HOW TO MANAGE YOUR ETHER
以太坊錢包介紹與分析

Ethereum research
林修平（Hsiu-Ping Lin）

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③ Vault
金庫

④ Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤ Lesson learned
從中獲得的經驗

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③ Vault
金庫

④ Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤ Lesson learned
從中獲得的經驗

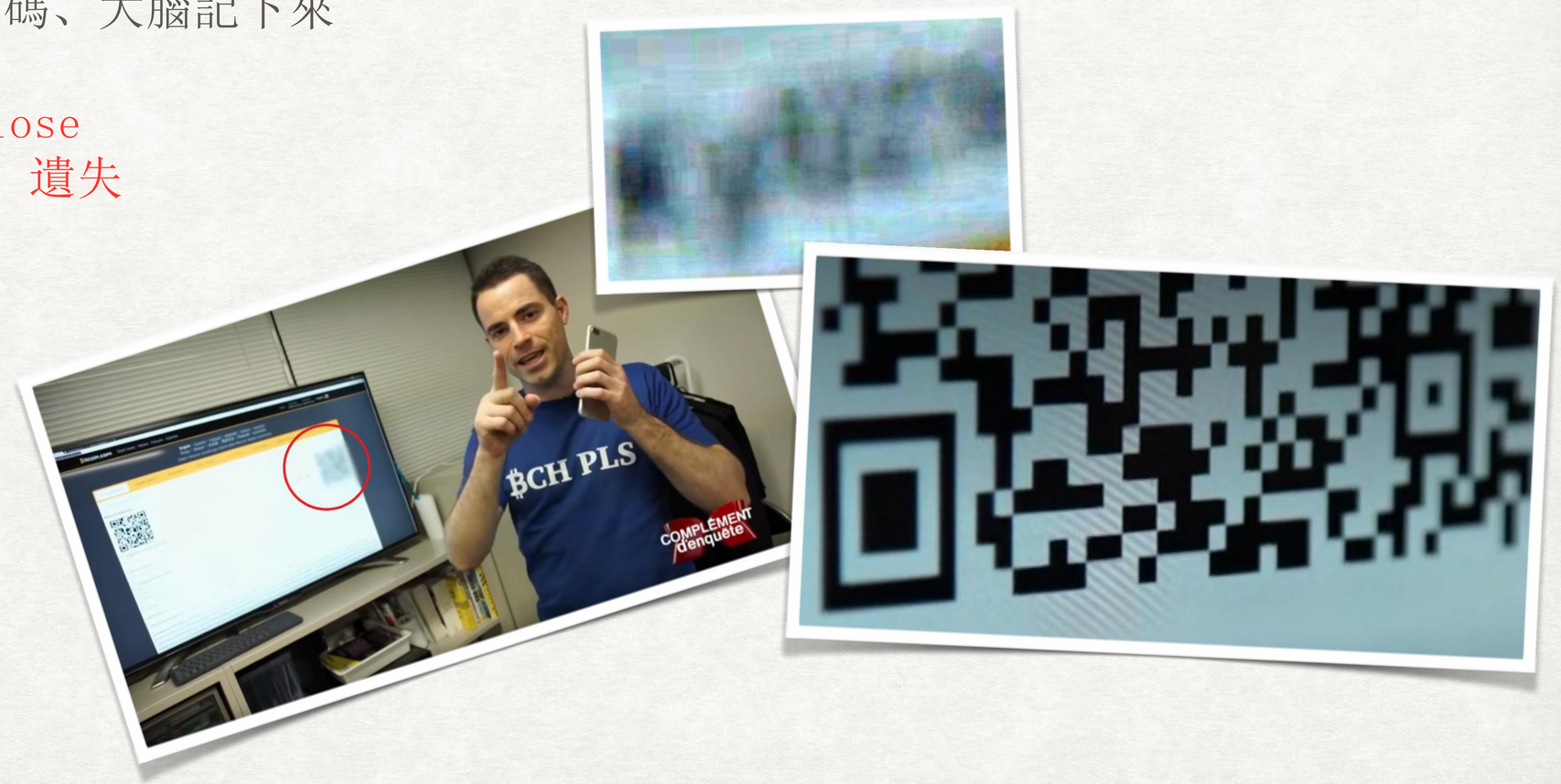
HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

- Write it down, QR code or Remember it
  抄下來、二維碼、大腦記下來
- Save it in computer
  存在電腦裡
- Wallet app, Exchange
  錢包軟體、交易所
- Hardware wallet
  硬體錢包
- Multi-sig wallet
  多簽錢包

HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

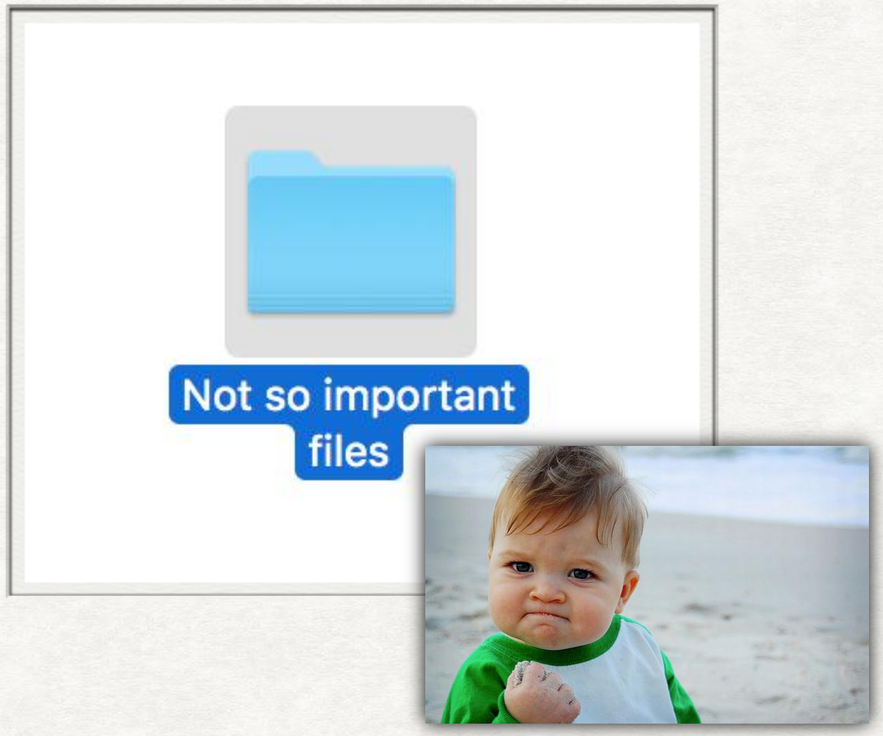
- Write it down, QR code or Remember it
  抄下來、二維碼、大腦記下來
  - Easy to lose
    容易失竊、遺失



https://medium.freecodecamp.org/lets-enhance-how-we-found-rogerkver-s-1000-wallet-obfuscated-private-key-8514e74a5433

HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

- Save it in computer
  存在電腦裡
  - Easier to hide it
    比抄在紙上好藏
  - Risk of nasty virus
    電腦病毒很難預防
- Air-gapped
  不連網



HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

- Wallet app, Exchange
  錢包軟體、交易所
  - UI
    使用者介面
  - Easy to manage
    方便管理
  - Trust the app
    信任app
  - Trust the exchange; no real ownership
    信任交易所，交出所有權

HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

- Hardware wallet
  硬體錢包
  - Fairly easy to manage
    方便管理
  - Protected
    受硬體保護
    - similar to Air-gapped device
      和不連網裝置相似
  - Relatively expensive
    成本相對地昂貴
  - Trust the manufacturer
    信任硬體製造商

HOW DO YOU MANAGE YOUR CRYPTOCURRENCY?
有哪些管道來管理自己的虛擬貨幣？

- Multi-sig wallet
  多簽錢包
  - Lower the risk
    降低風險
  - More complicate procedure
    較複雜的交易程序

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③ Vault
金庫

④ Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤ Lesson learned
從中獲得的經驗

MULTI-SIG
多簽錢包

MULTI-SIG
多簽錢包

- Multi-signature
  多重簽章
  - M of N signatures required
    N個簽名裡面的需要M個簽名許可
  - More secure than single key
    比單一私鑰更安全
  - Used by yourself
    自己使用
  - Used by multi party
    多方共用

MULTI-SIG
多簽錢包

- M-N Multi-sig in Bitcoin
  比特幣裡的多重簽章

MULTI-SIG
多簽錢包

- M-N Multi-sig in Bitcoin
  比特幣裡的多重簽章
  - Generate a multi-sig
    產生多重簽章
    - Get N public keys
      蒐集N把公鑰
    - Generate a multi-sig public key using these N public keys
      用這N把公鑰產生一個多簽公鑰
      - Addresses that start with a 3
        多簽公鑰的地址開頭為3

MULTI-SIG
多簽錢包

- M-N Multi-sig in Bitcoin
  比特幣裡的多重簽章
  - Spend from a multi-sig
    花費多重簽章的錢
    - Generate the transaction that spend from multi-sig public key
      產生一筆交易，錢的來源為多重簽章
    - sign the transaction and pass the signed transaction to next signer
      對這筆交易簽名再將簽好的交易傳給下一個人
    - after signed by M public keys, broadcast the transaction
      當超過M個人簽過了就將交易廣播出去

MULTI-SIG
多簽錢包

- M-N Multi-sig in Bitcoin
  比特幣裡的多重簽章
  - Involve more work
    更為複雜
  - More factors that can delay the completion of the transaction
    有更多因素會造成交易的延遲
    - More human error
      更容易產生人為錯誤
    - Network error
      更可能受到網路問題影響

MULTI-SIG
多簽錢包

- M-N Multi-sig in Bitcoin
  比特幣裡的多重簽章
  - What if the signed transaction is lost during signing process?
    如果簽名過程中遺失了?
    - Multicast to all other signers to prevent single point of failure
      簽完後傳送給其他所有尚未簽名的人來避免因為單一遺失而中斷

MULTI-SIG
多簽錢包

- M-N Multi-sig in Ethereum
  以太坊的多重簽章

MULTI-SIG
多簽錢包

- M-N Multi-sig in Ethereum
  以太坊的多重簽章
  - Generate
    產生多重簽章
    - Get N public keys
      蒐集N把公鑰
    - Deploy a contract with these N public keys as owners
      部署一個合約且將這N把公鑰設為擁有人
    - Deposit into the contract, as many times as you want
      存錢進此合約，且無次數限制

MULTI-SIG
多簽錢包

- M-N Multi-sig in Ethereum
  以太坊的多重簽章
  - Spend from a multi-sig
    花費多重簽章的錢
    - No limit on how many times you can withdraw
      領錢也無次數限制
    - M owners each generate a transaction that withdraw from the contract
      M個擁有人各自產生一筆花錢的交易
    - They each sign and broadcast the transaction
      擁有人對各自的交易簽名並廣播交易

MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章
  - You can change ownership anytime without deploying a new contract
    可隨時改變擁有人而不需部署一個新的合約
    - Reusable
      可重複使用

MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章
  - What if someone withheld the transaction and decide to sign and broadcast it later?
    如果某個擁有人當下不簽，之後才簽
    - Use an incremental value(Nonce) to identify valid transaction
      需要有一個不斷增加的值來識別交易的有效性

