信任機制(區塊鏈)之趨勢與運用

Dr. Liao (廖世偉), National Taiwan University

Outline

1. Introduction
2. Blockchain: Trust Machine
3. Blockchain 趨勢
4. Blockchain 運用

Introduction

中本聰是個穿越時代的人

Trust → FinTech → Digital Currency

or

Digital Currency → FinTech → Trust

?

An important question:

- Because for blockchain 工作者: Look at bitcoin’ethereum’ or blockchain such as Gcoin ?

區塊鏈，有事嗎？

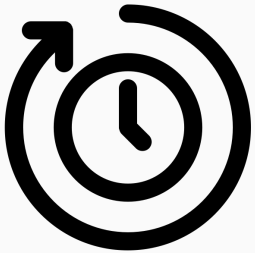
1. https://www.youtube.com/watch?v=LJUig2-tyPU
2. Caveat: 影片是我的學生辛苦為各位準備的，並非我做的。
3. Caveat: 影片內容可能太年輕人了，請包含。

Outline

1. Introduction
2. Blockchain: Trust Machine
3. Blockchain 趨勢
4. Blockchain 運用

Blockchain: Trust Machine

Blockchain: Trust Machine



增加交割效率

Efficiency

加強安全性

Finality

可信任性

Inspectability

區塊鏈: 增加數位經濟效率, Trust 將促進資產流動性

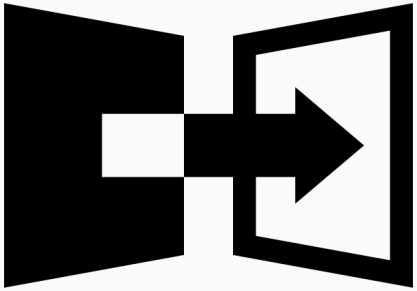
Trust Machine key feature

Efficiency
- No more 45-day latency on mileage transfer!
- Move mileage asset to Digi-ledger: From T+45 to T0
- No more $50,000 cost!
- Move cap-table to Digi-ledger: From $50,000 to 0

Finality
- Reduce counterparty risk

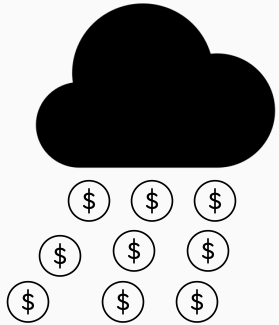
Inspectability (or Gongping, Gongzheng, Gongkai)
- API becomes possible
- Silicon Valley is about APIs.

Technology-wise 表述 Digital Finance問題

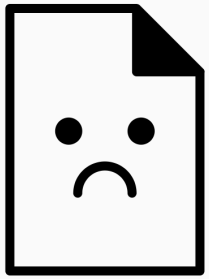


Duplication

Double Spending



Dilution



Data Lost

Digital Finance之前發展不起來的原因:

4D → Blockchain 如何一次解決:

Blockchain 如何解決 4D 問題

Duplication:

- Protected by the Proof-of-work mechanism
- Every client has a copy of the list of transactions

Double Spending:

- The transaction needs to be confirmed by the nodes in the Blockchain network.
- For example, Bitcoin protects against double spending by verifying each transaction added to the Blockchain to ensure that the inputs for the transaction had not previously been spent

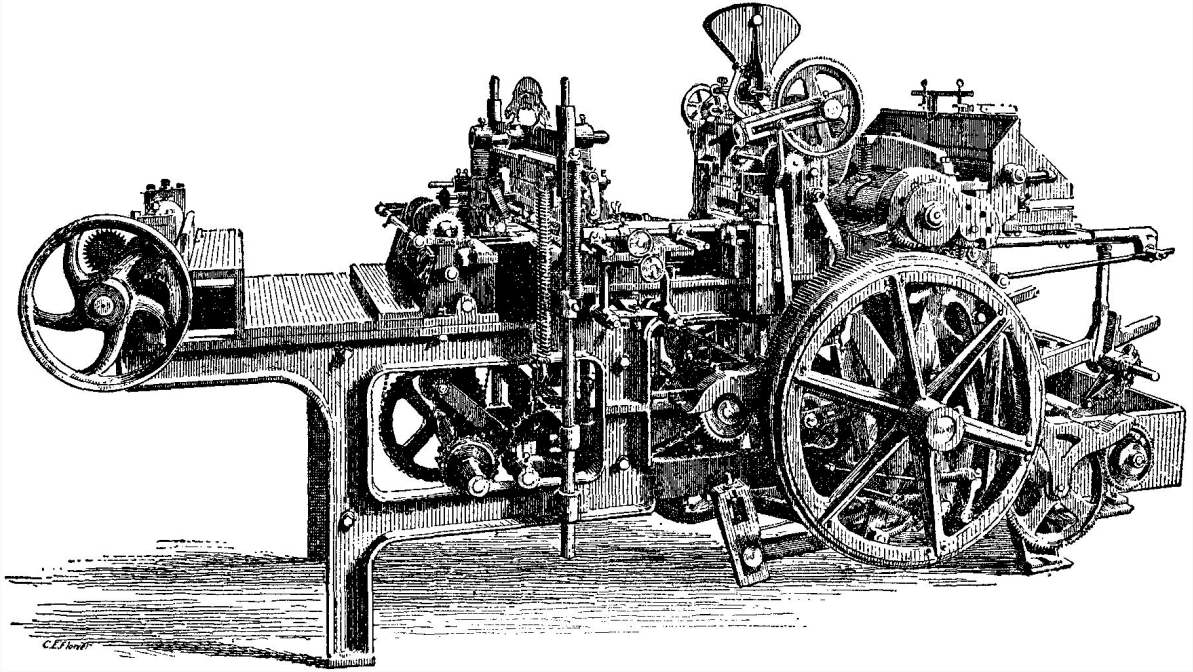
Dilution:

- The amount of coin can be limited by proper design via protocol and algorithm.
- Decentralized system: no central authority controls the right to issuing currency

Data lost:

- All transactions are trackable and are recorded in the Blockchain

區塊鏈的本質:
State-Machine based Trust Machine



Remember: blockchain machine is Mathematics-based:

- Math just works;

  You don’t need to trust it:
  Trust Machine vs. Trust Company

What kind of Trust Machine is it?

To database researchers:

Blockchain is NOT just a brand new data store.

To computer network researchers:

Blockchain is NOT just a P2P network.

To virtual machine and compiler professors:

Blockchain is NOT just a smart contract.

To programming language professors:

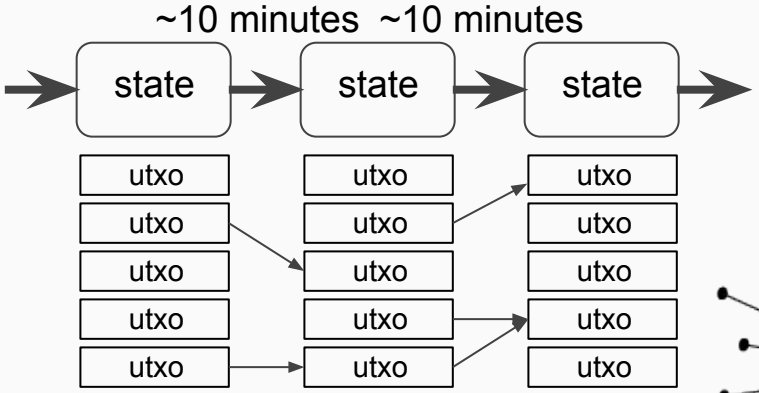
Blockchain is NOT just a programming platform.

Blockchain's key technology: State-Machine based immutable Trust Machine

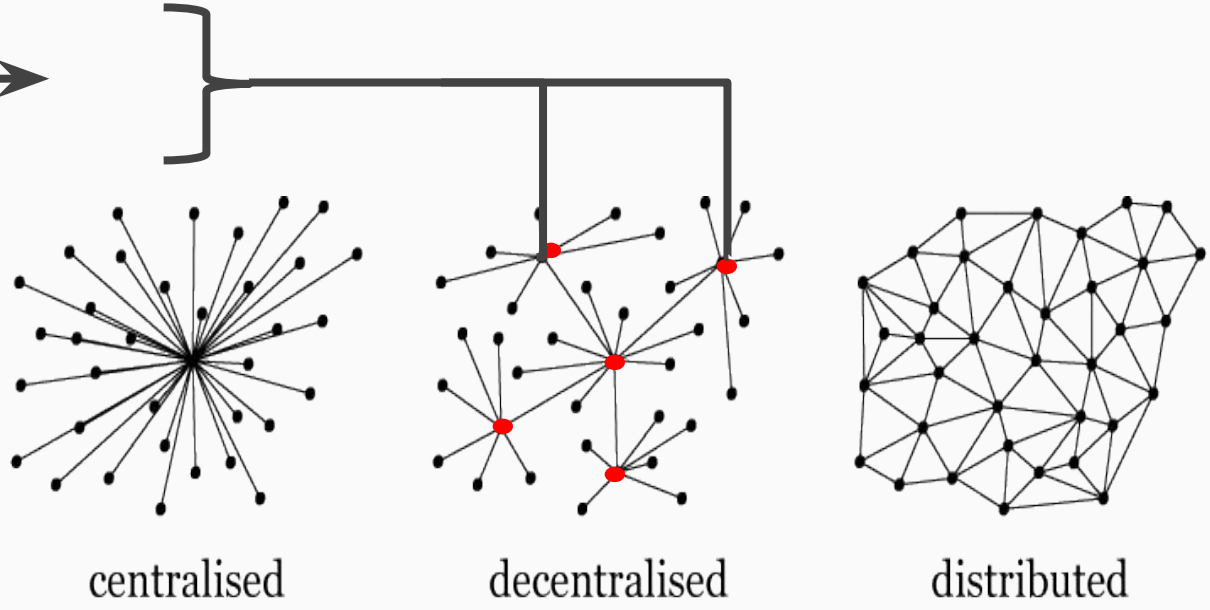
Don't jump to database, network, … applications. 蹲馬步.

