CS5363 Blockchain Technologies and Applications: Hw8 – Report
Decentralized App (DApp)

Name: 王念祖

Student Id: 112164507

Task: A Social Recovery Wallet DApp on Web3

Ⅰ、Background Introduction

在過往的紙幣時代，人們往往隨身攜帶一個小錢包，裡頭會存放自身的一小部份財產，用於出外交易使用，另外一大部份財產通常不會隨身帶著，而是存放於銀行之中，亦是將錢財託管於一個信任的第三方機構，不過，在 Web3 數位世界中，沒有所謂的可信任第三方，每個人皆生成一個 EOA 帳戶，用戶必須自行保管該帳戶所對應的私密金鑰，擁有這把金鑰是可以花費其 EOA 帳戶中的所有金錢，因此私鑰的管理變得極度重要，其中一種安全的保管方式，常用帶有安全加密芯片的設備來儲存此私密金鑰，不過若設備損壞，隨之保存於其中的私鑰也將遺失，導致用戶失去所有資產。

為了解決以上問題，有一派的研究是討論如何安全備份與還原私鑰，而另一個派研究，則是此次選擇實作的方法，使用情境可以想像成上述的錢包與銀行保險庫的例子，每個人都有一個錢包(EOA 帳戶)以及銀行保險庫(Smart Contract Wallet)，人們大多數的資產都存放在自己鏈上的保險庫之中，等自己手上的錢包金錢不足時，才去保險庫提領部份金錢來做使用，另外當我們覺得錢包錢太多時，也可以對保險庫匯入金額進行儲蓄，這個保險庫的錢只能有事先設定好的 EOA 錢包才可以提領，可以想像成 EOA 錢包即時這個保險庫的擁有者，不過，如前所述，當我們的 EOA 錢包中的私鑰遺失時，就如同現實世界我們遺失錢包，其中的金錢會一併遺失，不過我們儲存在銀行中的錢還是我們的，在 Web3 的世界中，我們會事先設定好此保險庫的監護人們，這群監護人們會為我們在私鑰遺失時，進行社交驗證，確認是我本人後，並且藉由我提供新錢包的地址，請監護人設定我的保險庫，改成新帳戶持有人是我提供的新錢包地址來做接管做提領，此流程需達到預期數量門檻的監護人覆議，就可以重新設定我們保險庫能有哪一個 EOA 錢包做新的提款帳戶。

Social recovery wallet 簡單來說也就是部署一個智能合約，裡頭實作錢包功能，並作為可靠且可驗證的保險金庫管理，最大特色是配合 social

authentication 中的監護人(guardian)角色來為金鑰遺失場景，做一個帳戶復原的機制。

II、Features of Smart Contracts

2-1 重要屬性 Important attributes

1. **address public owner;** //此保險庫的帳戶擁有者(EOA address)
2. **uint public totalDepositAmount;** //此保險庫的總金額（wei 單位）
3. **uint public threshold;** //閾值，要多少 guardian 數量同意
4. **bool public isRecovering;** // 現在是否是在“帳戶恢復”階段
5. **uint public currRecoveryRound;** // 現在已經是第幾次的恢復
6. **address[] public guardians;** //儲存現在的所有監護人

2-2 重要方法 Important functions

1. **constructor(uint _threshold, address[] memory _initialGuardians)**
   部署階段，需設定閾值，以及所有監護人的地址，這裡就會固定好 n of m ，譬如 n=5 個監護人與 m=2 個同意會成功，n of m 之後就不能在更改，帳戶擁有者之後只能更換監護人地址，並不能更改監護人數量與閾值的比例
2. **function depositMoney() external payable onlyOwner**
   存款功能（不能在恢復期間呼叫）
3. **function withdrawMoney(uint withdrawValue) external payable onlyOwner**
   提款功能（不能在恢復期間呼叫）
4. **function removeGuardian(address removingGuardian) external onlyOwner**
   帳戶擁有人可以移除（替換）監護人，不過因安全性需要等待一段時間（譬如 3 天），原因是避免惡意人士拿到帳戶擁有人的私鑰後，透過快速替換成對自己有利的監護人以轉移保險箱的帳戶擁有人
5. **function executeGuardianRemoval(address removingGuardian, address newGuardian) external onlyOwner**

等待移除監護人的時間到了，就可以立即刪除他，並需要在輸入一個新的監護人地址以替換移除者的位置，保持監護人數量的一致

6. function cancelGuardianRemoval(address removingGuardian) onlyOwner external

帳戶擁有者可以取消對監護人的移除操作

7. function transferGuardian(address newGuardian) external onlyGuardian

監護人可以將自己的監護人權利轉交給別人，但若該監護人目前正在移除期間，將不能轉移監護人權限給予他人

8. function initiateRecovery(address _newOwnerAddr) onlyGuardian external

任何一個監護人皆可以啟動此保險庫的恢復流程，發起此恢復流程的監護人，需指定希望將此保險庫權限交給哪一個的新帳戶擁有人地址

9. function supportRecovery(address _newOwnerAddr) onlyGuardian onlyInRecovery external