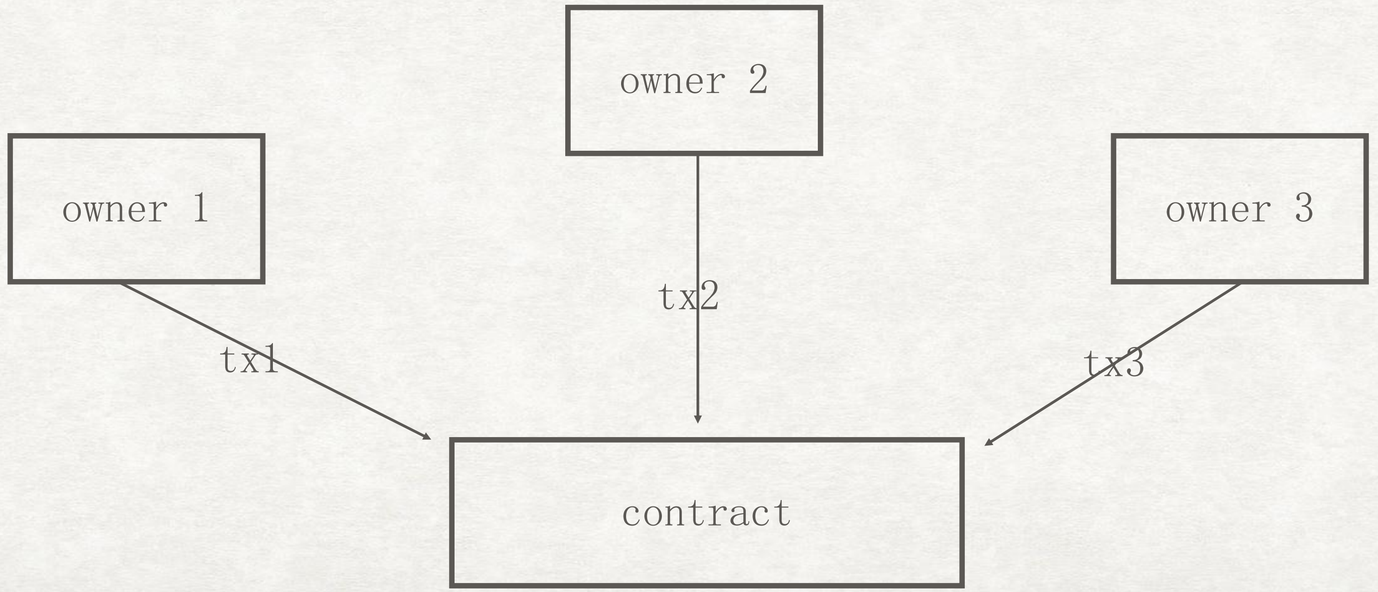
MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章
  - Small improvement on transaction cost: Save cost by aggregating signed transactions into one
    降低交易成本的小技巧：將擁有人的簽名交易集中成一個交易
    - ecrecover()
      - recover the signer from the signature
        從簽名資料還原出簽名人的地址

MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章
  - Save the cost by aggregating the signed transactions into one
    將擁有人的簽名交易集中成一個交易

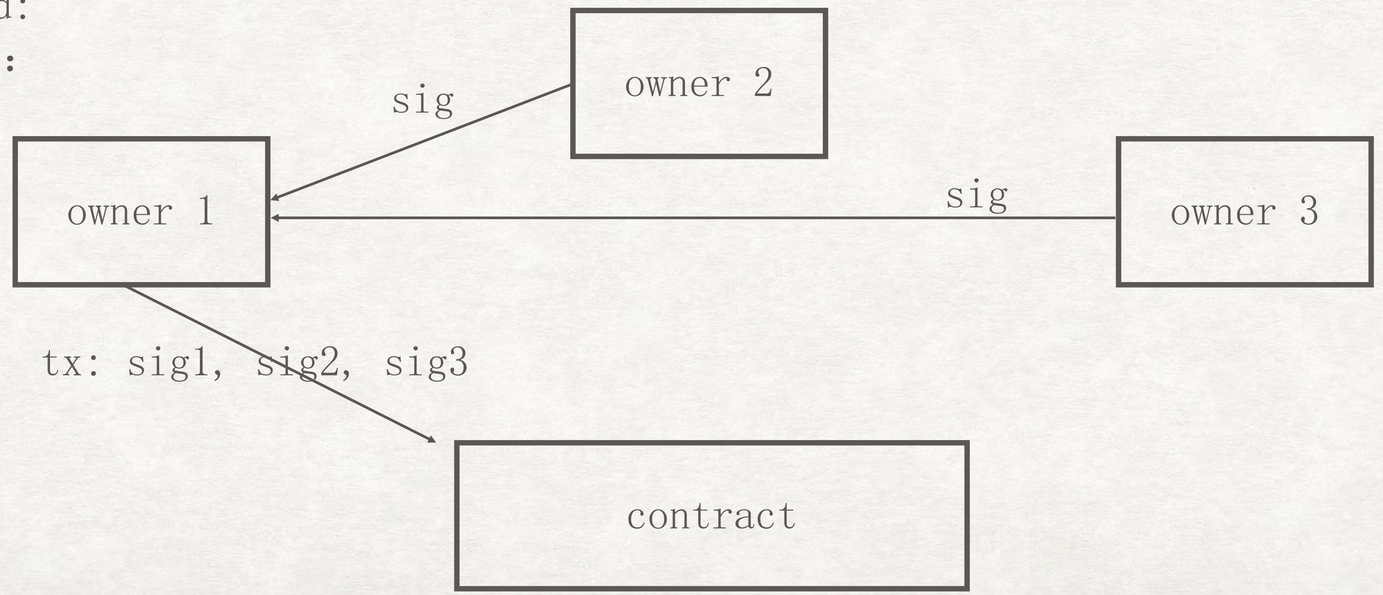
Normal:
正常情況:



MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章
  - Save the cost by aggregating the signed transactions into one
    將擁有人的簽名交易集中成一個交易

Aggregated:
集中交易後:



MULTI-SIG
多簽錢包

- Multi-sig example in Ethereum
  以太坊的多重簽章範例
  - https://github.com/ethereum/dapp-bin/blob/master/wallet/wallet.sol

MULTI-SIG
多簽錢包

- Multi-sig example in Ethereum
  以太坊的多重簽章範例
  - Operations type
    操作類型
    - Add/Remove/Change owners
      新增、移除、改變擁有人
    - Change requirement(M of N)
      改變多重簽章的門檻
    - Withdraw
      領錢

MULTI-SIG
多簽錢包

- Multi-sig example in Ethereum
  以太坊的多重簽章範例
  - Each operation comes with an unique ID
    每個操作都會有獨特的識別碼
    - ID: hash the informations related to the operation, like old/new owner address
      識別碼：對該操作的相關資訊做雜湊得到的值
      - withdraw ID: hash(receiver + amount + block number)
        領錢識別碼：對收錢方的地址、總數和區塊號碼做雜湊
      - operations with same info will have the same ID
        有相同相關資訊的操作會有相同的識別碼

MULTI-SIG
多簽錢包

- Multi-sig example in Ethereum
  以太坊的多重簽章範例
  - Each operation stays in a pool as a pending operation before receiving enough confirmation from owners
    每個不同的操作會待在一個等待區裡，直到該操作收到足夠的確認
    - pool is cleaned up if owners or requirement are changed
      如果擁有人有變動或簽章門檻有變動，等待區會被清空

MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章範例
  - Features like Daily limit
    還有像是每日數量限制的功能
    - No multi-sig required if amount spent is below daily limit
      如果當日花費還沒超過每日數量限制，則不需要多簽許可
    - Set/Reset limit is multi-sig required
      設定每日數量限制的值需要多簽許可

MULTI-SIG
多簽錢包

- Multi-sig in Ethereum
  以太坊的多重簽章範例
  - You can customize your wallet with little effort!
    不需很複雜的代碼就能客製化自己的功能
    - Turing complete FTW!!!

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③Vault
金庫

④ Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤ Lesson learned
從中獲得的經驗

VAULT
金庫

- Greater amount of money
  較大的金額
- Longer time to withdraw
  較長的領錢時間
- You can steal the key but you can’t steal the money
  小偷可以偷走私鑰，但偷不走錢

VAULT
金庫

- A vault has
  金庫相關資訊包含
  - Owner
    擁有人
  - Lock time: seconds/hours/days
    等待時間：秒、小時、日

VAULT
金庫

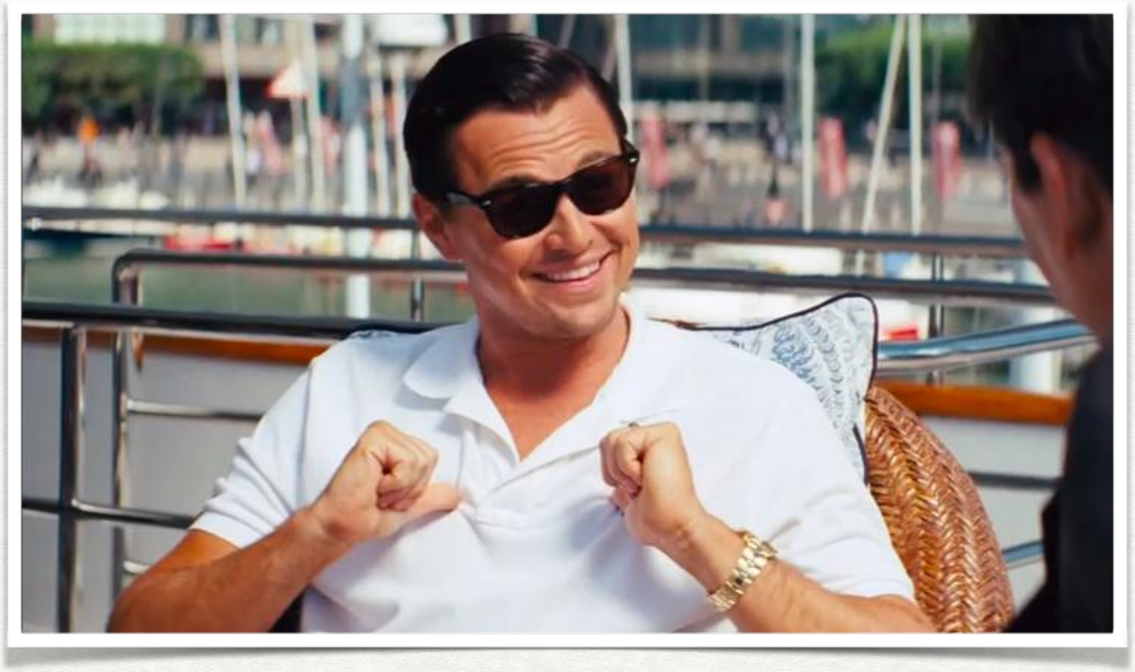
- A vault has
  金庫相關資訊包含
  - requestWithdrawal(amount, sendTo)
  - cancel(withdrawalIndex)
    - valid through lock time
      在等待時間內都可取消
  - finalize(withdrawalIndex)
    - after lock time
      在等待時間後才能完成領錢

