Age-USD Stablecoin

Overview of the Ergo smart-contract

ERGO

Alex Chepurnoy
Amitabh Saxena

INPUT | OUTPUT

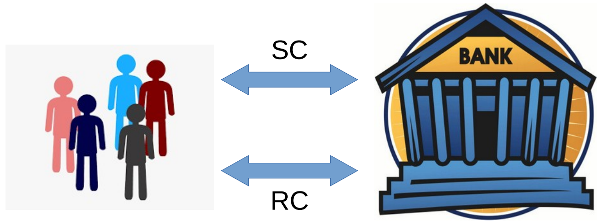
Dr. Bruno Woltzenlogel Paleo
Colin Edwards
Dr. Dmytro Kaidalov
Dr. Jann Müller

EMURGO

Nicolas Arqueros
Robert Kornacki

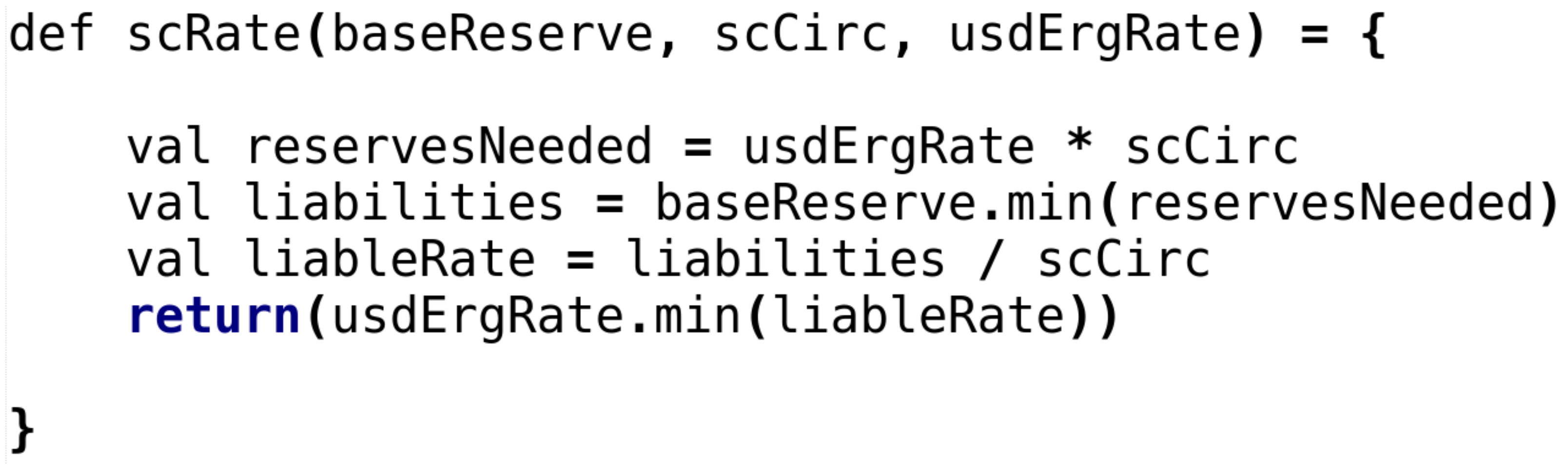
Stablecoin model

- Emulate a centralized bank that holds three assets:
  - Stable-coin (SC) tokens (issued by bank)
  - Reserve-coin (RC) tokens (issued by bank)
  - Base currency (BC) tokens (ERG, BTC, ETH, etc)
- Bank exchanges BC <---> SC and BC <---> RC
  - SC-rate: how many BC for 1 SC (units: ERG/SC)
  - RC-rate: how many BC for 1 RC (units: ERG/RC)
  - Peg-rate: how many BC for 1 USD (units: ERG/USD)
- SC-rate should be close to Peg-rate (i.e. SC ~ USD).
  - RC used to handle fluctuations
  - SC-rate determined using algorithm that takes as input:
    - Current BC reserves (B)
    - SC in circulation (S)
    - Actual rate of ERG/USD at the time of exchange (E)
  - RC-rate determined using algorithm that takes as input:
    - Current BC reserves (B)
    - SC in circulation (S)
    - RC in circulation (R)
    - Actual rate of ERG/USD at the time of exchange (E)