核心技術 first (See next 2 slides)

State machine-based immutable 多中心 Trust Machine



utxo: unspent transaction output



How the Internet and blockchain both work. Ledger is synchronized by “full” (red) nodes.

What kind of Trust Machine is it?

去中心化	去信任化	安全可靠	不可逆性
共同參與	公開透明	匿名性	無法篡改

信任機制

融合核心技術支持

1. 隱私安全: CoinJoin, CoinShuffle, Zero-knowledge proof, Coin Mix, Mix server, 51% attack
2. 數學: One way function
3. 密碼學: 單向散列演算法/公鑰加密/ECC加密算法/SHA-256演算法
4. 經濟模型: 貨幣的發行設計機制/賽局理論
5. 算法算力: Byzantine Generals problem/解難題/Hashcash/Markov Chain
6. Scalability (Global)
7. Flexibility (Governance structure, smart contract)

Outline

1. Introduction
2. Blockchain: Trust Machine
3. Blockchain 趨勢
4. Blockchain 運用

Blockchain 趨勢

「區塊鏈技術將改變金融業」

李顯龍, United Overseas Bank 80th anniversary (Nov 2015)

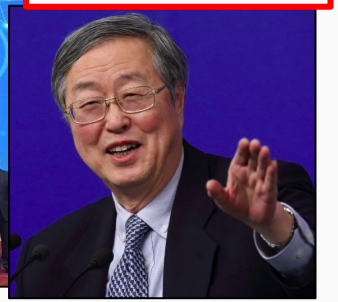
“Blockchains, which is used for bitcoin, but can also be used for many other applications like real-time gross settlement, or trade finance verification. So our banks and our regulators must keep up to date and up to scratch with these developments“

2016國際情勢

全球超過45家銀行組成R3聯盟
共同研發區塊鏈技術標準

中國政府牽頭成立：
中國區塊鏈研究聯盟
中關村區塊鏈產業聯盟

多國央行已展開
區塊鏈技術研究



Nasdaq

London Stock Exchange

TOKYO STOCK EXCHANGE

多國證交所已展
開區塊鏈交易之
測試與研究


E BOOSTER
BLOCKCHAIN MAKES POSSIBLE: INVESTMENT, DEVELOPMENT & OPERATIONS.
BUILDING A DIGITAL SILK ROAD FOR WIN-WIN COOPERATION
Information Infrastructure Partnership
数字丝路·合作共赢论坛
信息基础设施共建
WORLD INTERNET CONFERENCE
5 INFRASTRUCTURE BOOST

Outline

1. Introduction: 區塊鏈, 有事嗎?
2. Blockchain: Trust Machine
3. Blockchain 趨勢
4. Blockchain 運用

Blockchain 運用

當今局勢

1. 一銀案件的錢(等值 US$2m+):嫌犯預備用 Bitcoin 洗出去。
2. Ethereum forked 成 ETC (Ethereum Classic) and ETH, due to DAO案件:
   - 金額是一銀的30倍
   - Smart Contract 的風險
3. 香港交易所(for 美金/ETH/BTC) BFX 被hack $66m → 債卷發行 BFX 解決
4. 馬雲用區塊鏈做眾籌(聯合報:https://video.udn.com/news/534829)
5. 兆豐因協助洗錢嫌疑被罰款:$180m

Lessons for possible 監理沙盒:

如果要做監理沙盒, 基本criteria:

1. 需要性:
   a. 新加坡政府指出, 沒有必要在沙盒裡做的, 就不必在沙盒裡。
2. 進步性:真有 create value 嗎？Value 要有嚴謹性, 非吹泡泡。
   a. 自動化 and 智慧化的好處 應與 區塊鏈的好處分開分析
3. 創新性:Responsible Innovation
   a. 並且不是放出一個 experimental platform 就好 --- DAO will come.
4. 安全性:
   a. 消費者保護:風控精算能力 and 賠償準備機制
5. 實用性:
   a. 能落地在當地者。

區塊鏈應用

g-coin.org

支付結算系統

顧客忠誠計劃

crowdsourcing 平台

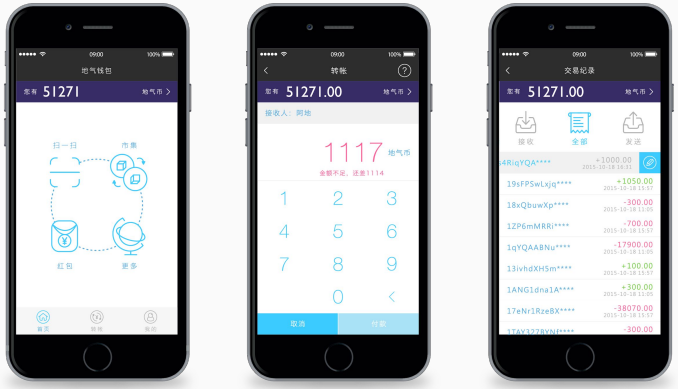
私募平台

票據平台

去中心化交易所

區塊鏈行動支付系統

- **問題**:線上支付等需要藉由區塊鏈達到數字化，使權益擁有人直接存取，進行交易、獲取交易資訊等，而不需透過中介商。
- **方案**:藉由區塊鏈技術提供一套 ID 管理系統，僅有私鑰的擁有者可以存取資產，擴充原有行動錢包的支付場景。



價值:區塊鏈免除交易中介者 (intermediaries) 的金流成本。**預估使用區塊鏈的支付結算成本可降至 0.00025 % (0.025 bp)** [1]

區塊鏈對支付系統的效益

區塊鏈強調交易即清算

✓不會有帳不平的問題

- ✓ GCOIN 區塊鏈平均數秒即完成結算

✓不需要額外的對帳成本 (人力+系統)

區塊鏈的不可竄改特性

✓讓系統安全性有多一層的保護；由區塊鏈技術降低的信任成本為全球金融機構每年省下200億美元。See [2]

區塊鏈對支付系統的效益

區塊鏈為一套 P2P 分散式架構系統

✓透過密碼學的保護，交易可直接在手持裝置端做完並且發送

- ✓ 降低server負擔
  - 經測試，交易的產生過程佔整體時間超過 70%
- ✓ 可支援小額線下付款，增加支付市場份額
  - 預估使用區塊鏈技術可增加10%支付市場份額[3]

支付區塊鏈的案例

- BIS(國際清算銀行)指出區塊鏈技術可能應用到中央銀行的運作
- 顧問公司艾特集團(Aite Group)預估銀行一年投資 7500 萬美元在區塊鏈技術改善支付系統
- 高盛集團(Goldman Sachs)替旗下的區塊鏈技術結算系統「SETLCoin」申請專利
- Circle 提供基於區塊鏈技術的美元轉帳，於2015年獲得 5000 萬美元融資
- Coinbase 提供數字貨幣的行動錢包服務，於2015年獲得7500萬美元融資

全球知名金融機構已積極加速佈局投資[4]

區塊鏈交易平台

- **問題:** 交易平台需要可追蹤性，讓金流可被追蹤並視覺化以避免被濫用。
- **方案:** 區塊鏈技術提供一份可共同驗證的分散式帳本系統，可應用於個種交易平台。

區塊鏈技術降低 99% 交易結算風險 [5]

區塊鏈對交易系統的效益

加速交割效率 (→15秒)

✓數字化股權藉縮短交割時間可降低 99% [6] 結算風險

可追蹤性、流動性

✓區塊鏈原生支援次級市場，交易門檻大幅降低

- ✓ 區塊鏈使 Pre-IPO trading 更具吸引力
- ✓ 2015年股權眾籌平均投資人數僅6人，平均融資成功率約為20%，金額更是只有總目標金額的7%。

區塊鏈的交易平台案例 (證券)

- **NASDAQ 使用區塊鏈發行 pre-IPO 股票**
  - 該技術可以降低系統性成本的80%至90%，因為交易的主要成本在於清算過程
- **Overstock 創建了tØ區塊鏈股權交易平台**
  - 2015 年 12 月，美國證券交易委員會（SEC）批准在 Overstock 通過區塊鏈來發行公司股票
  - 至少 5 家企業已通過該區塊鏈平台發行股票
- **UBS 於 2015 年 9 月完成測試並推出了 Smart Bonds 系統，用於各類債券的發行和交易**
  - UBS 認為，區塊鏈系統未來 2 年內一定會商業化