VAULT
金庫

- If nothing wrong happened
  沒出錯
  - requestWithdrawal -> wait -> finalize
- Key stolen!
  私鑰被偷
  - You requestWithdrawal -> thief cancel
  - Thief requestWithdrawal -> you cancel
  - Thief will waste their time on withdraw-and-cancel game
    小偷在這個情況下等於浪費時間

VAULT
金庫

- But what if you happened to need the money and thief also happened to know that :-P
  但如果小偷剛好知道你急需這筆錢呢



VAULT
金庫

- But what if you happened to need the money and thief also happened to know that :-P
  但如果小偷剛好知道你急需這筆錢呢
  - The thief sets up a contract with a function splitMoney
    小偷部署一個分錢用的合約
    - X% goes to thief and 1-X% goes to you
      進到合約的錢百分之X給小偷，剩下的給你
  - Also gives you the contract code and contract address so you can verify it’s correctness
    同時他也把代碼和合約位置給你，讓你驗證合約的正確性
  - Finally, requestWithdrawal which calls the splitMoney function of the contract
    接著觸發領錢，並把錢送到分錢合約，分錢合約再依比例拆分

VAULT
金庫

- Vault with third party
  有第三方加入的金庫
  - Let’s call it server
    這裡稱第三方為server

VAULT
金庫

- requestWithdrawal(amount, sendTo)
  - Both user and server can call
    擁有人和第三方都可呼叫
  - requestWithdrawal by server requires a longer wait time
    第三方的領錢請求需要等待較長的時間
- finalize(withdrawalIndex)
- cancel(withdrawalIndex)
  - Only user can call
    只有擁有人可以觸發

VAULT
金庫

- requestServerWithdrawal(amount, sendTo)
  - Only user can call
    只有擁有人可以觸發
  - No lock time
    沒有等待時間
- cancelServerWithdrawal(serverWithdrawalIndex)
- finalizeServerWithdrawal(serverWithdrawalIndex)

VAULT
金庫

- requestServerWithdrawal(amount, sendTo)
- cancelServerWithdrawal(serverWithdrawalIndex)
  - Only server can call
    只有第三方可以觸發
- finalizeServerWithdrawal(serverWithdrawalIndex)

VAULT
金庫

- requestServerWithdrawal(amount, sendTo)
- cancelServerWithdrawal(serverWithdrawalIndex)
- finalizeServerWithdrawal(serverWithdrawalIndex)
  - Only user can call
    只有擁有人可以觸發

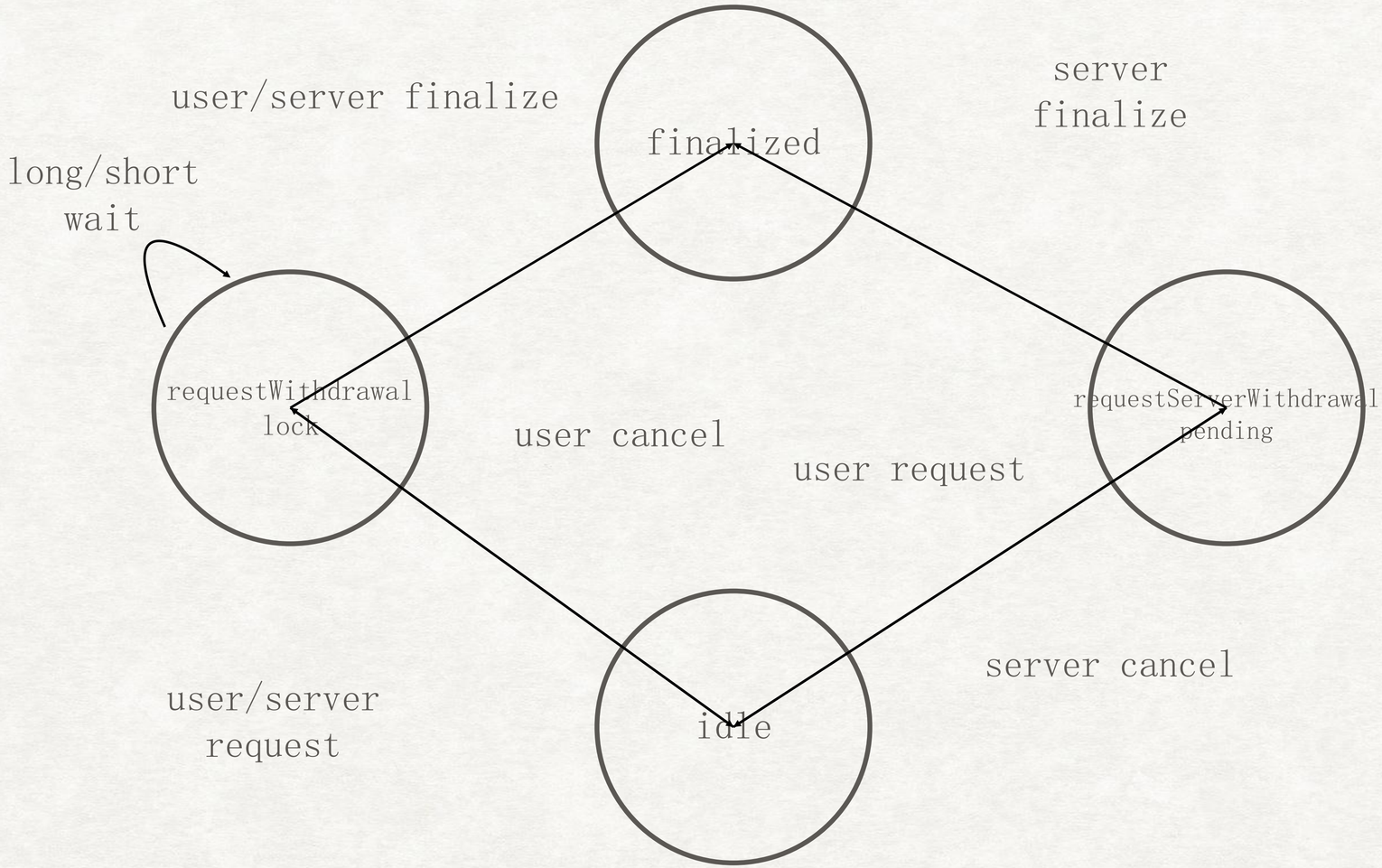
VAULT
金庫

- If nothing wrong happened
  沒出錯
  - requestWithdrawal -> short wait -> finalize
- 1. Key lost!
  私鑰忘了
  - server requestWithdrawal -> long wait -> finalize

VAULT
金庫

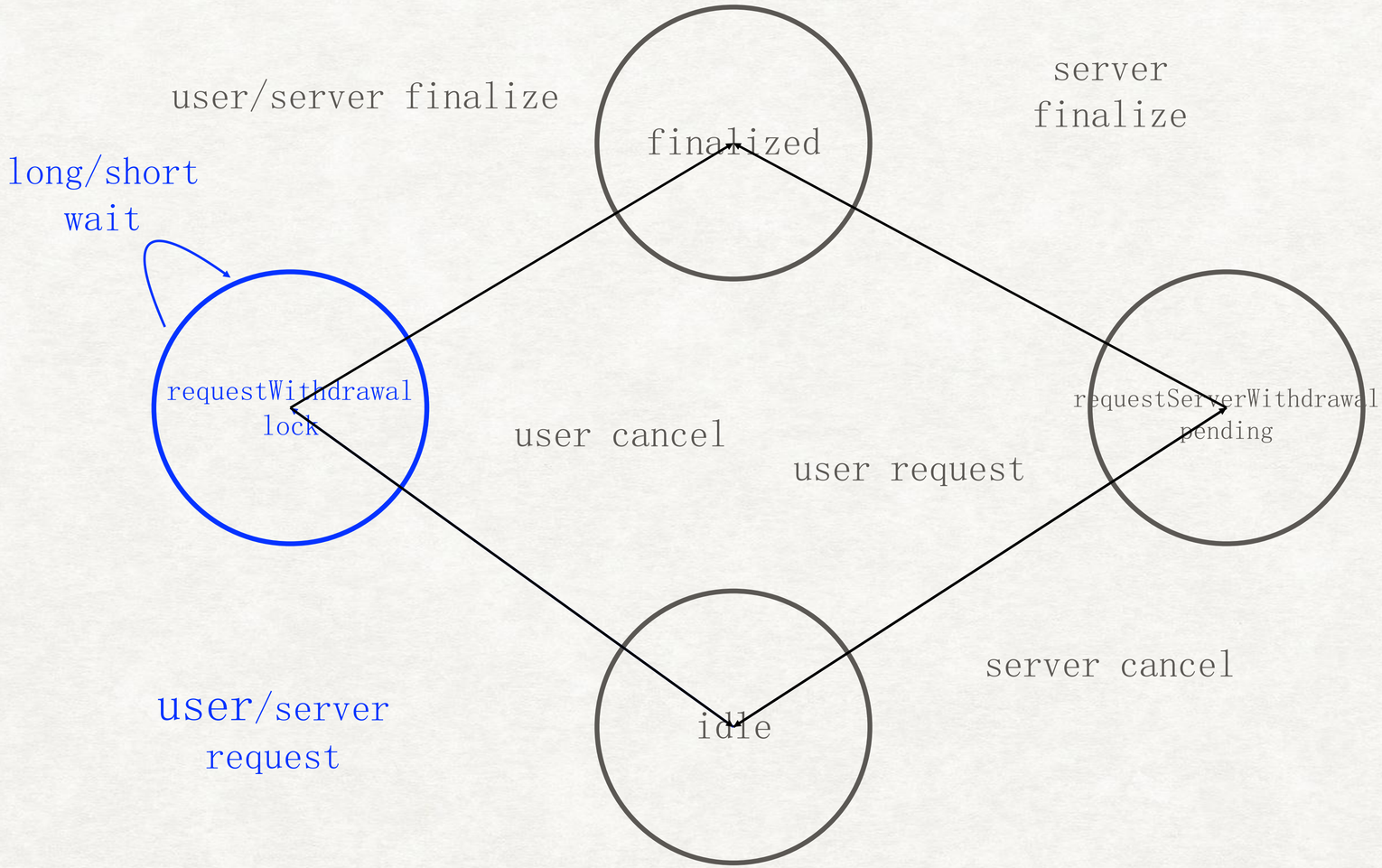
- 2. Key stolen!
  私鑰被偷
  - Thief requestWithdrawal -> you cancel
  - You requestServerWithdrawal -> server finalizeServerWithdrawal
- 3. Server compromised!
  第三方被盜
  - server requestWithdrawal -> you cancel
  - you requestWithdrawal -> short wait

VAULT
金庫


user/server finalize
finalized
server
finalize
long/short
wait
requestWithdrawal
lock
user cancel
user request
requestServerWithdrawal
pending
server cancel
user/server
request
idle