- SC-rate = F(B, S, E)
- RC-rate = G(B, S, R, E)

Computing SC-rate



```
def scRate(baseReserve, scCirc, usdErgRate) = {

    val reservesNeeded = usdErgRate * scCirc
    val liabilities = baseReserve.min(reservesNeeded)
    val liableRate = liabilities / scCirc
    return(usdErgRate.min(liableRate))

}
```

Computing RC-rate

```
def rcRate(baseReserve, scCirc, rcCirc, usdErgRate) = {

    val reservesNeeded = usdErgRate * scCirc
    val liabilities = baseReserve.min(reservesNeeded)
    val equity = baseReserve - liabilities
    val rcRate = if (equity == 0 || rcCirc == 0) rcDefaultRate
                 else equity / rcCirc
    return(rcRate)

}
```

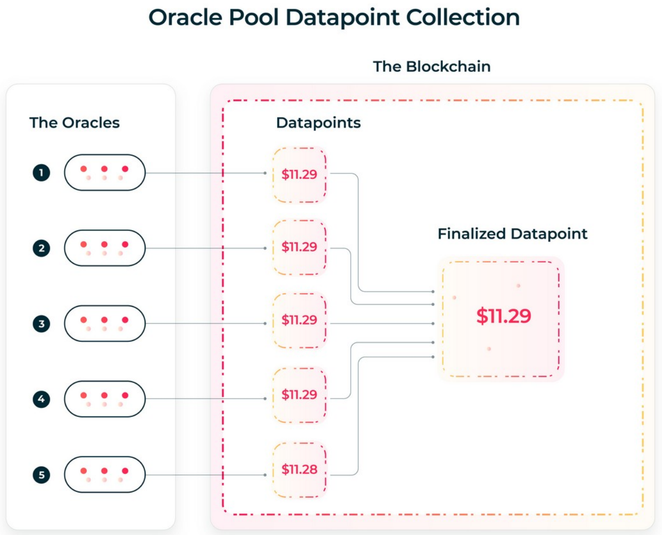
External Rate Source

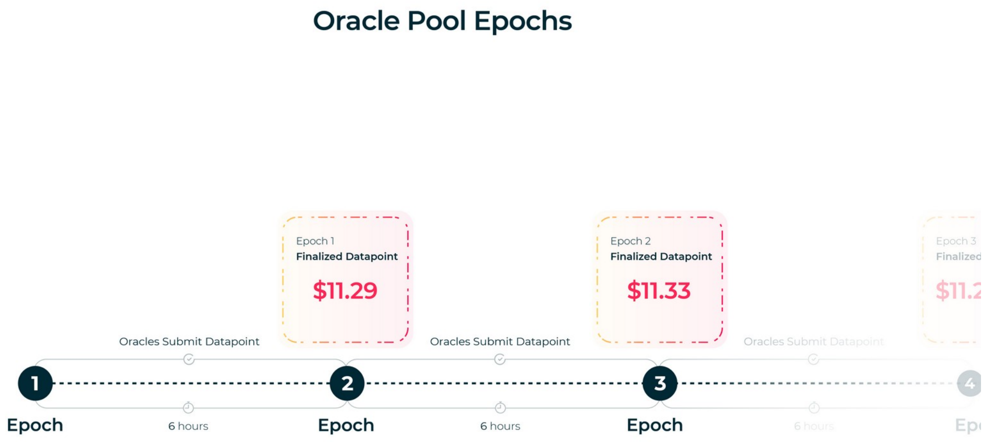
- Need ergUsdRate from reliable external source
- Ergo has concept of **oracle-pools**
  - Number of parties submit their own rates (oracle)
  - Average of all rates taken as ergUsdRate (pool)
  - Each rate must be within a max deviation
  - Some other checks (rate posted must be fresh, etc)

