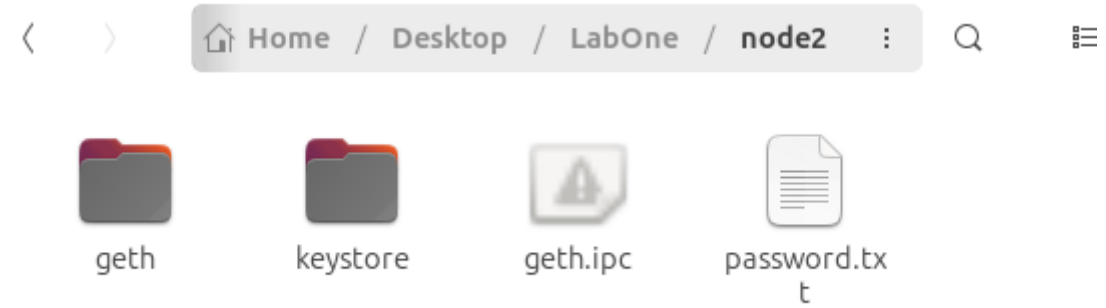
```
bootnode -nodekey boot.key -addr :30305
```

Вибір порту, що передається в `-addr`, є довільним, але публічні мережі Ethereum використовують `30303`, тому цього краще уникати. Команда `bootnode` повертає терміналу наступні логи, підтверджуючи, що він запущений:

```
ubuntu@ubuntu:~/Desktop/LabOne$ bootnode -genkey boot.key
ubuntu@ubuntu:~/Desktop/LabOne$ bootnode -nodekey boot.key -addr :30305
enode://c88b3cfe978fa70338e89005da898097c4dddeec6778b1b9827e6e28636c4aa78429f1f7f2d56fc52c24bbd0283ae3d00e67a988f797ffb
71d0c0c91fa8c503f@127.0.0.1:0?discport=30305
Note: you're using cmd/bootnode, a developer tool.
We recommend using a regular node as bootstrap node for production deployments.
INFO [04-23|13:17:52.429] New local node record                    seq=1,713,878,272,426 id=24cbd7b579a3cba8 ip=<nil> u
dp=0 tcp=0
```

5. Запуск node1 та node2

Попередньо створимо файл `password.txt` в кожній ноді. Файл зберігає той самий пароль «123456».



Запустимо node1 командою:

```
ubuntu@ubuntu:~/Desktop/LabOne$ geth --datadir node1 --port 30306 --bootnodes enode://c88b3cfe978fa70338e89005da898097c
4dddeec6778b1b9827e6e28636c4aa78429f1f7f2d56fc52c24bbd0283ae3d00e67a988f797ffb71d0c0c91fa8c503f@127.0.0.1:0?discport=30
305 --networkid 7795 --unlock 0xD475d859660C75107a001D6BB69adC996C27Ba98 --password node1/password.txt --authrpc.port 8
551 --mine --miner.etherbase 0xD475d859660C75107a001D6BB69adC996C27Ba98
```

Зауважимо, що ми розблоковуємо першу ноду з використанням вказаного в `password.txt` паролю. Розблокування дозволяє ноді підписати транзакцію. Окрім цього на ноді розпочато процес майнингу (`--mine`) та відповідно встановлена адреса, на яку буде приходити нагорода (`--miner.etherbase`). Після цього починається сам майнинг.

```
INFO [04-23|15:32:20.164] Unlocked account                         address=0xD475d859660C75107a001D6BB69adC996C27Ba98
INFO [04-23|15:32:20.165] Legacy pool tip threshold updated        tip=0
INFO [04-23|15:32:20.165] Legacy pool tip threshold updated        tip=1,000,000,000
INFO [04-23|15:32:20.166] Commit new sealing work                  number=362 sealhash=4db41b..f16a6f txs=1 gas=21000 f
ees=2.1e-05 elapsed="857.37µs"
INFO [04-23|15:32:20.866] Successfully sealed new block            number=362 sealhash=4db41b..f16a6f hash=bad84f..89bb
00 elapsed=699.896ms
INFO [04-23|15:32:20.866] Commit new sealing work                  number=363 sealhash=c9f5c2..039e62 txs=0 gas=0     f
ees=0       elapsed="195.728µs"
INFO [04-23|15:32:25.005] Successfully sealed new block            number=363 sealhash=c9f5c2..039e62 hash=493722..1aac
c0 elapsed=4.139s
INFO [04-23|15:32:25.008] Commit new sealing work                  number=364 sealhash=022d7a..94591a txs=0 gas=0     f
ees=0       elapsed=1.558ms
INFO [04-23|15:32:28.884] Looking for peers                        peercount=0 tried=1 static=0
INFO [04-23|15:32:30.002] Successfully sealed new block            number=364 sealhash=022d7a..94591a hash=956e89..9fa3
93 elapsed=4.994s
INFO [04-23|15:32:30.003] Commit new sealing work                  number=365 sealhash=181c65..c60161 txs=0 gas=0     f
ees=0       elapsed="228.146µs"
INFO [04-23|15:32:35.002] Successfully sealed new block            number=365 sealhash=181c65..c60161 hash=fa93ba..5f5e
fe elapsed=4.998s
```