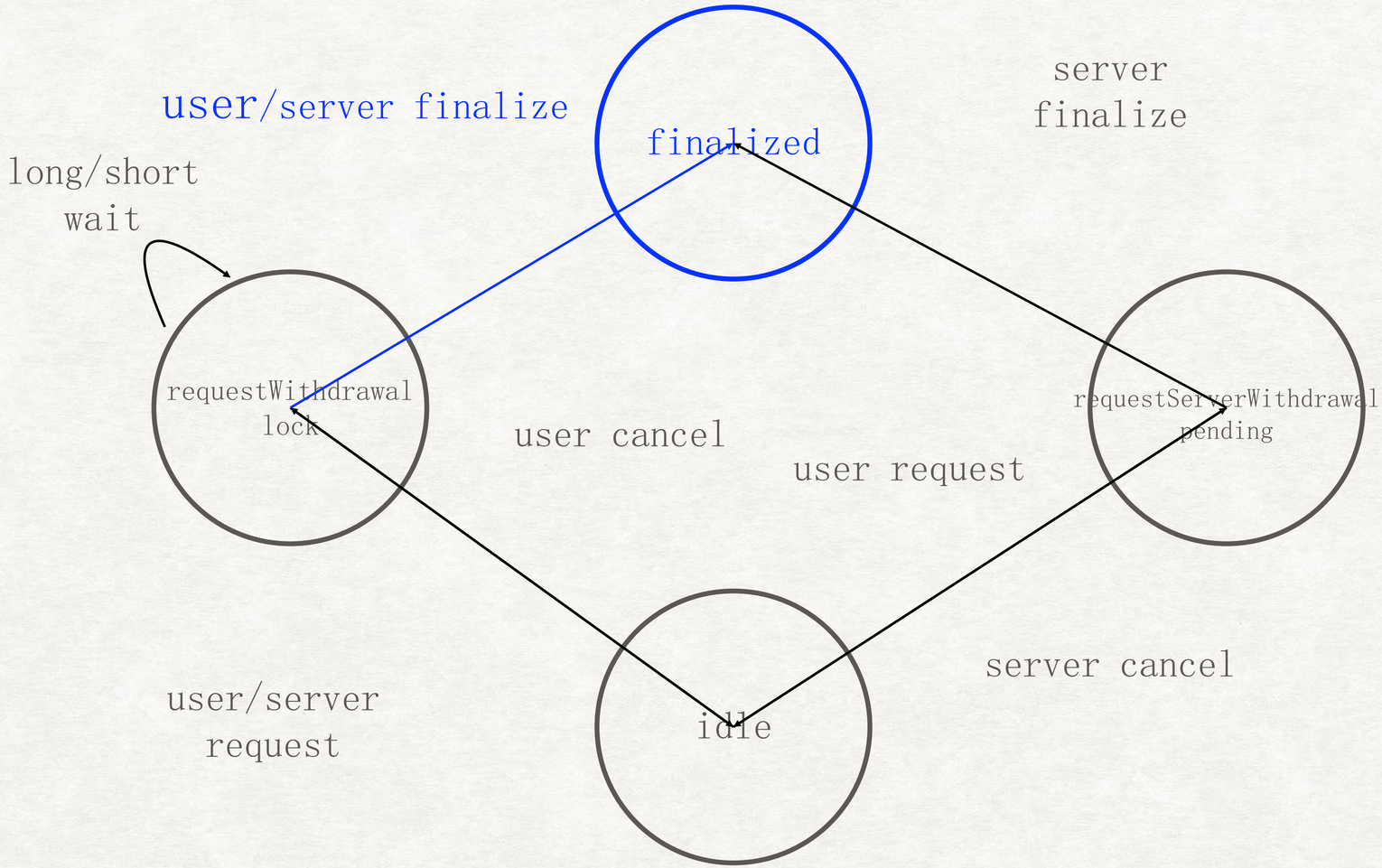
VAULT
金庫

- Normal:
  一般情況



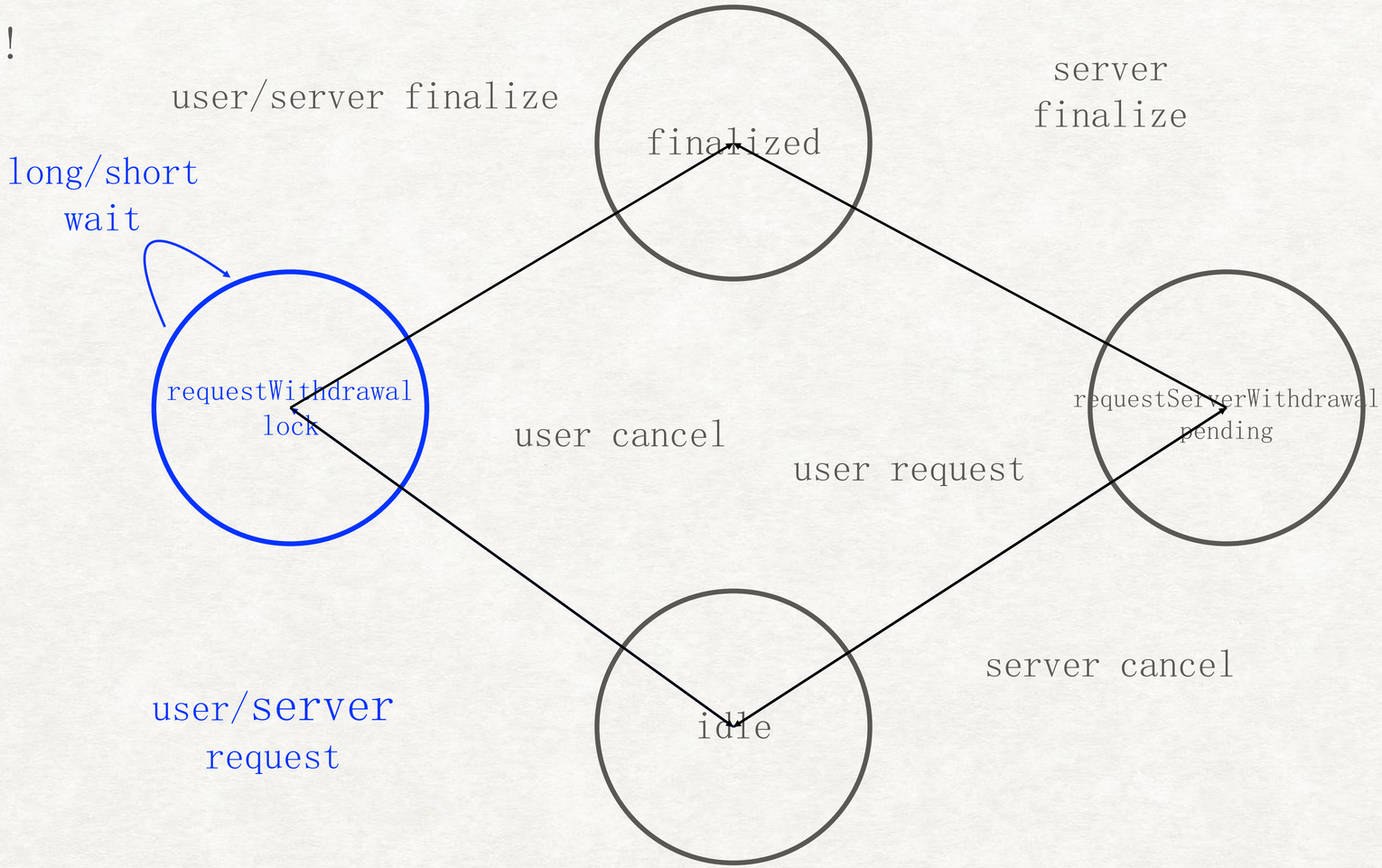
VAULT
金庫

- Normal:
  一般情況



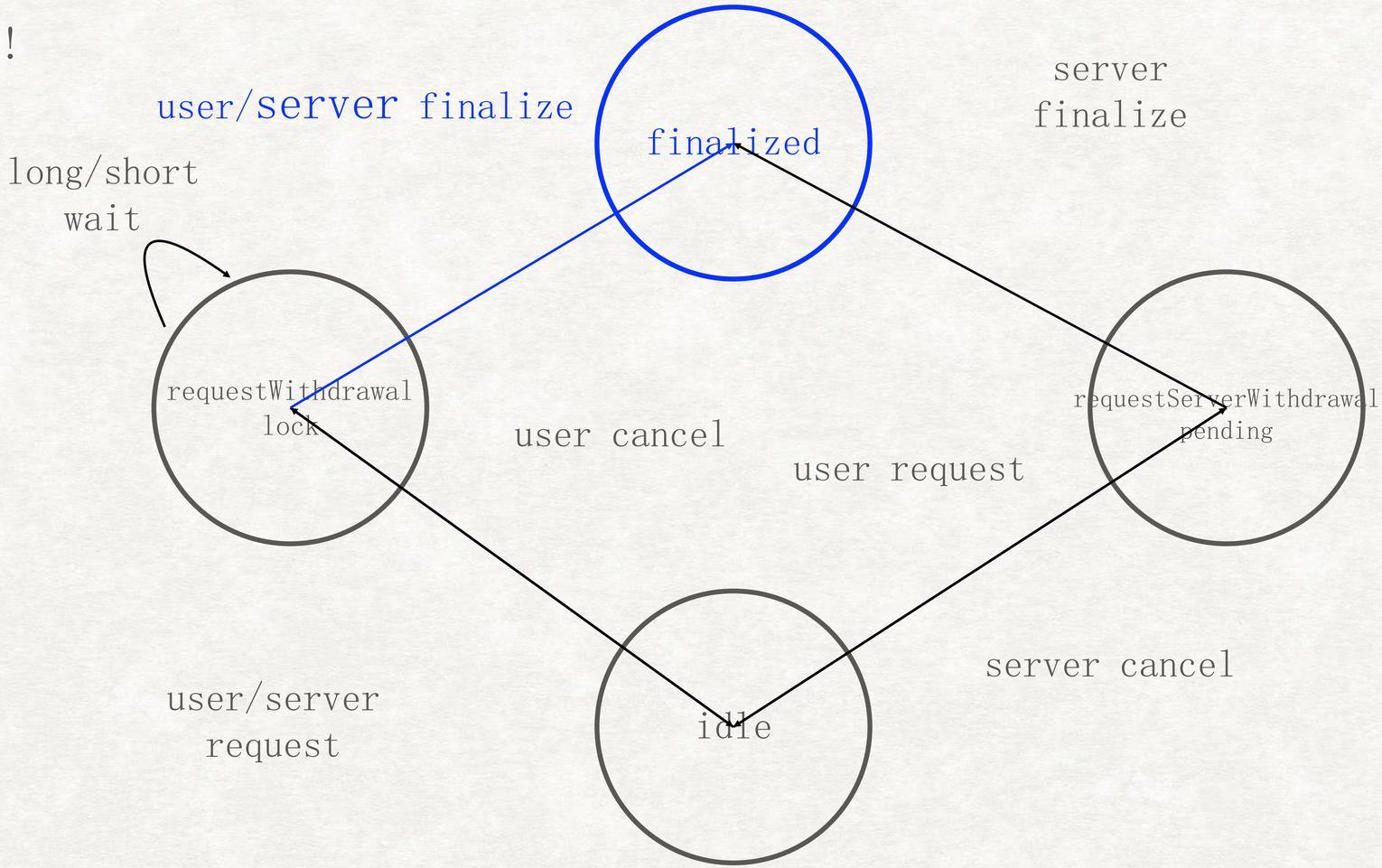
VAULT
金庫

- 1. Key lost!
  私鑰忘了



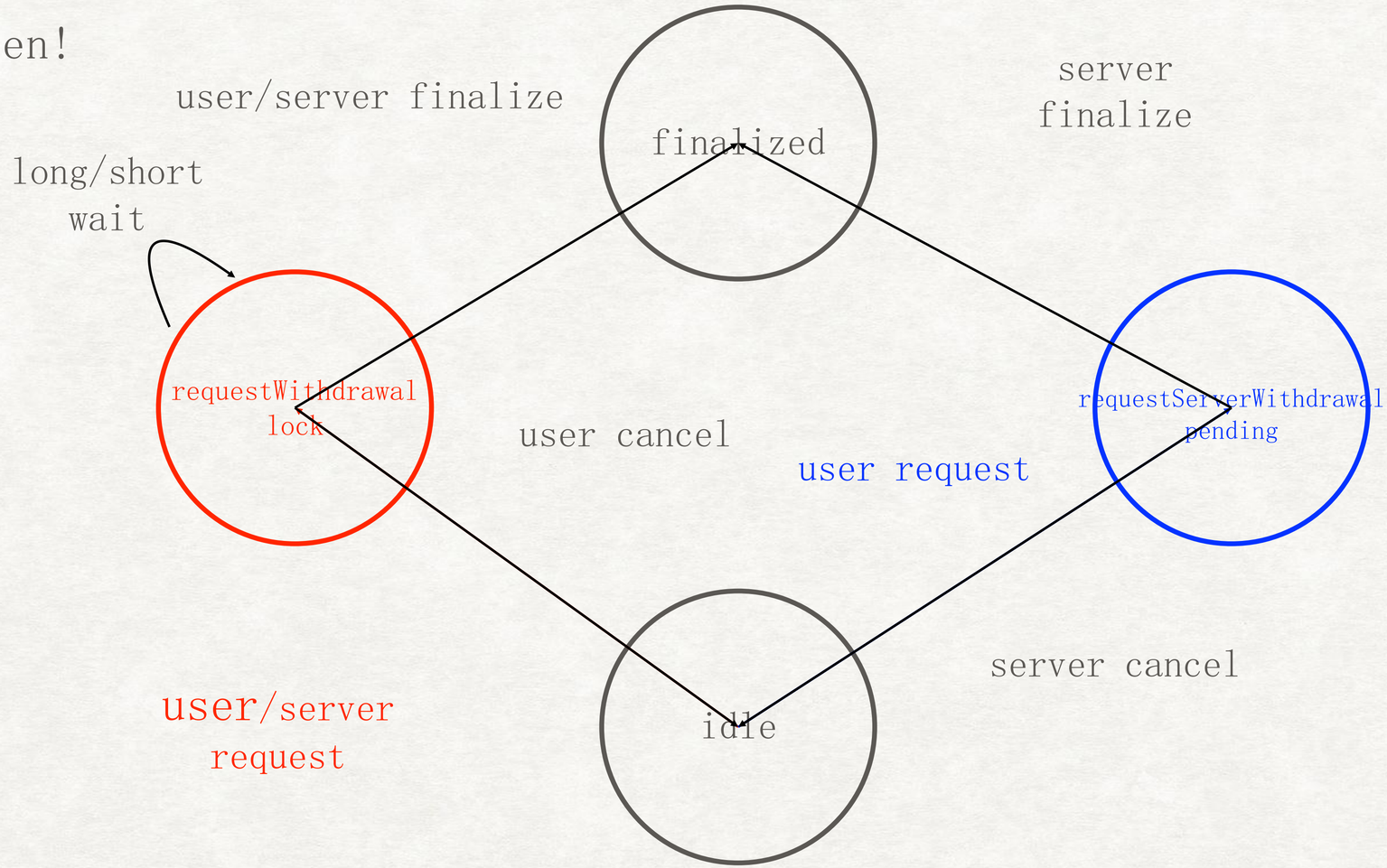
VAULT
金庫

- 1. Key lost!
  私鑰忘了



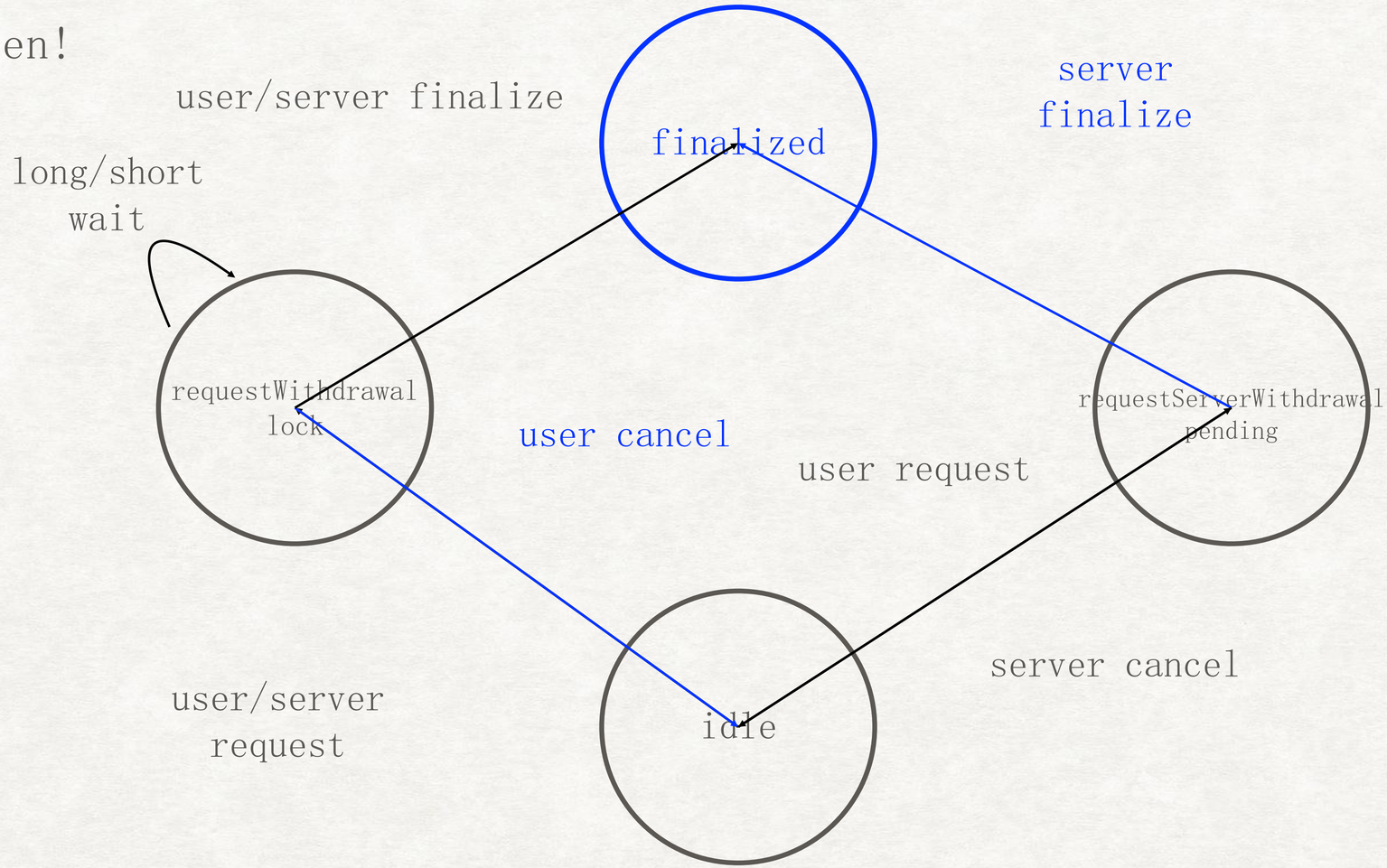
VAULT
金庫

- 2. Key stolen!
  私鑰被偷



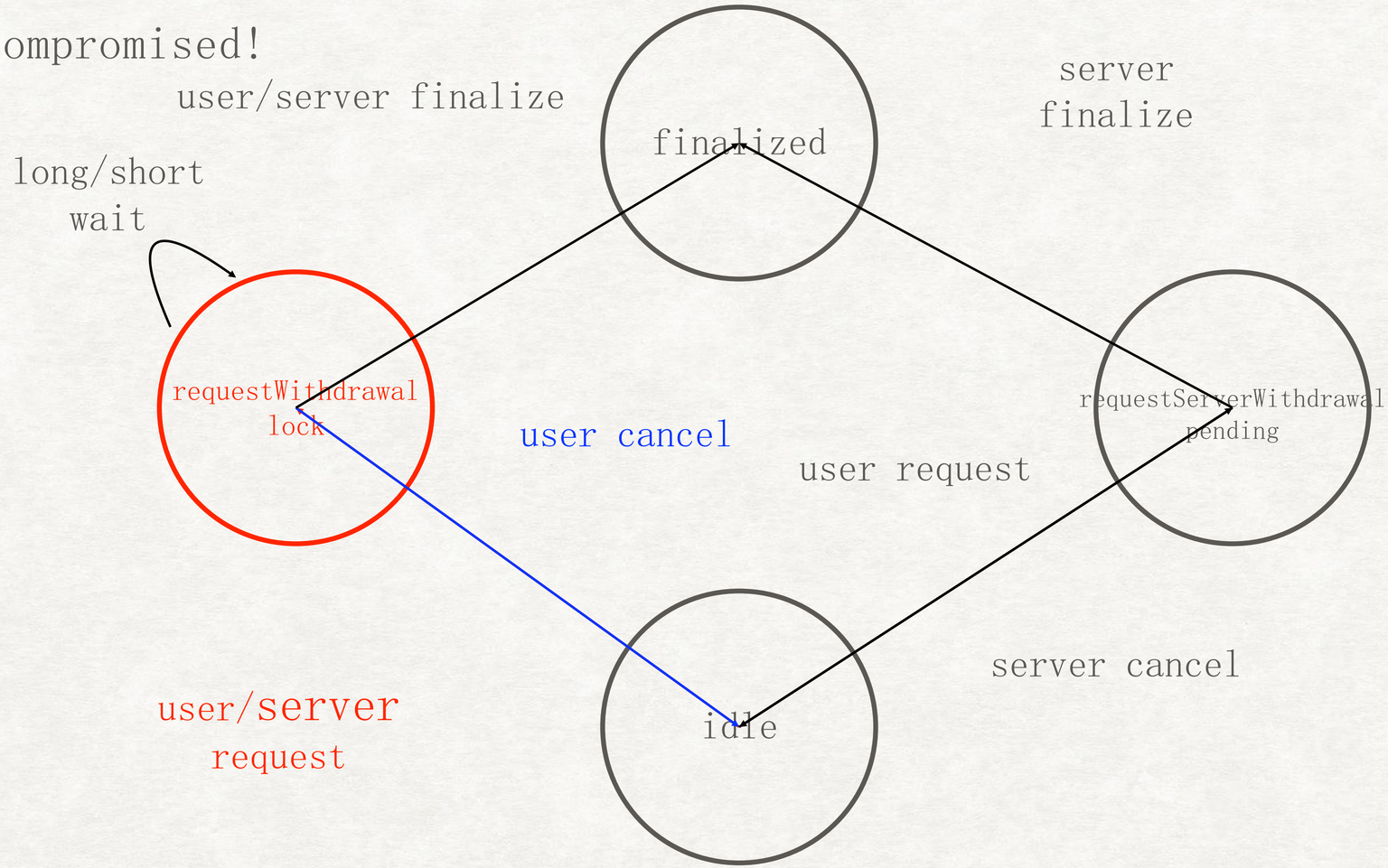
VAULT
金庫

- 2. Key stolen!
  私鑰被偷



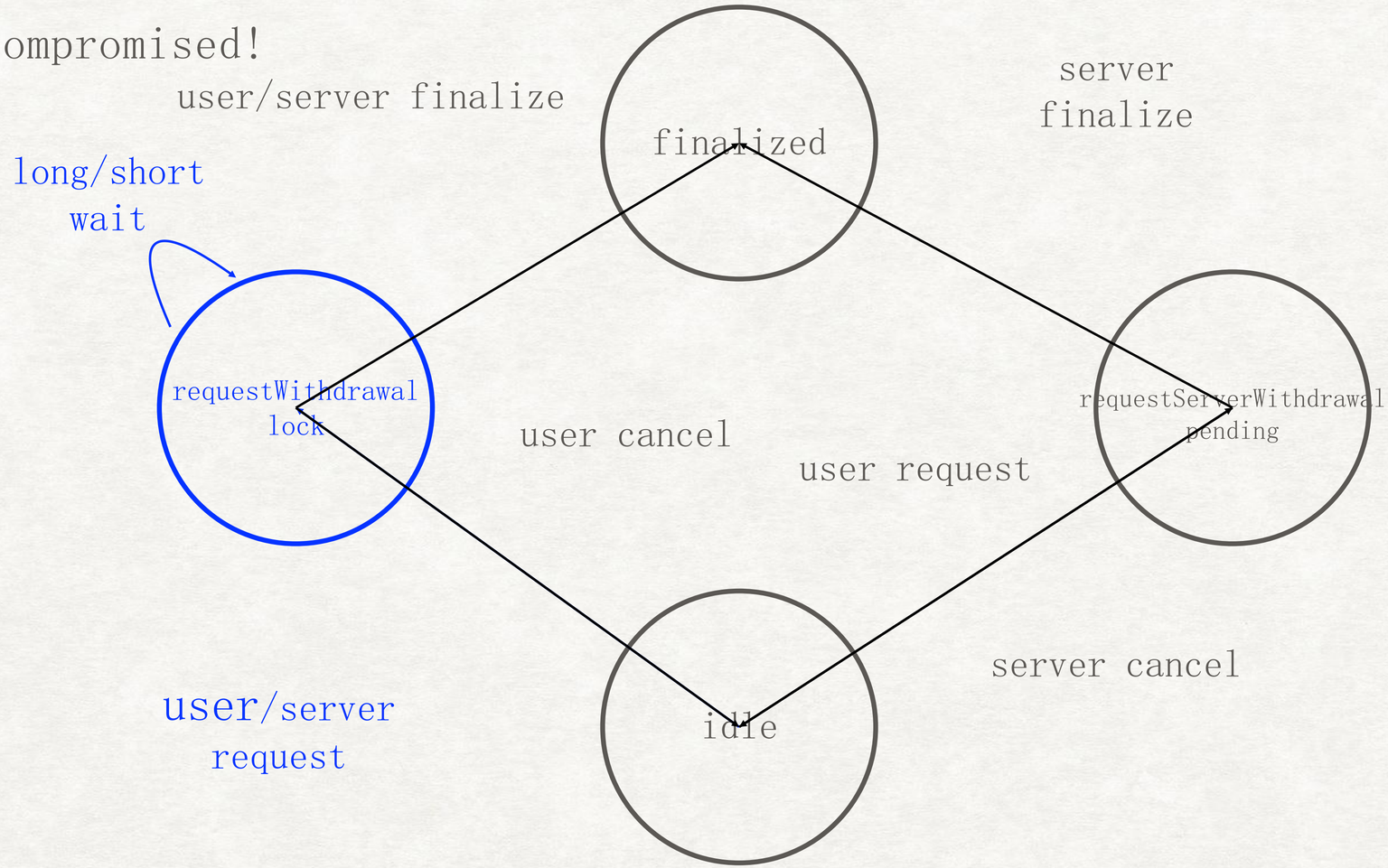
VAULT
金庫

- 3. Server compromised!
  第三方被盗



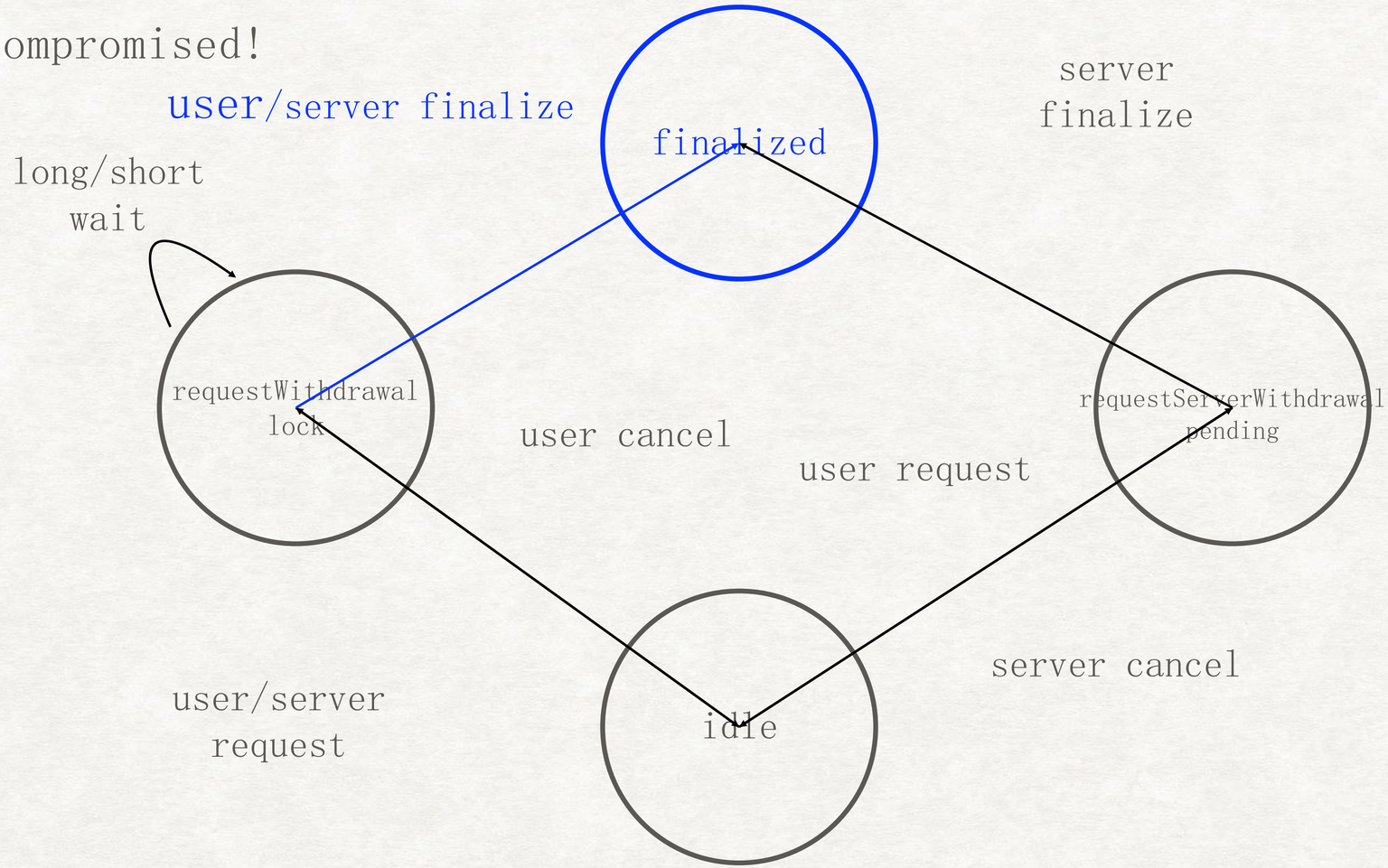
VAULT
金庫

- 3. Server compromised!
  第三方被盗



VAULT
金庫

- 3. Server compromised!
  第三方被盗



VAULT
金庫

- What happened if two of the three conditions matches?
  如果以上三種情況中的兩種同時發生呢?
  - BBBBOOOOOOOMMMM!!!!

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③ Vault
金庫

④Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤ Lesson learned
從中獲得的經驗

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- What happens if code gets too complex?
  代碼太複雜會發生什麼事?

  - BBBBUUUUUUGGG!!!!

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - July
  - Solidity
    - language to program your smart contract
      用來撰寫智能合約代碼的程式語言

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Solidity
    - Default visibility of function
      函式預設的使用範圍
    - Delegatecall
      特殊的合約呼叫函式
    - Fallback function
      Fallback函式

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet Parity多簽錢包被駭
  - Solidity
    - Visibility: Public or private
      函式使用範圍：公開或不公開
      - Default: Public
        預設範圍：公開
      - Viper(https://github.com/ethereum/viper)
        - Security first
          以安全為考量來設計
        - No default, developer has to explicitly declare it’s visibility
          沒有預設範圍，開發者必須清楚指定其使用範圍

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Solidity
    - Delegatecall
      特殊的合約呼叫函式
      - execute code of the callee in the context of the caller contract
        拿被呼叫者的代碼來在呼叫者的合約裡執行
      - Benefit: shared code
        好處：共享代碼

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Solidity
    - Fallback function: the function without a name
      Fallback函式：沒有名字的函式即為fallback函式
      - triggered by a pure transfer
        單純的轉錢會觸發此函式
      - triggered by an unidentified function call
        呼叫不存在的函式也會觸發此函式

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Let’s look into some part of the contract
    讓我們檢視一下合約裡的代碼

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - initWallet()
    - Initialize owners of the wallet
      初始化錢包的擁有人
    - SHOULD only be called once during wallet initialization
      理應只能在錢包合約部署時執行且只能執行一次
  - Delegatecall in fallback function
    在fallback函式裡使用delegatecall
    - to provide upgradability
      目的是為了在未來可以更新合約執行的邏輯