Oracle-pools

(collaboration with Emurgo Research)

https://tinyurl.com/oracle-pools



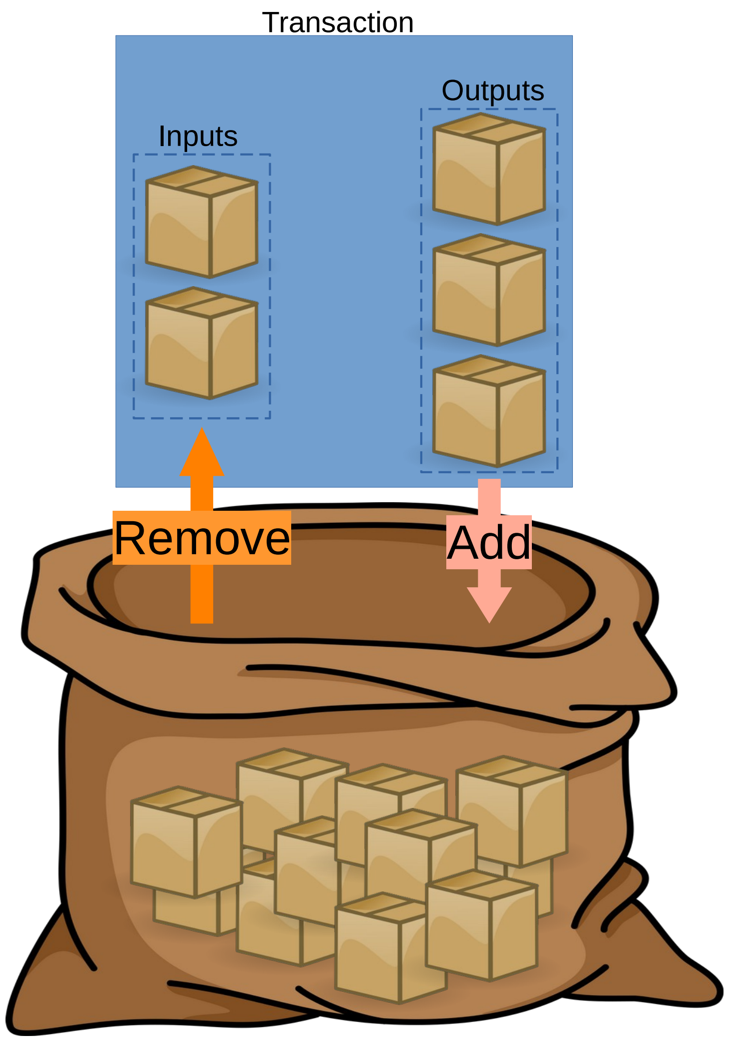


Ergo Platform

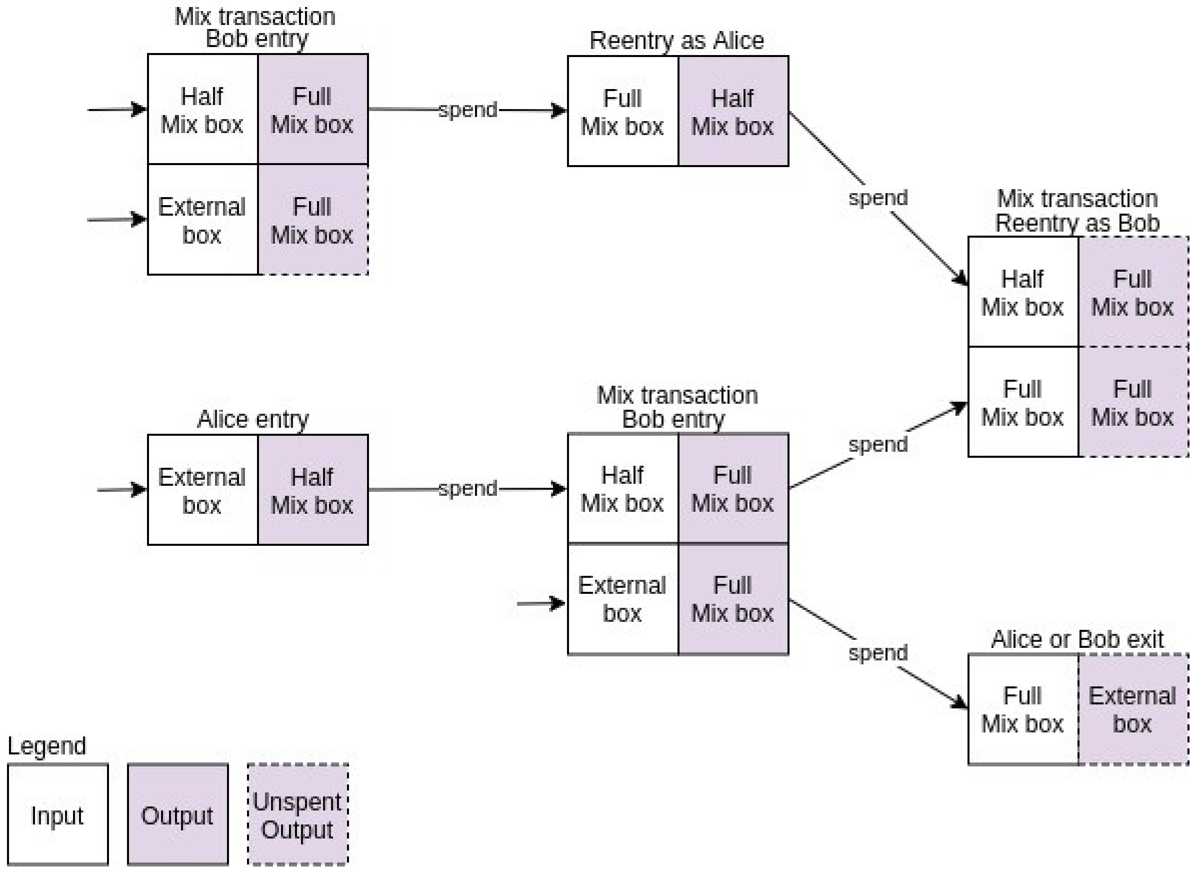
- UTXO-based blockchain
- Advanced DeFi features
  - ErgoScript (Scala-like language) for guard scripts (smart contracts)
  - Functional programming
  - Secondary Assets (NFTs, tokens)
- Scalability feaures
  - Storage rent
  - Light clients with full-node security
  - NiPoPoW

UTXOs

- Nodes keep a UTXO-set
  - A bag of boxes (box = UTXO)
  - Each box has unique id
- Transaction changes the bag
  - Removes one or more boxes
  - Adds one or more new boxes
  - Script in removed boxes evaluated
    - Must return true for tx to be valid
- Boxes in bag cannot be changed
  - Can only be removed
  - Change emulated by creating new box
    - With different id; other aspects preserved



UTXOs


Mix transaction Bob entry
Half Mix box
Full Mix box
External box
Full Mix box
spend
Reentry as Alice
Full Mix box
Half Mix box
spend
Mix transaction Reentry as Bob
Half Mix box
Full Mix box
Full Mix box
Full Mix box
Alice entry
External box
Half Mix box
spend
Mix transaction Bob entry
Half Mix box
Full Mix box
External box
Full Mix box
spend
spend
Alice or Bob exit
Full Mix box
External box
Legend
Input
Output
Unspent Output

Ergo UTXO (Box)

Four mandatory registers

- R0: Monetary value (Nano-Ergs)
- R1: Guard script
- R2: Creation info
- R3: Assets (secondary tokens)

Six optional registers (R4...R9)