任何一個監護人發現保險庫的恢復流程開始後，就可以覆議恢復流程，一樣需指定希望將此保險庫權限交給哪一個的新帳戶擁有人地址

10. function cancelRecovery() onlyOwner onlyInRecovery external

帳戶擁有人可以取消恢復流程，避免監護人共謀串通轉移帳戶到非帳戶擁有者想指定的新帳戶地址。

11. function executeRecovery(address newOwner, address[] calldata guardianList) onlyGuardian onlyInRecovery external

任何一個監護人可以提交所有覆議的監護人地址，並讓合約檢查是否符合閾值數量的監護人同意，之後就將保險庫權限改為指定的新帳戶地址。

2-2 Flow chart

III、How to use our Dapp?

3-1 Test and deploy our smart contract

啟動 Ganache，選 QUICKSTART

測試合約的正確性

```
daniel_wang@sd233097 social-recovery-dapp % cd social-recovery-dapp
daniel_wang@sd233097 social-recovery-dapp % ls
README.md          node_modules       package.json       src
contract           package-lock.json  public
daniel_wang@sd233097 social-recovery-dapp % cd contract
daniel_wang@sd233097 contract % truffle test
```

測試結果如下，確認沒有問題

```
Deployed the contract to 0xEF85AE8649f78E484555B38e01Cbb4BDf61Ec262
Set the threshold of the guardian recovery to 2


  Contract: SocialRecoveryWallet
    ✔ should initiate recovery process (135ms)
    ✔ should support recovery process (224ms)
    ✔ should execute recovery process (242ms)
    ✔ should not allow recovery initiation if already in recovery process (139ms)
    ✔ should prevent non-guardians from initiating recovery
    ✔ should prevent non-guardians from supporting recovery (100ms)
    ✔ should prevent non-guardians from executing recovery (193ms)
    ✔ should prevent recovery if threshold not met (133ms)
    ✔ should allow only owner to remove guardian
    ✔ should allow guardian removal if removal period was passed (450ms)
    ✔ should prevent guardian removal if removal period not passed (64ms)
    ✔ should allow only guardians to transfer guardian status
    ✔ should prevent transferring guardian during recovery process (72ms)
    ✔ should successfully transfer guardian status (429ms)
    ✔ should not allow recovery initiation if already in recovery process (78ms)
    ✔ should not allow recovery execution without sufficient support (123ms)
    ✔ should not allow recovery execution with wrong guardian address amount (132ms)
    ✔ should not allow recovery execution if not in recovery process
    ✔ should allow owner to deposit money (71ms)
    ✔ should allow owner to withdraw money (109ms)
    ✔ should revert when non-owner tries to deposit money
    ✔ should revert when non-owner tries to withdraw money (53ms)
    ✔ should revert when owner tries to withdraw more than the total deposit amount (65ms)


  23 passing (3s)
```

部署合約

```
daniel_wang@sd233097 contract % truffle migrate --network development

Compiling your contracts...
===========================
> Compiling ./contracts/SocialRecoveryWallet.sol
> Artifacts written to /Users/daniel_wang/blockchain/social-recovery-dapp/social-recovery-dapp/contra
ct/build/contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_deploy_contracts.js
=====================

   Replacing 'SocialRecoveryWallet'
   --------------------------------
   > transaction hash:    0x60534d6df0a31de06d9d78657f5e1b80d8d6bc6048414fe7905c6f8dbd41341d
   > Blocks: 0            Seconds: 0
   > contract address:    0xE098FC019f346A703CE5B911A1f0442824cbFE84
   > block number:        3
   > block timestamp:     1714408928
   > account:             0x430D8B99b68496670d0eb204D0cDfe0FEf4E52Cf
   > balance:             99.971461011298988392
   > gas used:            2832994 (0x2b3a62)
   > gas price:           3.340972956 gwei
   > value sent:          0 ETH
   > total cost:          0.009464956338510264 ETH

Deployed the contract to 0xE098FC019f346A703CE5B911A1f0442824cbFE84
Set the threshold of the guardian recovery to 2
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.009464956338510264 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.009464956338510264 ETH


daniel_wang@sd233097 contract %
```