區塊鏈股權交易平台 2 年內會商業化

DAO

超過10億台幣被DAO (盜), 並2016/7/20暴力fork了blockchain

- Important event: Time will tell eventually
  - What about immutability and other questions that come from this forking?
  - The original chain (Ethereum Classic) vs. The mutated chain?
- Ethereum has created a lot of extra work for exchanges in order to ‘accommodate’ their DAO losses...

DAO

DAO (Decentralized Autonomous Organization) 是一個群眾募資的智能合約

他運行於以太坊 (Ethereum) 的區塊鍊上

ref: https://daohub.org

https://en.wikipedia.org/wiki/The_DAO_(organization)

DAO

在以太坊上的使用者如果持有 DAO 幣，即可參與是否支持 DAO 上的募資計劃

當 DAO 上的某個計畫達到 51% 的 DAO 幣支持，該募資計畫即成功

募資計畫會將所得利潤回饋給投資者

DAO

DAO 的智能合約為了防範 51% 攻擊 (有攻擊者能夠控制一半以上的 DAO 幣)

或是有投資者在經過至少一週的討論以後，依然不想投資該募資計劃

則該投資者可以選擇退出該投資計畫

即是 DAO 對 “多數暴政” 的防範機制

投資者可於任何時刻退出，並獲得該時刻應得的投資利潤

DAO

該防範機制是不同意的投資者可以發起 “splitDAO”

將自己的 “DAO幣” 轉換為另外一個 “新的智能合約”，即是 “childDAO”