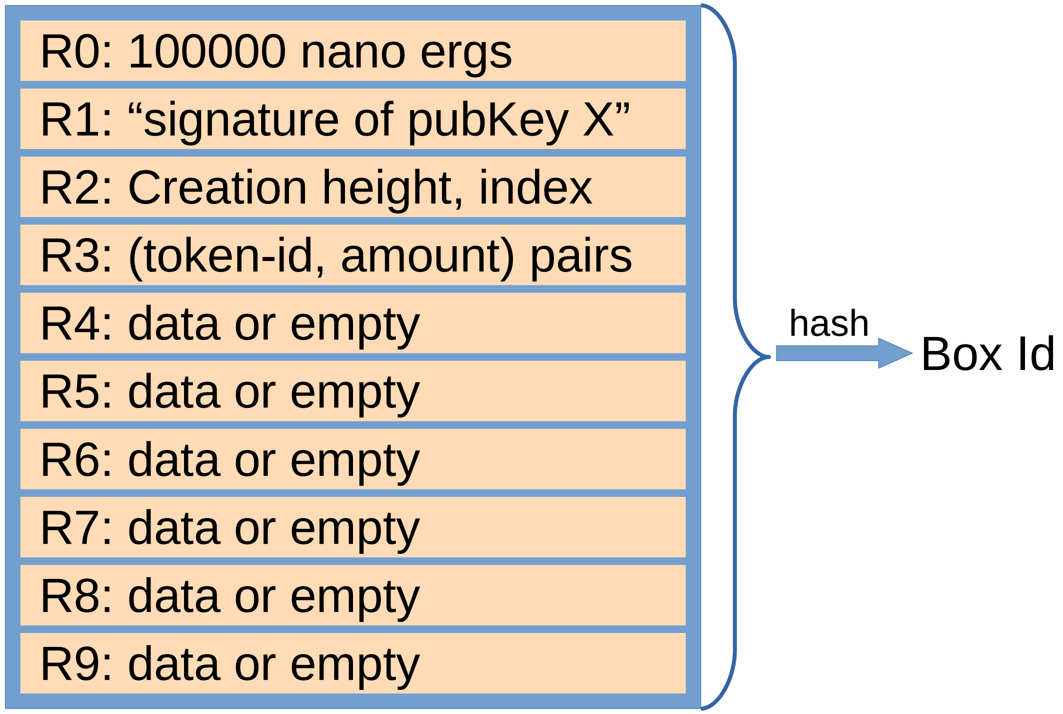
- Can contain aribitrary data

Assets

- Token-id: 32 bytes long
- Amount: Long value

Transaction can issue at most 1 asset

- Token id of issued asset = first input box Id
- NFT = Asset issued in quantity 1

