The Economics of FinTech, Lecture 3

Steven Kou[1]

[1]Department of Finance
Questrom School of Business
Boston University

Outline

Outline

Overview

- We will add MetaMask to the web browser Chrome
- We will use MetaMask to do some simple things, such as depositing and transferring some testing ETH's
- We will do a simple programming, which contains "Hello, world" and some simple arithmetic.
  - We will test it first on the JavaScript VM using Remix
  - We will deploy the code on the test network, using both Remix and MetaMask.
- We will then write codes for a larger programming, by issuing your own crypto token.
  - We will test it first on the JavaScript VM using Remix
  - We will deploy the code on the test network, using both Remix and MetaMask.
- We will verify the results on the Etherscan network.

Outline

Introduction to Metamask

- MetaMask is an add-on to the web browser Chrome
- MetaMask can be used to interact with blockchains, such as sending and receiving money, deploying programs related to smart contracts, etc.
- Alternatively, one can use Mist Browser.
- Install MetaMask from Google Web Store.
- Choose a password, which will automatically generate a new account for us.
- The new account will have an account name and a private key.
- Then after clicking "accept" to some standard items, you have an account and vault (12 words) created.

Introduction to Metamask

- These 12 words are the only way to restore a MetaMask account.
- These words are called mnemonic.
- These 12 word mnemonic is linked to all accounts, each with an address, public key and private key.
- See a demonstration in https://iancoleman.io/bip39/
- MetaMask also allows you to link to a hardware wallet, such as Trezor or Ledger.

Introduction to Metamask

- There are different Ethereum networks, Main Network (meaning real money), Ropsten, Kovan, Rinkeby test networks (meaning no real money), Localhost 8545, and Custom RPC.
- In the course, we will only use the test networks.
- Note that one account is used across all Ethereum networks.
- Request free testing ether from https://faucet.metamask.io/
- Request free testing ether from https://faucet.ropsten.be/
- Request free testing ether from http://rinkeby-faucet.com
- Note that all the test networks are decoupled.

Simple Operations in Metamask

- Create multiple accounts
- Do a transfer within two accounts
- A transaction object consists of the transaction number (called nonce), to (address), value (amount), gasPrice (the amount of ether the sender is willing to pay per unit gas), startGas/gasLimit (the maximum units of gas the sender is willing to pay), plus some cryptographic signature numbers generated using the sender's private key.

Outline

Workflow of the transaction

- An example of transaction
- Request free testing ether from http://rinkeby-faucet.com
- Address sent to Server
- Server used web3 library (just like using JavaScript to interact with web pages) to create a "transaction" object
- Server sent "transaction" object to the Rinkeby test network
- Waited for the transaction to be confirmed
- Server sent success message back to the browser

Solidity Programming Language

- Solidity is similar to JavaScript, with an extension ".sol"
- Solidity can be written by any text editor
- Then the Solidity code is complied to generate some Bytecode for the Ethereum networks to process
- A simple text editor is Remix, https://remix.ethereum.org/

“Hello World”

- Line 1 is the version of the Solidity.
- Line 3 gives the contract name.
- Line 4 is a remark line.
- Line 7 defines a variable message, which is public. All public variables will be recorded in blockchain permanently.
- Anything write to the blockchain will cause (gas or money).
- Lines 9 to 11 defines a function with input.
- A function has 3 parts, name with input variables, function type, and return types.The last part is optional.
- For the input variable note the word “string memory”.
- Function types: public (any one can call this function), private (only this contract can call), view (can read but cannot modify the blockchain data, i.e. no gas involved), pure (not even read the data), payable (involve ether).

“Hello World”

- Lines 12 to 14 return a message. Note that difference between “returns” and “return” in the code.
- In all input variable, one has to specify whether the input variable is just in the memory (which will disappear after finishing the session) or in the storage.
- In the doMath part, “uint” means unassigned integers from 0 to 2^{256}.
- Note Solidity can only handle integers (not fractions or decimals).
- For example, in Solidity $5/6 = 0$, $-10 = 2^{256} - 10$, and $2^{256} + 10 = 9$.
- Thus, one has to be very careful about operations.
- Also a pure function cannot read blockchain data.

Test "Hello World"

- If there is something wrong with the code, there will be error messages on Remix
- Remix is not just a code editor. It also has a fake Ethereum network.
- After verifying there is no error message, one can compile the code.
- Then one can click the run icon.
- Note that the gas limit is the maximum amount of gas one wants to spend.
- Choose JavaScript VM for our test purpose.
- Later when we deploy the code using the Ethereum test network, which needs fake ether, we can use Injected Web3.
- Value is an icon that we will use when the contract involves fake or real ether.

Test “Hello World”

- At the bottom half of the page are the items related to the contract output.
- Note that the contact name is already there after the compile.
- There are two ways to deploy a contract, either use the icon deploy or use the blockchain address of the icon (if the contract is already recorded on the blockchain).
- In the JavaScript VM, the address is not a real address on any (test or real) blockchain.
- Later after we use the test network, we can deploy the contract from the contract address.
- For now, we can simply click the deploy icon.

Test “Hello World”

- Now you can see various outputs.
- Note that the output changes, once the input changes.
- Also the input for string should contain “ and “.
- If we change the function to be pure, then the code cannot go through.
- Note since our functions are public, any one can use the function.
- Note that the above function outputMessage is redundant.
- Delete a few lines of code will not change the existing output, unless one do the compile and deploy the contract again.

Some Remarks.

- Every time we write the data to the blockchain we have to pay gas.
- View a data set does not cost gas.
- In the JavaScript VM, there is no mining; thus the transaction is immediate.
- In the test network, the transaction is not immediate due to simulated mining.
- To get a nicer web output, one can use Web3.js to interact with the Solidity, just like one can use JavaScript to design web pages.

Gas and Wei

- 1 ether $= 10^{18}$ wei.
- See wei converter
- Basic gas costs: add 3 wei, substract 3 wei, multiply 5 wei ....
- See https://github.com/ethereum/EIPs/blob/master/EIPS/eip-150.md
- One can speed up the mining process by increase the gas unit.
- In MetMask, there are 3 simple recommended gas level, slow, fast, fastest.

Outline

Set up the right account

- First, make sure that you have enough test ether.
- Second, make sure that you sign in to Metmask and click the right account to run the code.
- Third, make sure that you use the same Chrome browser to run Remix.
- Fouth, click Injected Web3 in Remix.
- If you have done all these steps correctly, you should see the Remix recognize your account and show the correct account value.

Deploying the “Hello” contract to a test network

- Follow the same step as before, i.e. running compile, run and deploy.
- Click Deploy.
- You should see MetMask asking you how many ethers you are willing to pay.
- After you choose the ether amount, there will be a mining time.
- Then there will be confirmation.
- You can check the transaction on the blockchain, via Etherscan.
- You can also get the contract number to run the contract publicly.
- Congratulations! Your first contract is officially on the blockchain!

Outline

The Code

- function transfer
- event
- function require
- emit
- return true

Issuing Your Own Contract: Test on Remix

- Copy and paste account numbers
- Be careful with the selected account

Outline

Issuing Your Own Contract on the Blockchain

- Run things on the blockchain with MetaMask
- Copy the contract ID
- Using the contract ID to load the token to MetaMask

HWK Problem

- Issue 300 shares of your own token with
- name: Your Name (FirstnameLastname), symbol: Your School ID
- Send me 50 shares of the token to the following address
- 0xb105C18B4535DAD952F7D0Ab2553fC0223759A42
- Please email me your contract ID, so that I can add the token and verify the contract.