Resource-Oriented Programming

in Libra Move and Flow Cadence

Hsuan Lee 李玄

Co-founder & CEO, portto

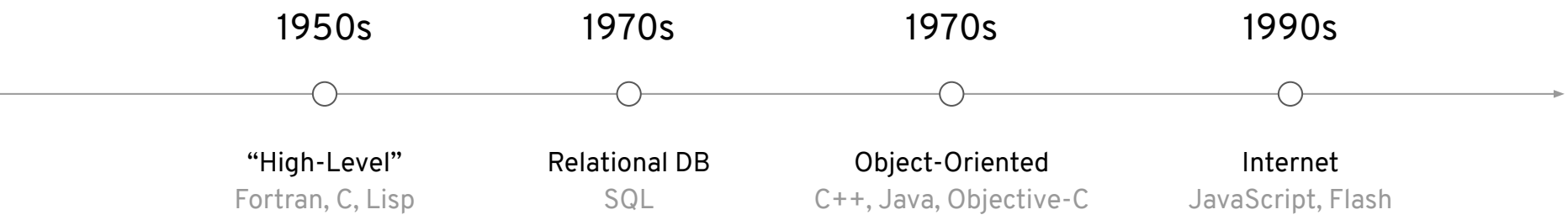
Who am I?

1. Co-founder & CEO @ portto
2. VP of Engineering @ Cobinhood & DEXON
3. Software Engineer @ 17 Media, Agoda, Yahoo

Outline of this Speech

1. The history of programming languages
2. Pain-points in smart contracts
3. Resource-oriented programming
4. Some examples

History of programming languages



Programming on blockchain

- Introduced by Ethereum
- General purpose programming
- Suitable for
  - Transfer scarce assets
  - Control access
  - Provide auditable execution
  - Provide traceable proof??

What’s wrong with current model?

- Centralized ledger
- Reduce chance of parallelism
- Data structure does not reflect ownership
- Huge attack surface
- Difficult to audit & analyze

Common attacks

1. Reentrance
    a. DAO hack:
    https://quantstamp.com/blog/what-is-a-re-entrancy-attack

    b. ERC777 + Uniswap / Lendf.me:
    https://www.abmedia.io/detailed-explanation-of-uniswaps-erc777-reentry-risk/

2. Abuse authorization
    a. Parity wallets got locked:
    https://github.com/openethereum/openethereum/issues/6995

    b. Centralized ERC20:
    https://etherscan.io/address/0xc12d1c73ee7dc3615ba4e37e4abfdbddfa38907e

Access Control
Who you are (list)

Scarce Assets
Data structure

Security
Contract level

Existing

v.s.