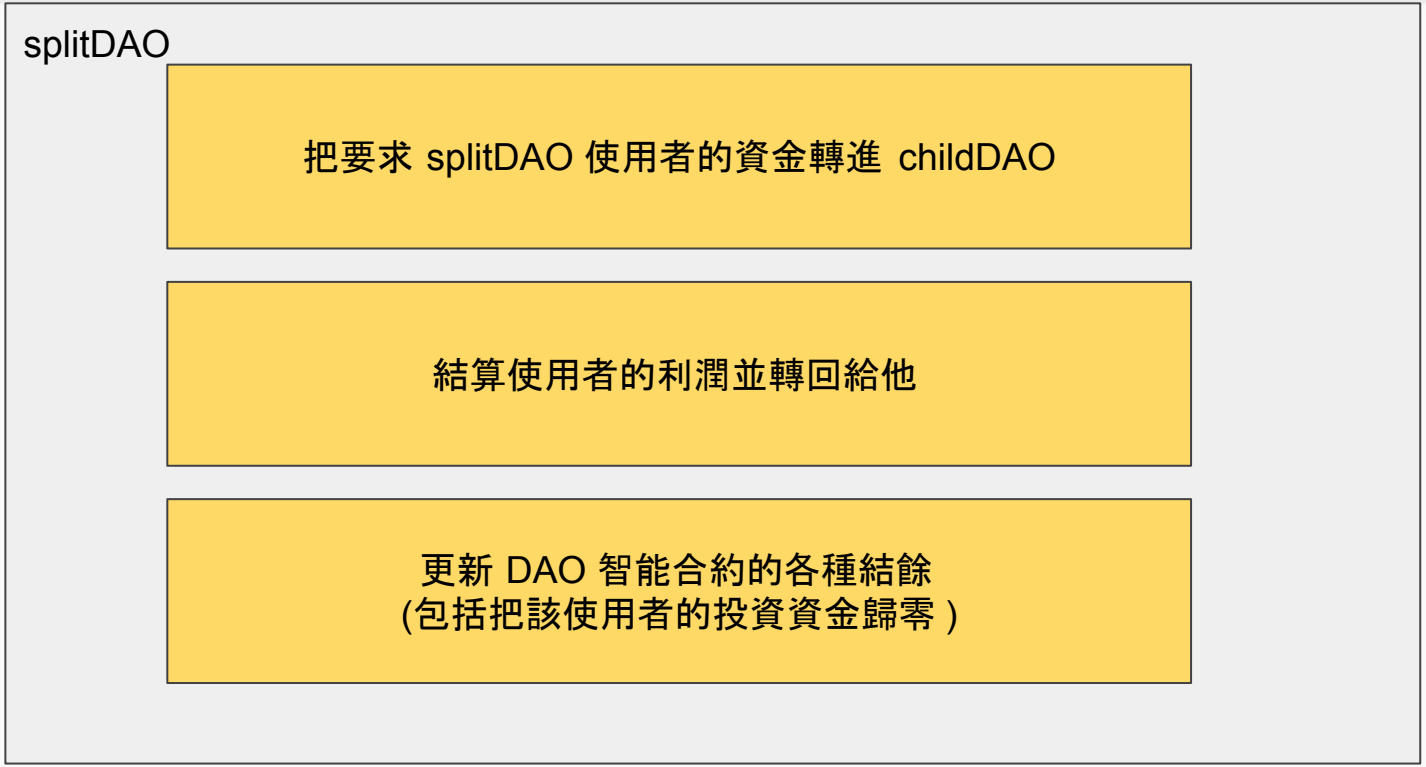
childDAO 並不屬於原來的 DAO幣，因此不會參與那個募資計劃

Race to empty 攻擊

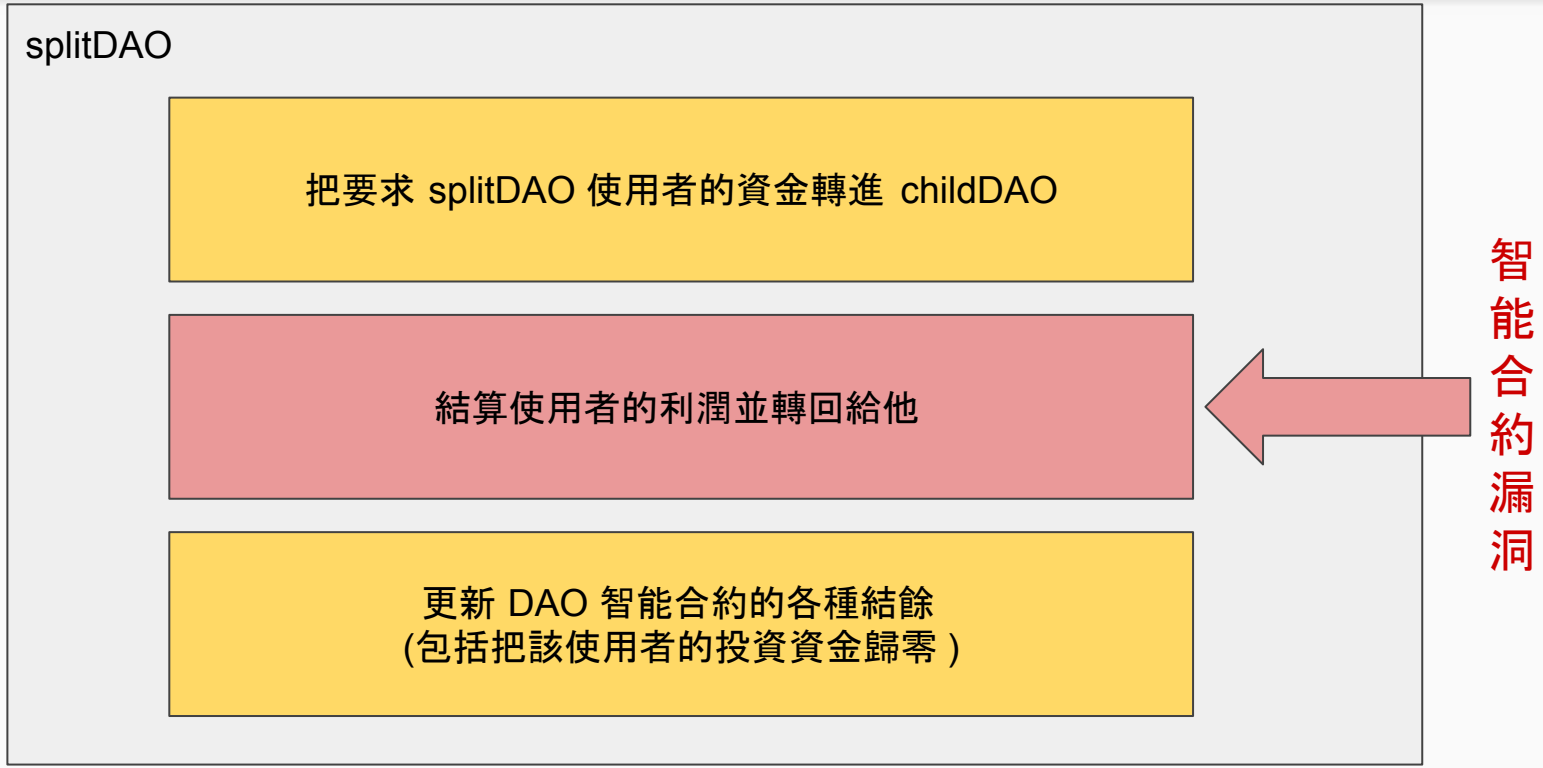
但 DAO 的智能合約在設計上有漏洞，導致攻擊者可以發動一種攻擊

“Race to Empty”