3-2 configure and activate Dapp

找到 social-recovery-dapp/social-recovery-dapp /contract/build/contracts/SocialRecoveryWallet.json, 並獲取兩個值 “address” 跟 “abi”, 並且只需要複製其值

```
social-recovery-dapp > contract > build > contracts > {} SocialRecoveryWallet.json > {} networks > {} 5777 > address
22705    "compiler": {
22708    },
22709    "networks": {
22710      "5777": {
22711        "events": {},
22712        "links": {},
22713        "address": "0xE098FC019f346A703CE5B911A1f0442824cbFE84",
22714        "transactionHash": "0x60534d6df0a31de06d9d78657f5e1b80d8d6bc6048414fe79
22715      }
22716    },
```

```
social-recovery-dapp > contract > build > contracts > {} SocialRecoveryWallet.json > [ ] abi
> "abi":                                                  Aa ab .* 1 of 1
1  {
2    "contractName": "SocialRecoveryWallet",
3    "abi": [
4      {
5        "inputs": [
6          {
7            "internalType": "uint256",
8            "name": "_threshold",
9            "type": "uint256"
10         },
11         {
12           "internalType": "address[]",
13           "name": "_initialGuardians",
14           "type": "address[]"
15         }
16       ],
17       "stateMutability": "nonpayable",
18       "type": "constructor"
```

把以上這兩個值，貼到 social-recovery-dapp/social-recovery-dapp/src/pages/abi/SocialRecoveryConfig.js

```
social-recovery-dapp > src > pages > abi > JS SocialRecoveryConfig.js > ...
1  // use ctrl+ F to find "address": in Social_Recovery/build/contracts/SocialRecc
2  export const CONTACT_ADDRESS = "0xE098FC019f346A703CE5B911A1f0442824cbFE84";
3
4  // use  ctrl+ F to find "abi": in Social_Recovery/build/contracts/SocialRecover
5  export const CONTACT_ABI =  [
6    {
7      "inputs": [
8        {
9          "internalType": "uint256",
10         "name": "_threshold",
11         "type": "uint256"
12       },
13       {
14         "internalType": "address[]",
15         "name": "_initialGuardians",
16         "type": "address[]"
17       }
18     ],
19     "stateMutability": "nonpayable",
```