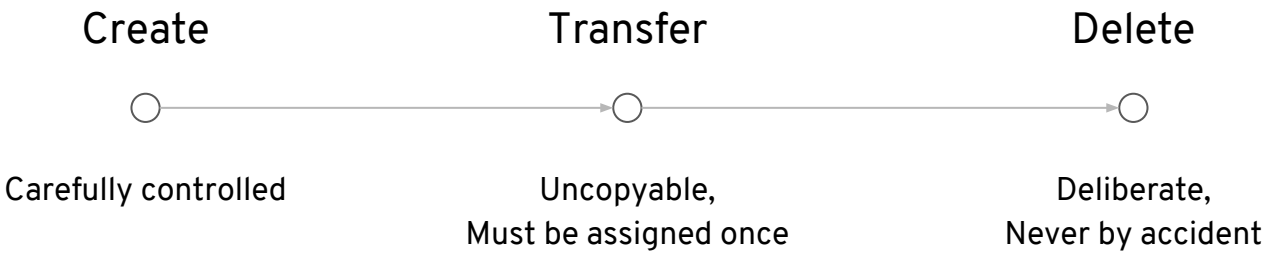
Access Control
What you have

Scarce Assets
Resources

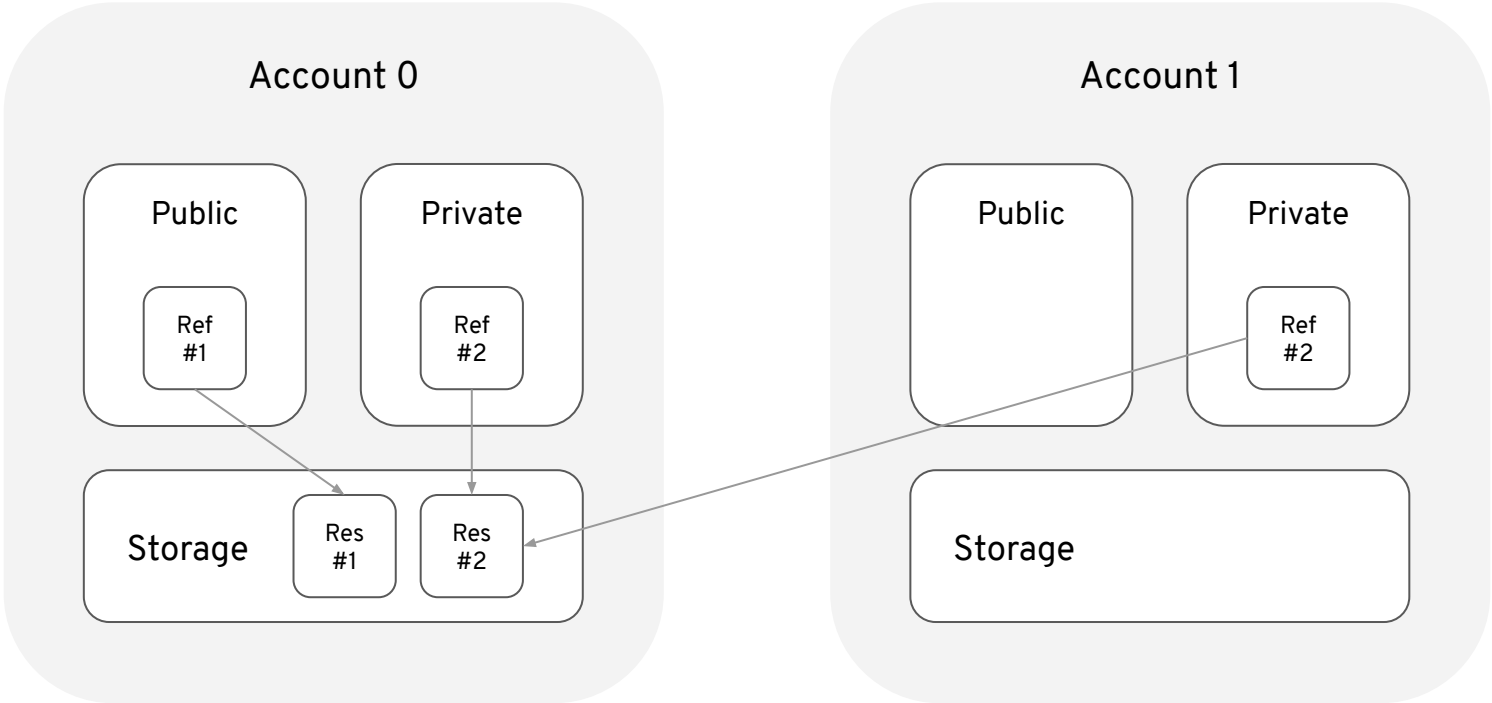
Security
VM level

ROP

Resource lifecycle



ROP accounts

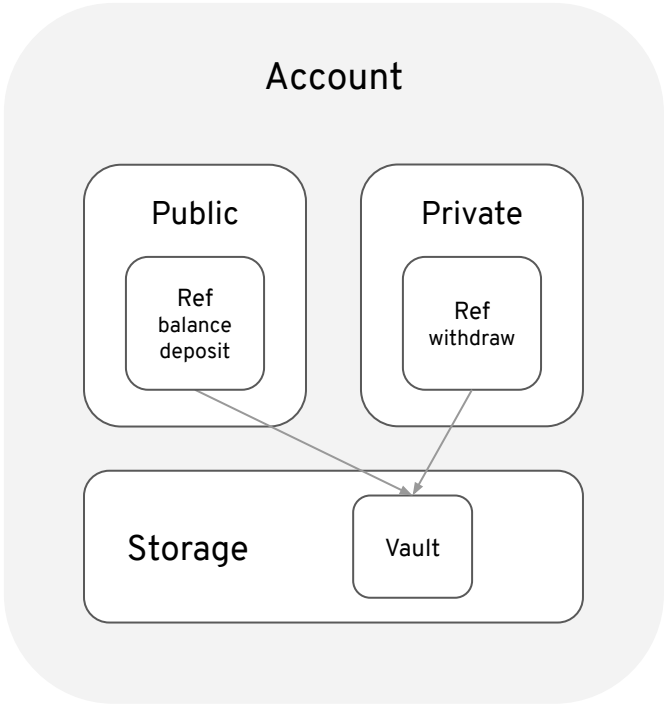


Ethereum fungible token

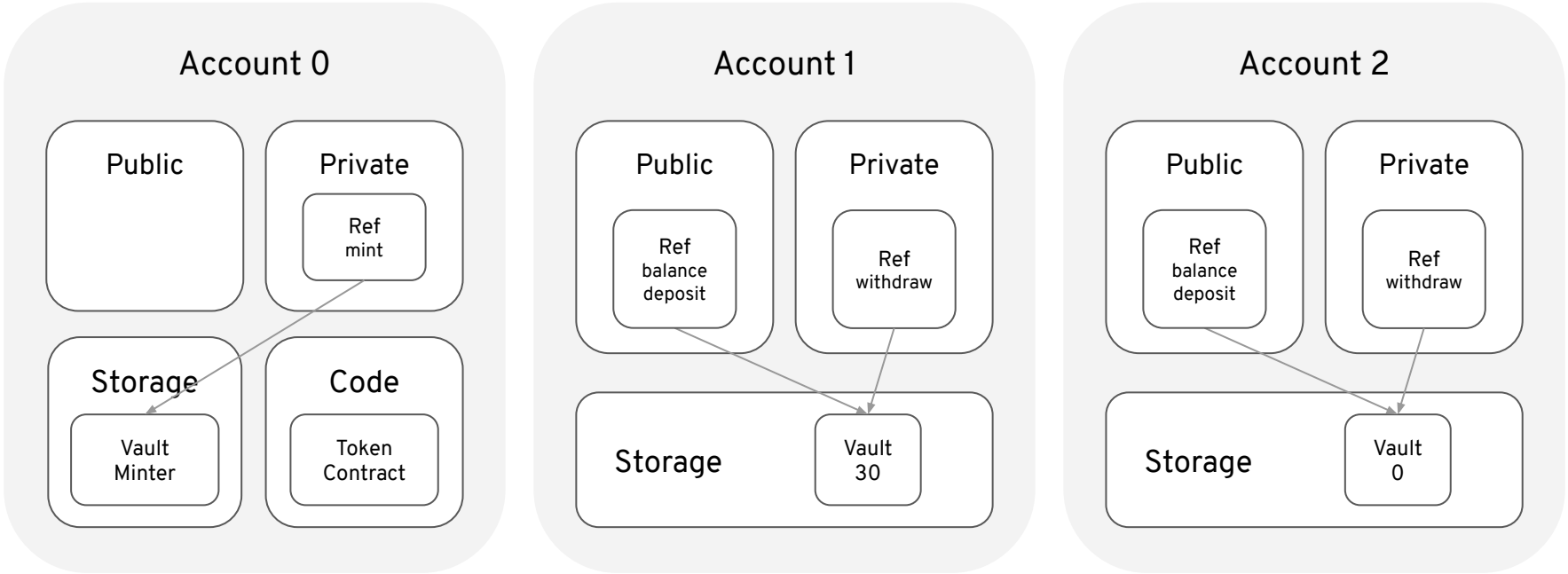
```
contract ERC20 {
  mapping (address => uint256) private _balances;

  function _transfer(address sender, address recipient, uint256 amount) {
    // ensure the sender has a valid balance
    require(_balances[sender] >= amount);

    // subtract the amount from the senders ledger balance
    _balances[sender] = _balances[sender] - amount;

    // add the amount to the recipient's ledger balance
    _balances[recipient] = _balances[recipient] + amount
  }
}
```

ROP fungible token

```
pub resource Vault: Provider, Receiver {
  pub var balance: UFix64

  init(balance: UFix64) {
    self.balance = balance
  }

  pub fun withdraw(amount: UFix64): @Vault {
    self.balance = self.balance - amount
    return <-create Vault(balance: amount)
  }

  pub fun deposit(from: @Vault) {
    self.balance = self.balance + from.balance
    destroy from
  }
}
```



ROP fungible token



ROP fungible token demo

https://play.onflow.org/26b79fc4-bde4-4783-85ca-b5bdbfdbc543

ROP Advantages

- Built-in security
- Less human error
- Better parallelism
- State rent made possible
- Resource hierarchy

Resource-oriented resources

- Getting Started With Move
- Cadence Language Reference
- Cadence Fungible Tokens

Download Slides

Good Stuff

Flow 台灣開發者社群

Blocto 開發者 Discord

幣安台灣徵才中

Questions?