Race to empty 攻擊



Race to empty 攻擊

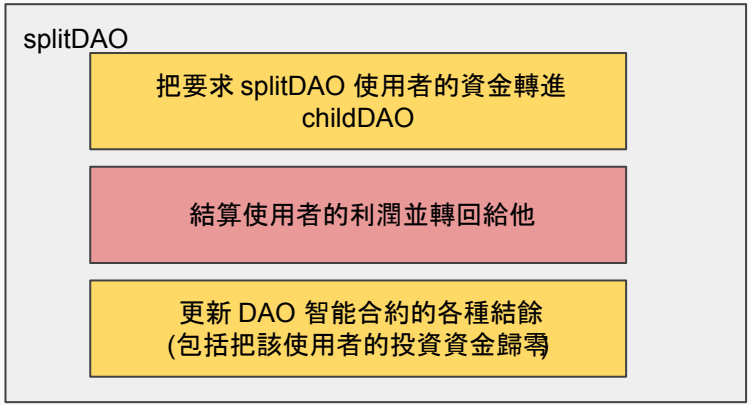


Race to empty 攻擊

攻擊者利用這個漏洞，在程式運作到

紅色區塊的時候，讓程式暫停在這裡，並且

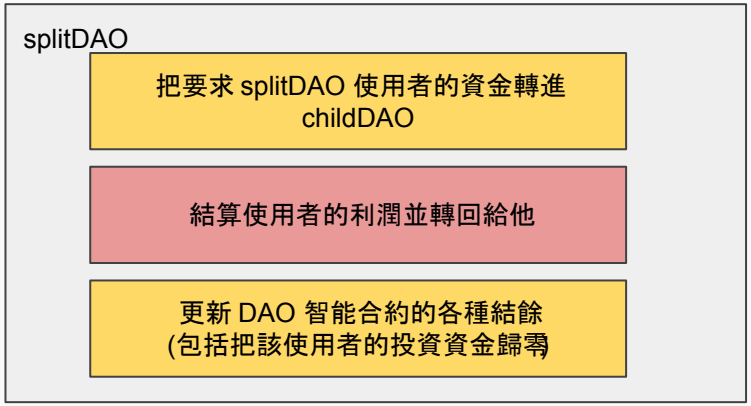
再次呼叫了 splitDAO



Race to empty 攻擊

於是攻擊者就可以一直要求智能合約募資計畫

反覆的退相同數目的 DAO幣到自己的 childDAO



Race to empty 攻擊

打個比方

Alice 眼睛不好

Alice 跟小明借了 100 元，今天小明來跟 Alice討債

Alice 剛從錢包抓出 100 元的時候，小明就把它偷走，並且跟 Alice 說：

“欸你剛剛沒抓到錢，你還沒還我 100 元” (splitDAO還沒運行到清算)

小明重複這個步驟很多次 (重複呼叫 splitDAO)

Race to empty 程式說明

以下將對程式碼進行說明

ref: http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/

http://vessenes.com/deconstructing-thedao-attack-a-brief-code-tour/

http://ethfans.org/posts/115 (上一篇的簡體翻譯)

http://ethfans.org/topics/419

http://ethfans.org/posts/116

splitDAO

```
function splitDAO(
        uint _proposalID,
        address _newCurator
    ) noEther onlyTokenholders returns (bool _success) {

        ...

        // Move ether and assign new Tokens
        uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;
        if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
            throw;

        ...

        // Burn DAO Tokens
        Transfer(msg.sender, 0, balances[msg.sender]);
        withdrawRewardFor(msg.sender); // be nice, and get his rewards
        totalSupply -= balances[msg.sender];
        balances[msg.sender] = 0;
        paidOut[msg.sender] = 0;
        return true;
    }
```

splitDAO

```
function splitDAO(
        uint _proposalID,
        address _newCurator
    ) noEther onlyTokenholders returns (bool _success) {

        ...

        // Move ether and assign new Tokens
        uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;
        if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
            throw;

        ...

        // Burn DAO Tokens
        Transfer(msg.sender, 0, balances[msg.sender]);
        withdrawRewardFor(msg.sender); // be nice, and get his rewards
        totalSupply -= balances[msg.sender];
        balances[msg.sender] = 0;
        paidOut[msg.sender] = 0;
        return true;
    }
```

splitDAO

```
function splitDAO(
        uint _proposalID,
        address _newCurator
    ) noEther onlyTokenholders returns (bool _success) {

        ...

        // Move ether and assign new Tokens
        uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;
        if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
            throw;

        ...

        // Burn DAO Tokens
        Transfer(msg.sender, 0, balances[msg.sender]);
        withdrawRewardFor(msg.sender); // be nice, and get his rewards
        totalSupply -= balances[msg.sender];
        balances[msg.sender] = 0;
        paidOut[msg.sender] = 0;
        return true;
    }
```