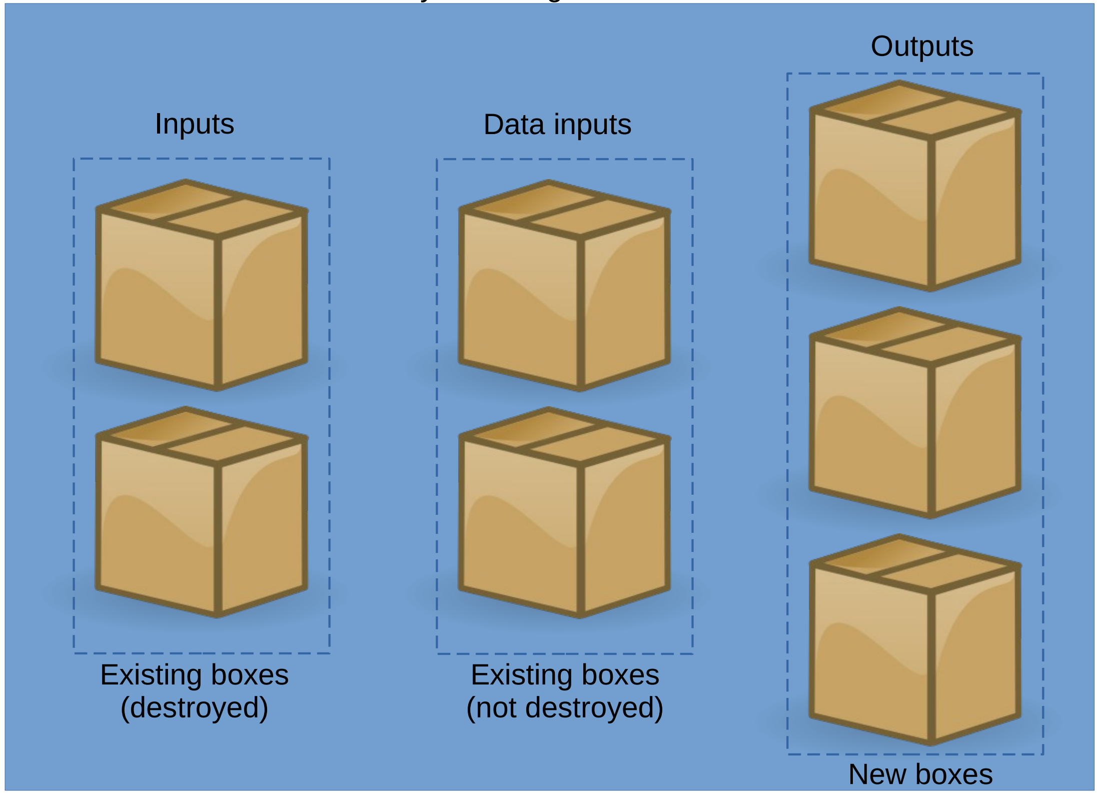


Ergo Transaction

Three sets of boxes

- Inputs (boxes destroyed)
  - Script executed
- Data inputs (read-only boxes)
  - Script NOT executed
- Outputs (new boxes created)

Anatomy of an Ergo transaction



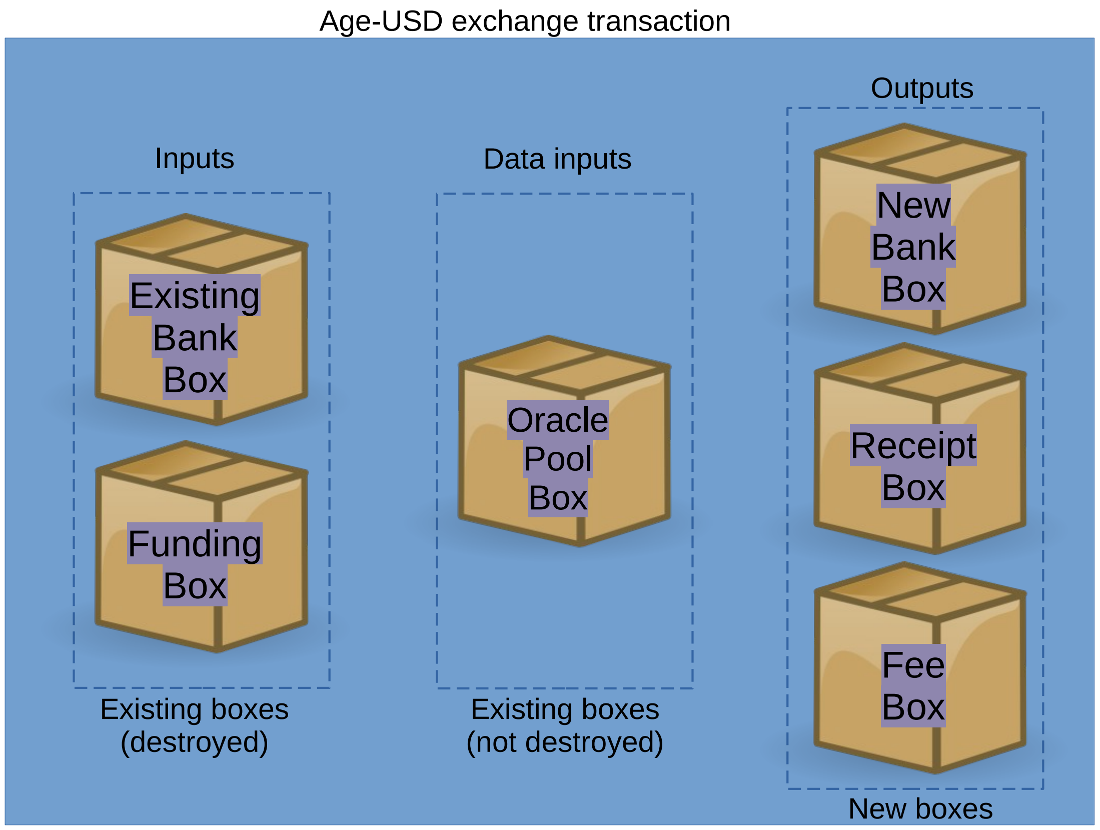
Ergo Stablecoin Transaction

Types of boxes

- Bank box holds reserves and SC/RC tokens
  - Only one box exists at any time
- Funding box contains Ergs/SC/RC tokens for exchange
- Receipt box contains Erg/SC/RC tokens from exchange
- Fee box for paying transaction fee
- Oracle-pool box contains **ergUsdRate** in R4
  - Only one box exists at any time