在此 social-recovery-dapp/social-recovery-dapp/ 路徑下，啟動 Dapp ($ npm start)

```
Compiled successfully!

You can now view social-recovery-dapp in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

連線到 http://localhost:3000/，應該類似長這樣，但右上的帳戶文字可能顯示不同文字或是沒有任何文字，其文字表示目前與 Dapp 連線的帳戶地址

先敘述一下，現在的情況，因爲預設在 Smart Contract 在部署階段時，預設 threshold = 2, initialGuardians = Ganache 的前三個合約地址

接下來，先用 MetaMask 創建以上的一些帳戶，作為等一下使用場景的測試，譬如 account[0] (帳戶擁有人＋監護人), account[1] (監護人), account[3] (欲轉移的新帳戶擁有人)

MetaMask 匯入方式大同小異，這邊示範 account[0] 的方法

1

2

3

4

5

6

7

8

9

10

設定好後，將帳戶連線 Dapp ，可以重新整理 Dapp 網頁頁面

可以發現右上角的帳戶地址已經改為設定好的 ganache 的 account[0] 地址了，而其他的 account[1], account[3]也是同樣步驟新增至 MetaMask，只是之前設置的 localhost 網路可以沿用不需重新設定。

第一次連入 Dapp 都需要確認連線，記得即使連線，Dapp 還是要重新整理讓右上角帳戶文字確保為對應要連線帳戶

接下來我們用 Account2 (帳戶擁有人登入)

回到 Dapp 頁面，首先介紹左上角的「logo」跟「首頁」點下去後，都是導向此 Welcome to Social Recovery Wallet 字樣的主頁面

3-3 Deposit and Withdraw

接下來點擊右邊數過來第三個「帳戶金額」，若使用帳戶擁有人登入 Dapp，則會顯示“存錢”和“領錢”按鈕可以操作

但如果是用非帳戶擁有人登入 Dapp，則無此兩個按鈕，但一樣看得到帳戶擁有人地址以及保險庫總金額，如下圖

點擊「存錢」按鈕，輸入存款金額，按下確認

此時 MetaMask 會跳出交易確認

確定 8000 wei 已經存入保險庫

點擊「領錢」按鈕，輸入提款金額

確定提出 2000 wei ， 保險庫已經扣除剩 6000 wei

如果要領出超過金庫總金額是無法的，會被擋下來。

3-4 Guardian management

接著來介紹右上 Navbar 的倒數第二個「監護人管理」功能點下去

同樣若用帳戶擁有者＋監護人權限登入，以下畫面介紹權限顯示：

假設我們試著刪除表格中第三個監護人，按下對應其位置的刪除鍵

Metamask 一樣需要您按下交易確認

可以看到現在第三個監護人變成等待刪除中，因為我們為了測試，在部署合約中，我們只需等到區塊鏈時間 10 秒即可變為刪除，但因為是測試鏈只有我們會挖新區塊更新時間，因此我們需做一個其他交易動作已更新最新區塊時間。我們可以在刪除第二個監護人，等一下可以測試我們「取消」功能

可以看到第三個監護人已經可以立即刪除了，我們就執行第三個監護人的「立即刪除」，不過刪除前，他需要我們新增新的監護人地址，我們就從 ganache 中挑選 account[4]

Metamask 一樣需要您按下交易確認

可以看到第三個監護人已經換成新的帳戶地址擔任，我們剛剛加入刪除排程中的第二個監護人地址，也因等待時間到了，因此變成可以立即刪除，但這次我們要測試「取消」功能

Metamask 一樣需要您按下交易確認

可以看到第二個監護人又恢復到任期狀態

3.5 Account Recovery

接著來介紹右上 Navbar 的最後一個「帳戶復原」功能點下去

同樣若用帳戶擁有者＋監護人權限登入，以下畫面介紹權限顯示

先使用 account[1] (僅監護人身份) 與 Dapp 連線

接下來發起交易，並且我們先隨意指定一個 account[7] 來表示自己想將金庫權限隨意轉移給他人。

Metamask 一樣需要您按下交易確認

此時我們可以發現帳戶已經進入第一次的恢復流程，並且帳戶狀態是恢復中，上方的按鈕也變成可以覆議跟總結，並且下方的表格也紀錄著 account[1]是希望將新 owner 跟 account[7]，接著我們先回到 account[0](就是擁有帳戶擁有者＋監護人權限)的帳戶，將其重新連線 Dapp。

此時可以發現帳戶在恢復狀態下，存錢與領錢的按鈕都不再顯示，此時 account[0] 的帳戶復原頁面比一般監護人多一個按鈕，也就是取消恢復流程，因為 account[0] 發現自己沒有委託恢復，但 account[1]卻偷啟動恢復流程想轉移，幸好 account[0] 即使發現並按下「取消恢復流程」

Metamask 一樣需要您按下交易確認

帳戶狀態又復原回正常狀態

接下來我們來走一次復原成功流程，假設今天 account[0]遺失私鑰，無法在發起交易，他快速聯絡 account[1]監護人，希望他發起恢復流程，並交給他希望轉移的新帳戶(account[3])，因此我們用 account[1] 登入連線 Dapp，並發起第二次的恢復流程，並輸入 account[0]交代的 account[3]作為新帳戶擁有人

Metamask 一樣需要您按下交易確認

可以看到第二次的恢復流程啟動，並且 account[1]此次選 account[3]作為 newOwner，不過要真正成功復原當初設定需要兩個數量監護人贊同，因此 account[0]需聯絡第三個監護人 account[4]，account[4]對 account[0]的本人進行社交認證成功後，同意為其執行帳戶復原操作，因此我們登入 account[4] 連線 Dapp，複製 account[1]提出的地址，點選“覆議恢復請求”，貼上後送出。

Metamask 一樣需要您按下交易確認

並且看到已有兩個監護人支持同一個新帳戶擁有人地址後，就可以按下“總結提交“按鈕，並再次輸入新帳戶擁有人地址以及覆議的兩位監護人地址

Metamask 一樣需要您按下交易確認

到帳戶管理頁面，可以發現帳戶擁有人已經被成功轉移給新帳戶 account[3]！

IV、Other Discussion

<社交恢復錢包的延伸安全性問題討論>：

社交恢復錢包需要關切的安全議題，主要就是「共謀攻擊 collusion attack」，因為此方法是基於一群監護人彼此提出要讓哪一個新帳戶來接管此合約錢包，並且當有一個新帳戶地址達到設定閾值數量的監護人附議時，此新帳戶能夠接管原帳戶擁有者成為新帳戶擁有者，在這個機制下，原帳戶擁有者實際很可能要求監護人能提出他要求的新帳戶地址，但一種可能是並未被監護人們所採納，反而讓監護人們串通一起將新帳戶地址的提議改為監護人們自行受益的帳戶，進而架空原帳戶持有者，目前對於該情形的應對方案就是，帳戶

擁有者可以透過將不聽話的監護人移除監護人權限（不過基於安全問題需要一段時間），以及帳戶持有人有權可取消發起帳戶恢復的操作，透過這兩點，帳戶持有者可以取消監護人提出的錯誤新帳戶地址提案，並且隨著持續追蹤取消提案，並等待一個固定時間後，就可將不聽話的監護人移除監護人列表。

但此設計會引發另一個問題，因爲發起社交恢復操作是由監護人權限去觸發的，因此監護人若無預警發起未經帳戶持有人要求發起的帳戶恢復交易，並暗中串通其他監護人覆議，達到未經授權的轉移帳戶的情形發生，為避免上述情形，帳戶持有者須時時監控鏈上資訊，避免有惡意監護人發起未經帳戶持有者允許的帳戶恢復，這部份也算是非常麻煩。