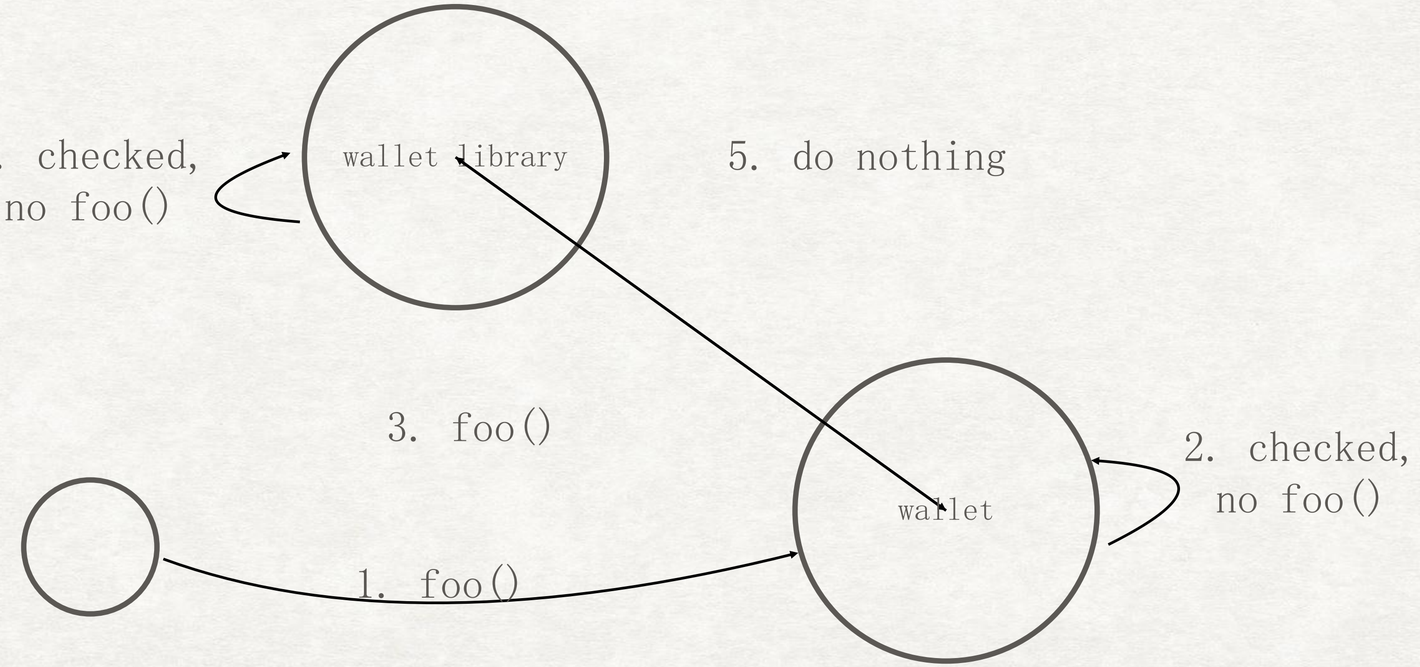
ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - So what went wrong?
    所以哪邊出錯了？

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - delegatecall in fallback function
    在fallback函式裡使用delegatecall

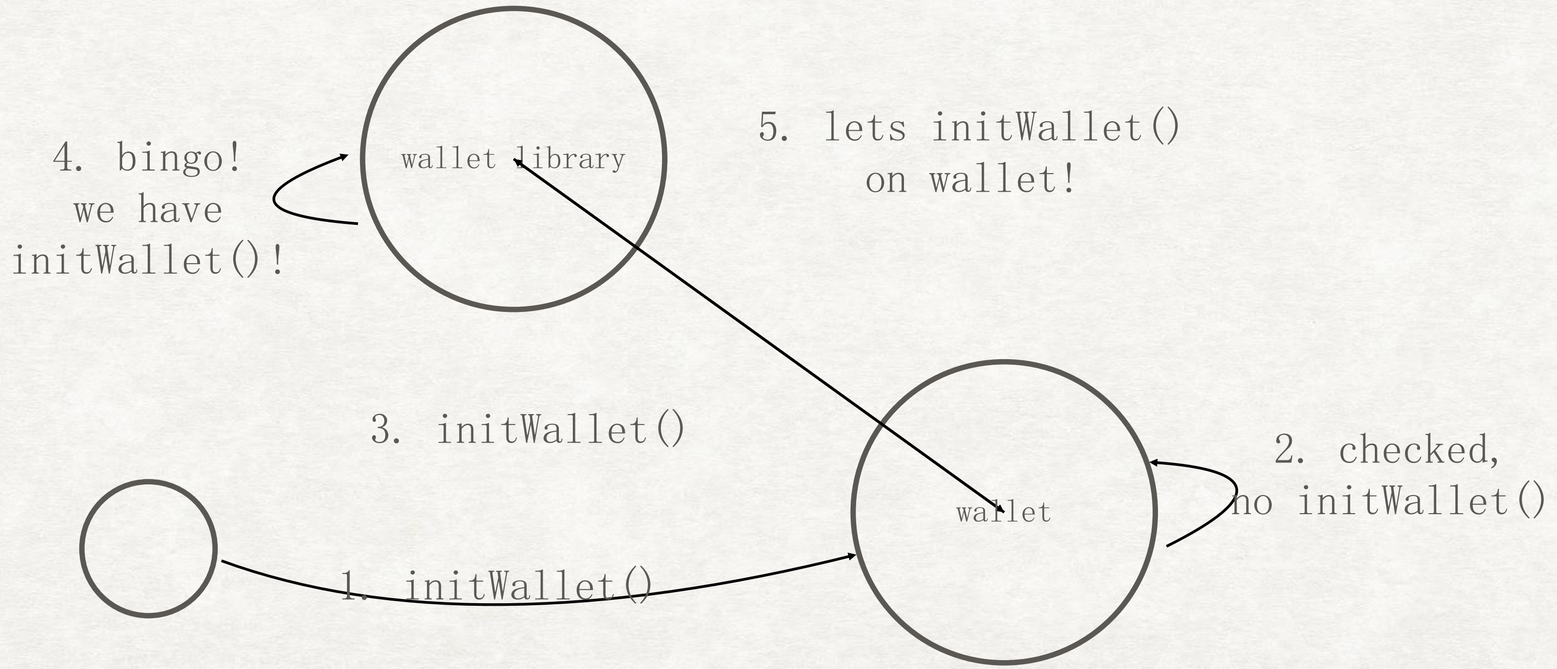
Normal:
正常情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - delegatecall in fallback function
    在fallback函式裡使用delegatecall

Hacked:
錯誤情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Patch
    修補措施
    - Add a check to prevent re-initialization of wallet
      新增條件來避免初始化函式被二次執行

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Safe!……
    - for the next four month
      至少接下來四個月都是安全的

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - November
  - Solidity
    - Library
      函式庫

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Solidity
    - Library
      函式庫
      - Contract libFoo{…}
      - Library libFoo {…}
        - Pure code, no storage
          純代碼，不儲存任何資料

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭

  - WalletLibrary in Parity’s multi-sig wallet is a contract
    Parity多簽錢包的函式庫是一個合約

  - and it’s address is hard-coded in multi-sig wallet contract
    此合約的地址寫死在多簽錢包合約的代碼裡

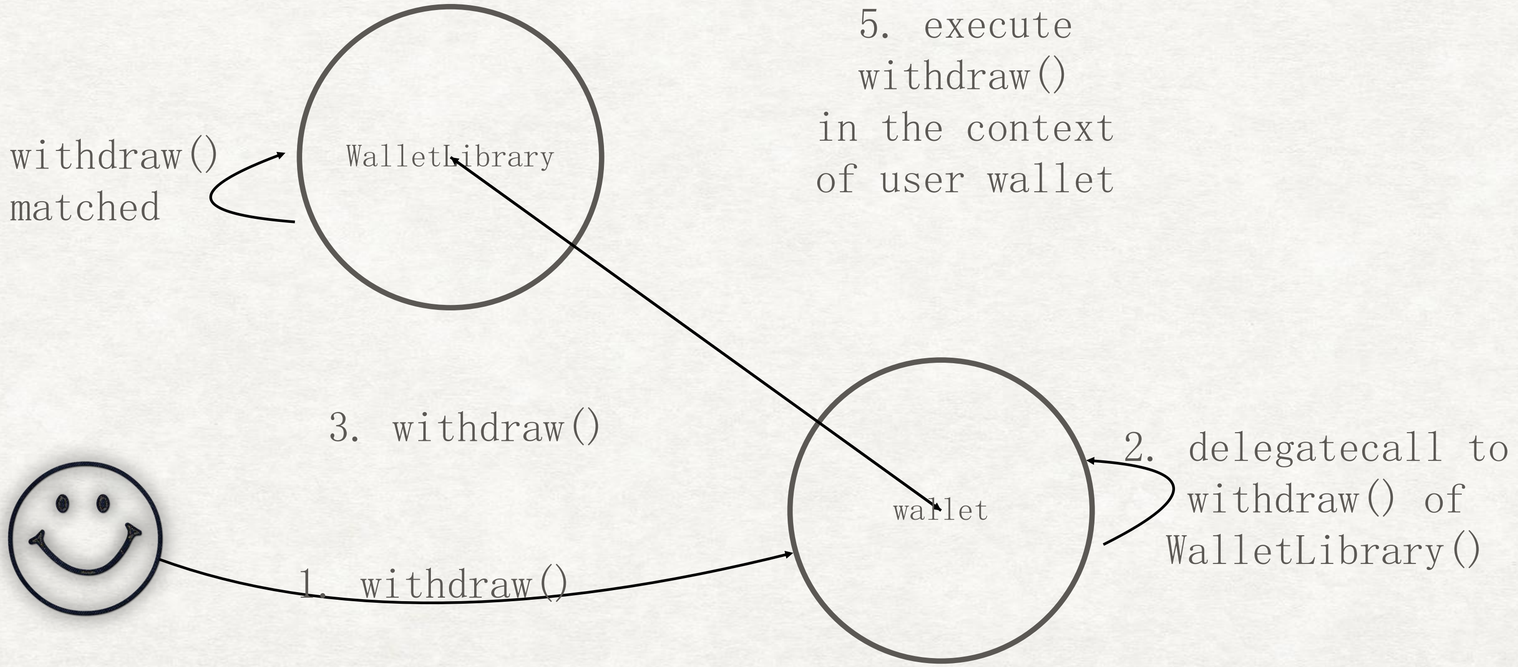
ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - So what went wrong?
    所以哪邊出錯了？