Для другої ноди просто розблокуємо акаунт за допомогою команди:

```
ubuntu@ubuntu:~/Desktop/LabOne$ geth --datadir node2 --port 30307 --bootnodes enode://c88b3cfe978fa70338e89005da898097c
4dddeec6778b1b9827e6e28636c4aa78429f1f7f2d56fc52c24bbd0283ae3d00e67a988f797ffb71d0c0c91fa8c503f@127.0.0.1:0?discport=30
305 --networkid 7795 --unlock 0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A --password node2/password.txt --authrpc.port 8
552
```

Отримаємо відповідне інформаційне повідомлення:

```
INFO [04-23|14:34:02.774] Unlocked account                         address=0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A
INFO [04-23|14:34:12.129] Looking for peers                        peercount=0 tried=1 static=0
INFO [04-23|14:34:22.385] Looking for peers                        peercount=0 tried=0 static=0
INFO [04-23|14:34:32.587] Looking for peers                        peercount=0 tried=0 static=0
WARN [04-23|14:34:36.842] Post-merge network, but no beacon client seen. Please launch one to follow the chain!
INFO [04-23|14:34:42.754] Looking for peers                        peercount=0 tried=1 static=0
INFO [04-23|14:34:52.948] Looking for peers                        peercount=0 tried=0 static=0
INFO [04-23|14:35:02.999] Looking for peers                        peercount=0 tried=0 static=0
INFO [04-23|14:35:13.037] Looking for peers                        peercount=0 tried=0 static=0
INFO [04-23|14:35:23.089] Looking for peers                        peercount=0 tried=1 static=0
```

6. Використання консолі JS

Аби увійти в консоль першої ноди використаємо команду:

`geth attach node1/geth.ipc`

```
ubuntu@ubuntu:~/Desktop/LabOne$ geth attach node1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.15-stable-c5ba367e/linux-amd64/go1.21.6
coinbase: 0xd475d859660c75107a001d6bb69adc996c27ba98
at block: 147 (Tue Apr 23 2024 14:38:09 GMT+0000 (UTC))
 datadir: /home/ubuntu/Desktop/LabOne/node1
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

В якості перевірки виведемо баланс кожної ноди у `wei`:

```
> eth.getBalance('0xD475d859660C75107a001D6BB69adC996C27Ba98')
1000000000000000000
> eth.getBalance('0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A')
1000000000000000000
```

7. Виконання транзакції

Для виконання транзакції виконаємо наступні команди:

```
eth.sendTransaction({
```

```
  to: '<adress>',
  from: '<adress>',
  value: 25000
});
```

В якості відправника виступає 1-ша нода, в якості отримувача 2-га.

```
> eth.sendTransaction({  to: '0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A',  from: eth.accounts[0],  value: 50000000});
"0x4d338a5982a4d63146a0ed81bbb9c7b5d3938e25d57ae8ab2f963a95d04084f7"
```

В даному випадку в якості відправника був вказаний нульовий індекс акаунтів, тобто перша нода. К-сть `wei 50000000`. Нові баланси мають вигляд:

```
> eth.getBalance('0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A')

10000000000050000000
> eth.getBalance('0xD475d859660C75107a001D6BB69adC996C27Ba98')

9999999999949681000
```

Кількість на отримувачі змінилась вірно. Через помилки у ф-ції транзакції при першій спробі деяка кількість `wei` була втрачена, натомість в результаті зупинки процесу майнингу деяка кількість була повернута. В загальному вдалось «намайнити» таку кількість блоків:

```
> eth.blockNumber
1813
```

8. Інформація про транзакцію

Виконаємо наступну транзакцію:

```
> eth.sendTransaction({from:eth.accounts[0], to:"0x6eb25208B5Dfb95654C1a86C70d80c840C8dAb0A", value: web3.toWei(1, "mwe
i")})
"0x7bfcc853d8d0f4de22a5a1f616cf6af0ce370722d7502dc5c848609ea1504fba"
```

В цьому випадку `1 mwei = 1000000 wei`, тоді як `1 ETH` = 10^{18} `wei`.

Отримати інформацію про транзакцію можна скориставшись командою:

Формат транзакцій для Ethereum спочатку був єдиним стандартом, але з часом він еволюціонував і дозволив використовувати інші формати транзакцій. Набір інструкцій в об'єкті транзакції виглядає наступним чином:

- `from` - адреса відправника.
- `to` - адреса отримання (якщо це EOA, то транзакція передаватиме вартість. Якщо рахунок смарт-контракту, транзакція буде використовувати код контракту).
- `value` - сума ETH, яка буде відправлена з адреси відправника (деномінована в Wei)
- `gasLimit` - максимальна кількість одиниць газу, яку можна використати.
- `nonce` - номер, який використовується для відстеження черговості транзакцій та запобігання повторним атакам
- `maxPriorityFeePerGas` - максимальна кількість газу, яка буде включена в якості чайових для майнера.
- `maxFeePerGas` - максимальна кількість газу, яку майнер готовий заплатити за транзакцію (включаючи `baseFeePerGas` і `maxPriorityFeePerGas`).
- `signature` - походить від приватного ключа акаунта-відправника і створюється, коли відправник підписує транзакцію.

Висновки

В процесі виконання роботи була розгорнута приватна мережа `ethereum` за допомогою `Geth`, який є клієнтом `Ethereum` написаному на мові `Go`. Створено `genesis` файл мережі та згенеровано дві ноди та бут-ноду. На одній

з нод запущено процес майнингу. Обидві ноди були розблоковані за допомогою вказаного пароля та здійснена перевірка балансу і транзакція між двома вузлами мережі. Створення приватної мережі може бути використано розробниками для тестувань або розробки власних рішень.