把投資金額轉成 childDAO

splitDAO

```
function splitDAO(
        uint _proposalID,
        address _newCurator
    ) noEther onlyTokenholders returns (bool _success) {

        ...

        // Move ether and assign new Tokens
        uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;
        if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
            throw;

        ...

        // Burn DAO Tokens
        Transfer(msg.sender, 0, balances[msg.sender]);
        withdrawRewardFor(msg.sender); // be nice, and get his rewards
        totalSupply -= balances[msg.sender];
        balances[msg.sender] = 0;
        paidOut[msg.sender] = 0;
        return true;
    }
```

把投資金額轉成 childDAO

歸還利潤

splitDAO

```
function splitDAO(
    uint _proposalID,
    address _newCurator
) noEther onlyTokenholders returns (bool _success) {

    ...

    // Move ether and assign new Tokens
    uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitData[0].totalSupply;
    if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false)
        throw;

    ...

    // Burn DAO Tokens
    Transfer(msg.sender, 0, balances[msg.sender]);
    withdrawRewardFor(msg.sender); // be nice, and get his rewards
    totalSupply -= balances[msg.sender];
    balances[msg.sender] = 0;
    paidOut[msg.sender] = 0;
    return true;
}
```

把投資金額轉成 childDAO

歸還利潤

清算

有問題的部分：歸還利潤

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])
        throw;

    uint reward = (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];

    reward = rewardAccount.balance < reward ? rewardAccount.balance : reward;

    if (!rewardAccount.payOut(_account, reward))
        throw;
    paidOut[_account] += reward;
    return true;
}
```

有問題的部分：歸還利潤

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])
        throw;

    uint reward = (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];

    reward = rewardAccount.balance < reward ? rewardAccount.balance : reward;

    if (!rewardAccount.payOut(_account, reward))
        throw;
    paidOut[_account] += reward;
    return true;
}
```

有問題的部分：歸還利潤

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
    if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_account])
        throw;

    uint reward = (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_account];

    reward = rewa                    e < reward ? rewardAccount.balance : reward;

    if (!rewardAccount.payOut(_account, reward))
        throw;
    paidOut[_account] += reward;
    return true;
}
```

出問題的地方

payOut

```
function payOut(address _recipient, uint _amount) returns (bool) {
    if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
        throw;
    if (_recipient.call.value(_amount)()) {
        PayOut(_recipient, _amount);
        return true;
    } else {
        return false;
    }
}
```

出問題的 payOut

```
function payOut(address _recipient, uint _amount) returns (bool) {
    if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
        throw;
    if (_recipient.call.value(_amount)()) {
        PayOut(_recipient, _amount);
        return true;
    } else {
        return false;
    }
}
```

出問題的 payOut

```
function payOut(address _recipient, uint _amount) returns (bool) {
    if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
        throw;
    if (_recipient.call.value(_amount)()) {
        PayOut(_recipient, _amount);
        return true;
    } else {
        return false;
    }
}
```

把利潤退回去

退回利潤

因為每個使用者的錢包可以設定默認函數

所以可以在錢包收到這筆利潤的時候，再次呼叫 DAO 的 splitDAO

此時第一個 splitDAO 還在等錢包的默認函數結束

所以就卡住了

退回利潤

此時第一個 splitDAO 還在等錢包的默認函數結束

所以就卡住了

而且還有新的 splitDAO 呼叫，開始落入迴圈

“Recursive calling vulnerability”

其他問題

- 轉送代幣的部分用 recipient.call.value()() 並不指定最多能用多少 gas，所以這個 execution 能使用所有的 gas

  改進 => 使用 recipient.send，gas 只有預設的 2300 gas

- 即使限制 gas 的數量，攻擊者仍然可以針對這個弱點繼續攻擊

  改進 => 先做清算才做實際轉送

區塊鏈應用 → 天龍八部

g-coin.org

支付結算系統

顧客忠誠計劃

crowdsourcing 平台

私募平台

票據平台

去中心化交易所