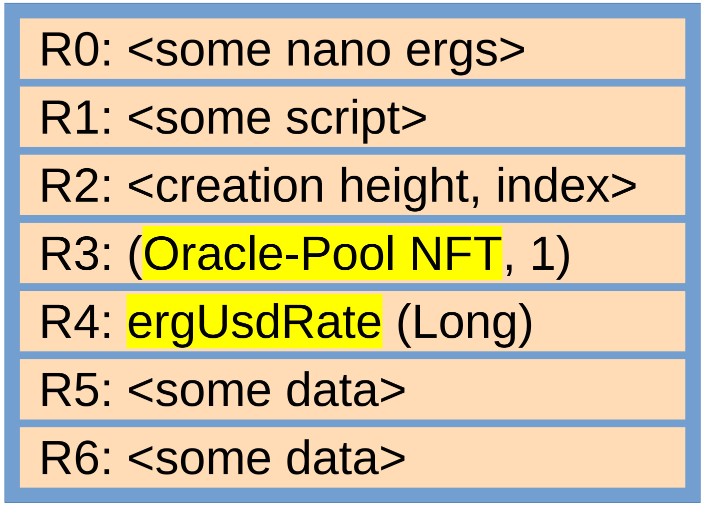
Types of exchange transactions

- Purchase SC
  - Funding box has Ergs to give to bank
  - Receipt box has SC tokens from bank
- Redeem SC
  - Funding box has SC tokens to give to bank
  - Receipt box has Ergs taken from bank
- Purchase RC
  - Funding box has Ergs to give to bank
  - Receipt box has RC tokens taken from bank
- Redeem RC
  - Funding box has RC tokens to give to bank
  - Receipt box has Ergs taken from bank



Oracle-Pool Box

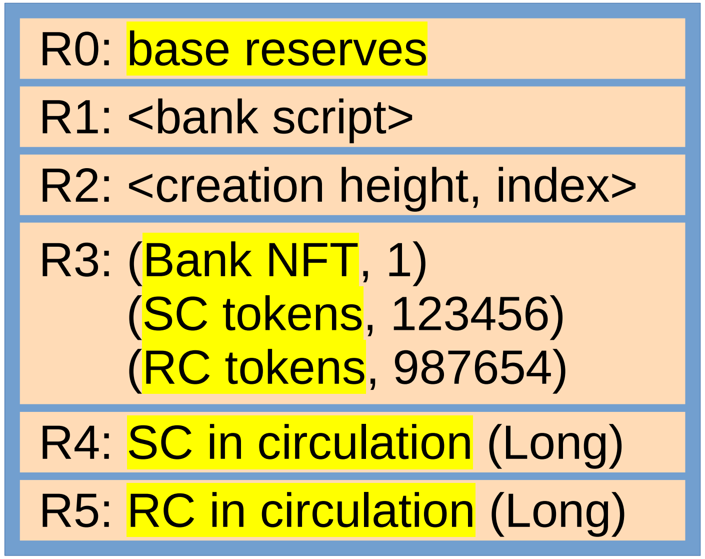
- Used as data input
  - Script will NOT be executed
- Tokens
  - NFT to uniquely identify box
- Optional registers:
  - R4: Average ergUsdRate
  - R5: Some data (not relavent)
  - R6: Some data (not relavent)



R0: <some nano ergs>
R1: <some script>
R2: <creation height, index>
R3: (Oracle-Pool NFT, 1)
R4: ergUsdRate (Long)
R5: <some data>
R6: <some data>

Bank Box

- Used as input
  - Script WILL BE executed
- Tokens
  - NFT to uniquely identify box
  - SC tokens for exchange
  - RC tokens for exchange
- Registers
  - R4: SC in circulation
  - R5: RC in circulation
- Monetary value used as base-reserve



R0: base reserves

R1: <bank script>

R2: <creation height, index>

R3: (Bank NFT, 1)
(SC tokens, 123456)
(RC tokens, 987654)

R4: SC in circulation (Long)

R5: RC in circulation (Long)