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - contract WalletLibrary

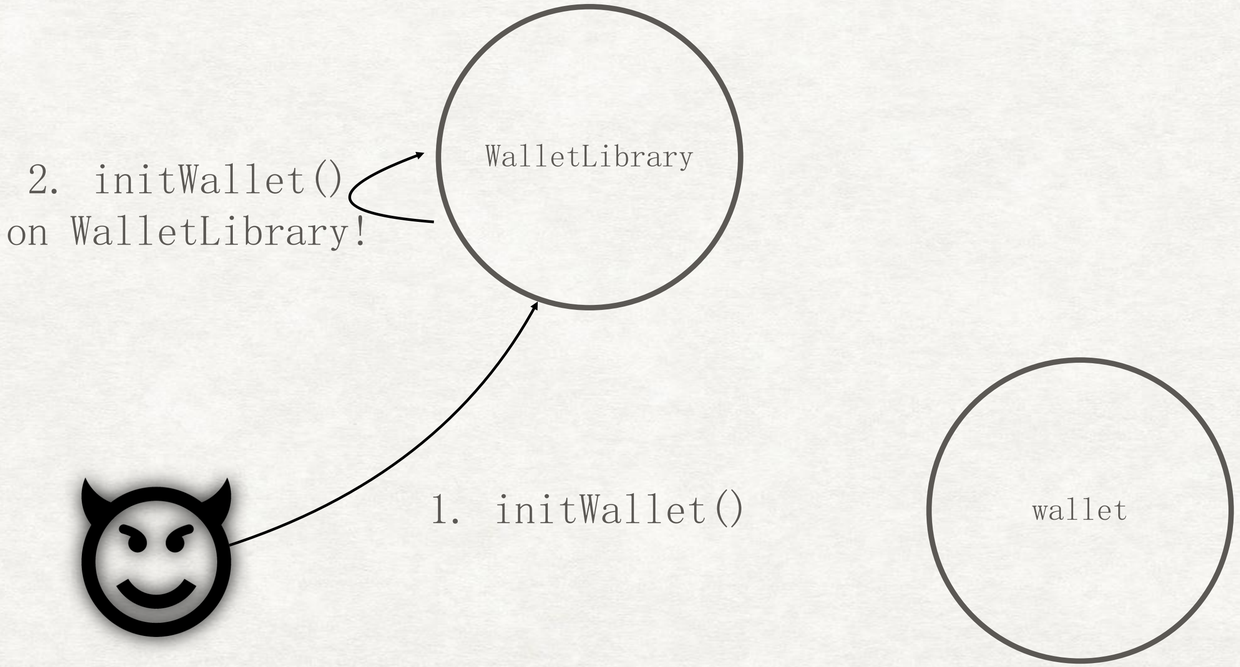
Normal:
正常情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - contract WalletLibrary

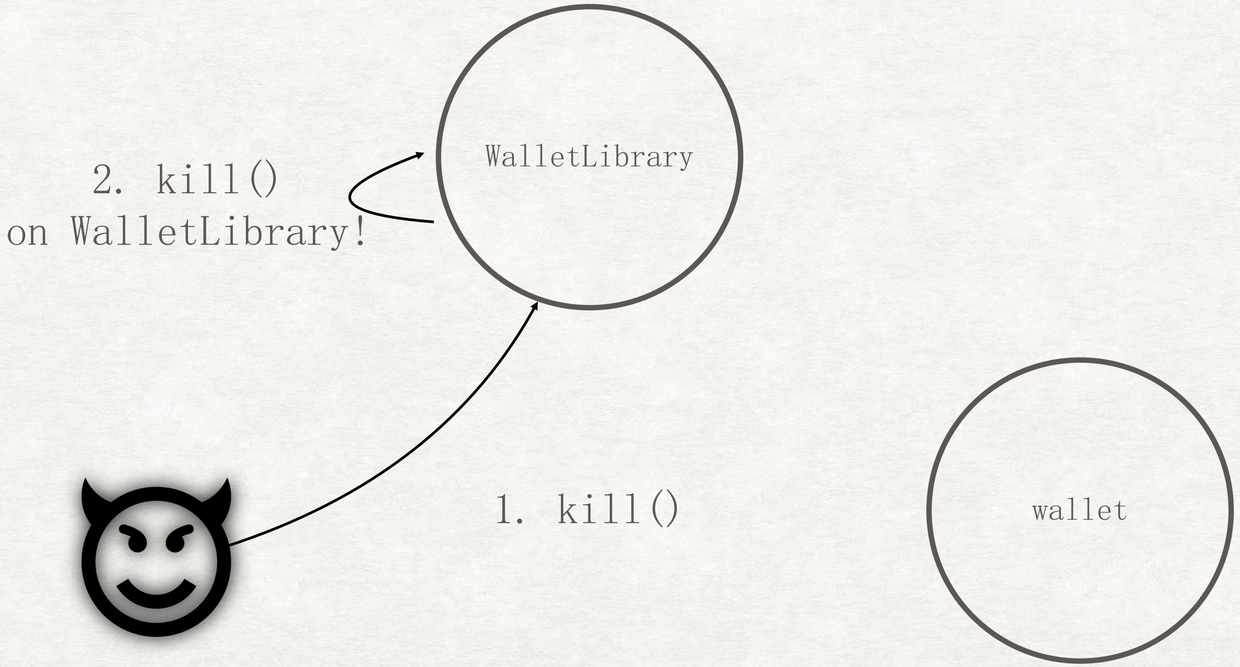
Hacked:
錯誤情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - contract WalletLibrary

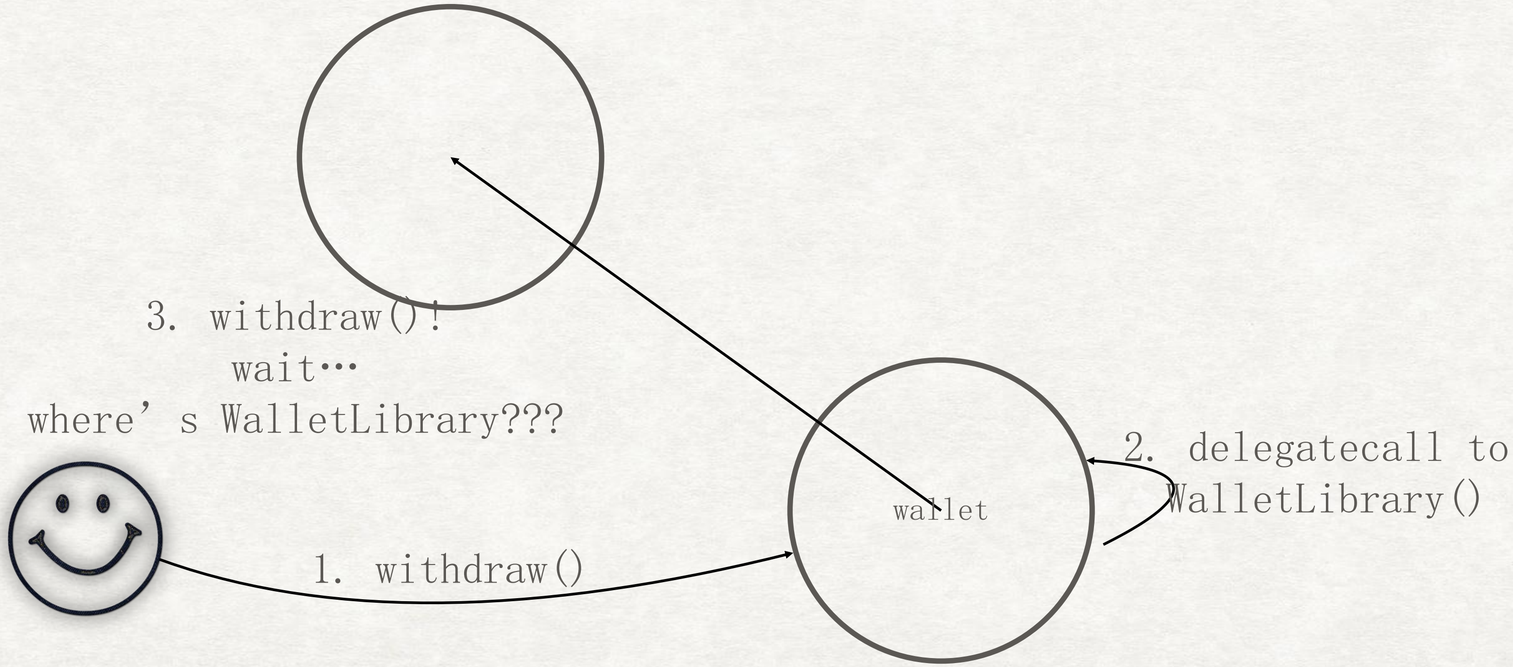
Hacked:
錯誤情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - contract WalletLibrary

Hacked:
錯誤情況:



ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - around 510000 ether locked
    約有51萬枚以太幣卡在錢包裡

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- Hack on Parity multi-sig wallet
  Parity多簽錢包被駭
  - Why not make WalletLibrary a Library?
    為什麼不將WalletLibrary設為函式庫？
    - Upgradability
      可更新合約
      - But it’s still not upgradable! Address of WalletLibrary is stored as a variable but no way to change it
        但它也沒有因此變得可更新！WalletLibrary的地址雖然是由變數儲存但無法更改

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- What can we do to prevent it from happening again?
  有哪些方法怎麼預防這類問題?
  - Follow best practice
    遵守典範代碼
    - https://github.com/ConsenSys/smart-contract-best-practices
    - https://github.com/dapphub
    - https://github.com/toadkicker/solidity-patterns

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- What can we do to prevent it from happening again?
  有哪些方法怎麼預防這類問題?
  - Audit important contract
    審計重要的合約
    - https://openzeppelin.org
    - https://media.consensys.net/preparing-for-a-smart-contract-code-audit-83691200cb9c

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- What can we do to prevent it from happening again?
  有哪些方法怎麼預防這類問題?
  - Audit important contract
    審計重要的合約
    - Make sure it’s audited by someone el
      確保是由他人來審計代碼!

ATTACKS ON MULTI-SIG WALLET
針對多簽錢包漏洞的攻擊

- What can we do to prevent it from happening again?
  有哪些方法怎麼預防這類問題?
  - Design safer languages
    設計更安全的合約設計語言
    - Viper
  - Design safer pattern
    設計更安全的模式
    - Simple -> safe
      簡單即是安全

① How do you manage your cryptocurrency?
有哪些管道來管理自己的虛擬貨幣？

② Multi-sig
多簽錢包

③ Vault
金庫

④ Attacks on multi-sig wallet
針對多簽錢包漏洞的攻擊

⑤Lesson learned
從中獲得的經驗

LESSON LEARNED
從中獲得的經驗

Simple -> safe!

LESSON LEARNED
從中獲得的經驗

- Simple -> safe
  簡單即是安全
  - How simple?
    多簡單?
    - No other mutable states
      沒有其他的可變數
      - Owner
        擁有人
      - Daily limit
        每日數量限制
      - etc.

LESSON LEARNED
從中獲得的經驗

- Simple -> safe
  簡單即是安全
  - How simple?
    多簡單?
    - One function: withdraw(amount, sendTo, sigs)
      只有一個函式
      - sigs: signature from owners
        sigs包含所有擁有人的簽章
      - check the signatures all together
        一次檢查所有的簽章
      - wrong signature or not enough signatures -> FAIL
        簽章錯誤、簽章不足，則交易失敗

LESSON LEARNED
從中獲得的經驗

- Simple -> safe
  簡單即是安全
  - How simple?
    多簡單?
    - Only two states of a withdrawal
      領錢只會有兩種狀態
      - Success
        成功
      - Not success
        不成功

LESSON LEARNED
從中獲得的經驗

- Simple -> safe
  簡單即是安全
  - Trade off between Safety and Convenience
    安全的代價是沒那麼方便

- http://vitalik.ca/files/vault_proposal_nqdqwt21.txt
- https://paritytech.io/blog/security-alert.html
- http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/
- https://blog.gridplus.io/toward-an-ethereum-multisig-standard-c566c7b7a3f6
- https://medium.com/@ChrisLundkvist/exploring-simpler-ethereum-multisig-contracts-b71020c19037

THANK YOU