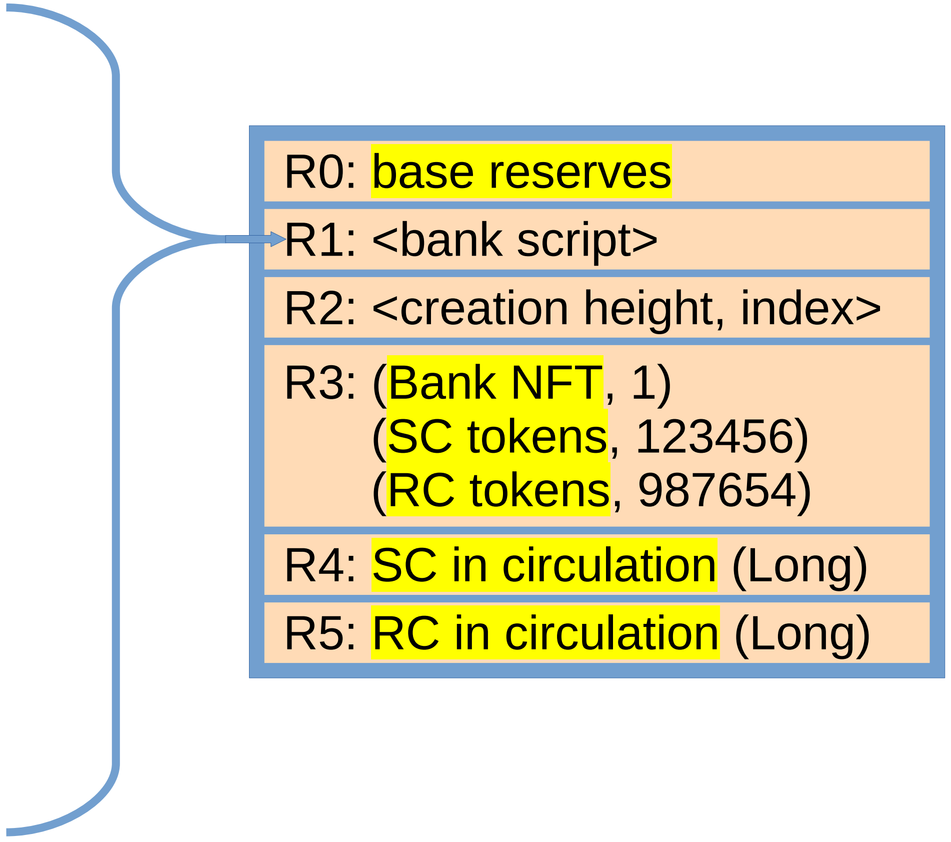
Receipt Box

- Output of transaction
  - Owned by end-user (Alice)
- Tokens
  - SC/RC tokens purchased
- Registers
  - R4: Delta in SC/RC (circulation)
  - R5: Delta in bank base reserves
- Monetary value contains redeemed Ergs

R0: Redeemed nanoErgs
R1: <Alice public key>
R2: <creation height, index>
R3: (SC/RC token, 123)
R4: Delta SC/RC
R5: Delta base reserves

Bank Box Contract

- I should be the first input and my copy the first output s.t:
  - Script and bank NFT is preserved
  - Rest updated as per exchange rules
    - SC/RC tokens, R4/R5, Monetary value
- Second output must be receipt box
  - First token index contains SC/RC purchased, if any
  - R4 contains SC/RC delta (negative if redeeming)
  - R5 contains base reserve delta (negative if dedeeming)
- First data input must be oracle-pool box
  - Identified by hardwired Oracle pool NFT id
  - R4 of the oracle pool box contains ergUsdRate
- Exchange rules:
  - Only one of SC/RC can be exchanged in one tx
  - SC-rate/RC-rate determined using the formula earlier
  - Base reserve delta must be
    - SC delta * SC-rate (if SC exchange)
    - RC delta * RC-rate (if RC exchange)
  - Fee in base-currency (Erg) added to base reserve delta
- Final reserve ratio must be between 400% and 800%



R0: base reserves
R1: <bank script>
R2: <creation height, index>
R3: (Bank NFT, 1)
(SC tokens, 123456)
(RC tokens, 987654)
R4: SC in circulation (Long)
R5: RC in circulation (Long)