xUDT & RGB++ Develop

Dylan - RGB++ Developer

xUDT 资产在 CKB 上发行

xUDT RFC

Example Code

测试网交易示例

```
# CKB TX
input:
    empty_cell:
        lock: xudt_owner_lock(secp256k1, JoyID, etc.)


output:
    xudt_cell:
        amount: 21,000,000 * 10^8 (decimal=8)
        type: xudt
        lock: xudt_owner_lock

    info_cell:
        data: xudt-info (name, symbol, decimal, etc.)
        lock:
            unlockable lock
```

xUDT 资产在 CKB 上转让

Example Code

测试网交易示例

```
# CKB TX
input:
    xudt_cell:
        amount: 10,000 * 10^8 (decimal=8)
        type: xudt
        lock: xudt_owner_lock(Secp256k1, JoyID, etc.)

output:
    xudt_cell:
        amount: 1000 * 10^8 (decimal=8)
        type: xudt
        lock: reciver_lock(Secp256k1, JoyID, etc.)

    xudt_cell:
        amount: 9000 * 10^8 (decimal=8)
        type: xudt
        lock: xudt_owner_lock(Secp256k1, JoyID, etc.)
```

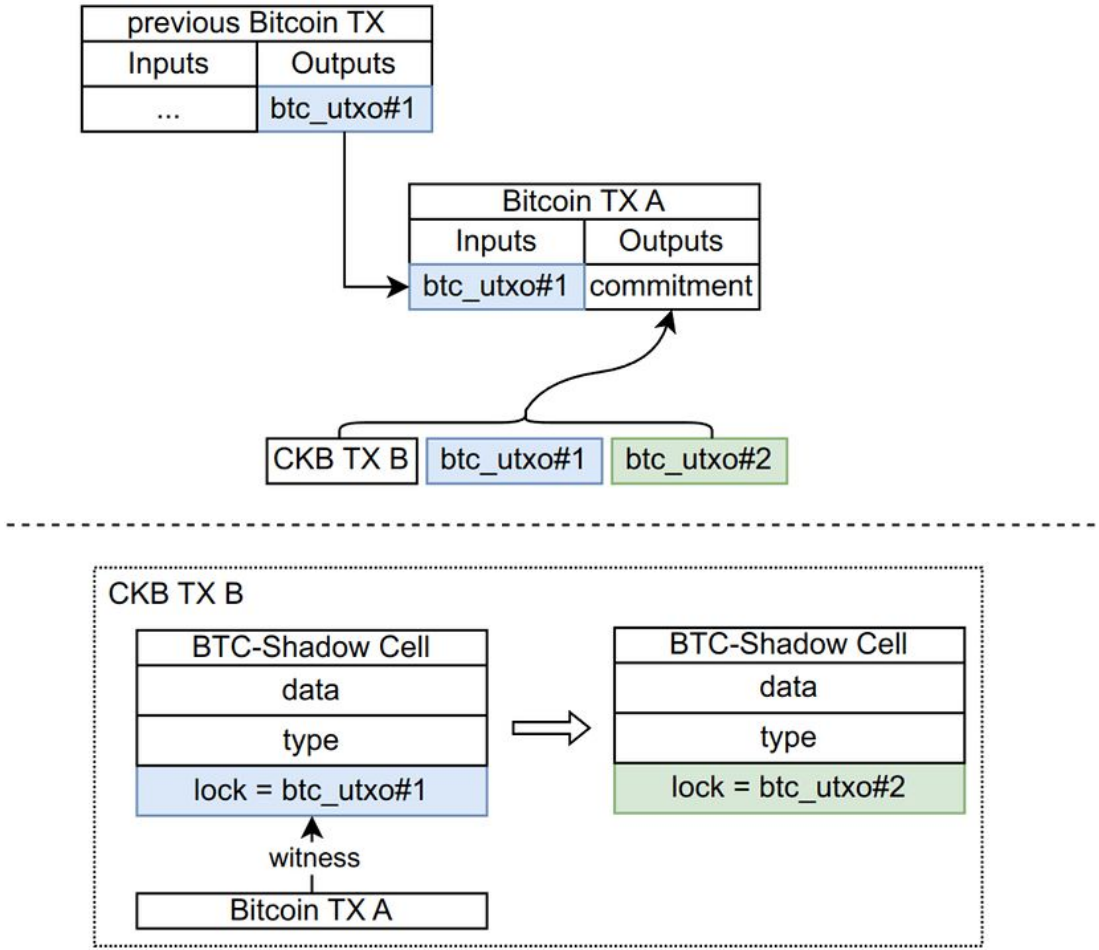
什么是 RGB++？

CKB 状态更新由 BTC 交易决定

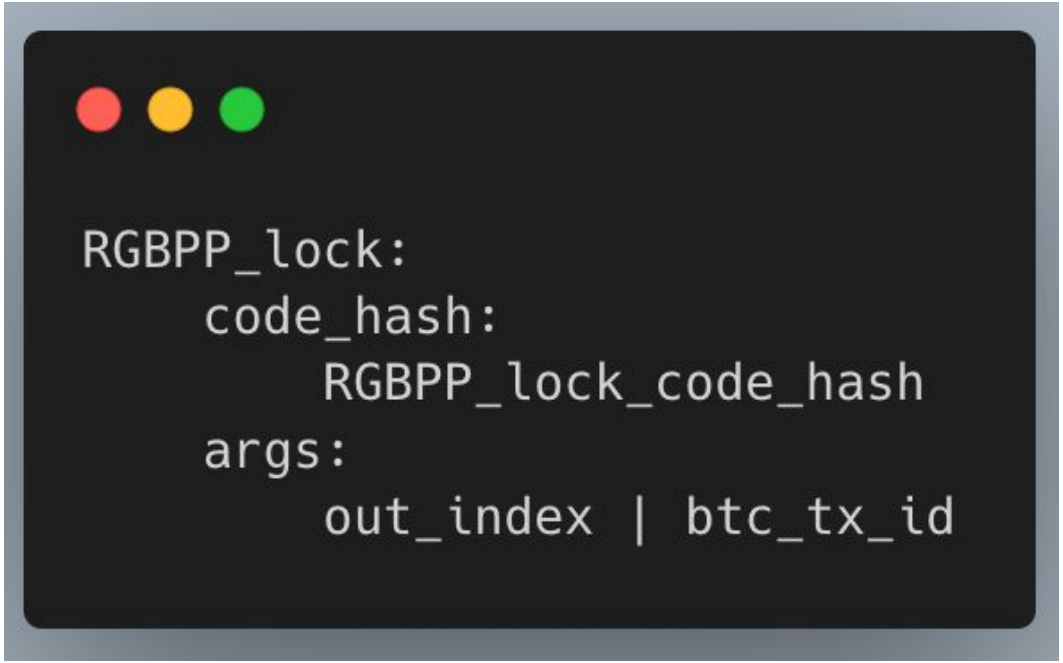
- CKB TX Lock 是 BTC UTXO

CKB 扩展 BTC 资产可编程性

- CKB TX Type 决定资产类型



RGB++ Lock Script 数据结构



RGB++ 交易的一般构造流程

1. 创建 RGB++ CKB virtual tx
2. 根据 virtual tx 生成 RGB++ commitment
3. 创建包含 commitment 的 BTC tx，并签名上链
4. 将 BTC txId 和 virtual tx 发送到 RGB++ Queue Service
5. 如果不使用 Queue Service，可以参考 launch 和 local examples

RGB++ 资产在 BTC 上发行

1. 创建一个用于资产发行的 CKB RGB++ Cell
2. 在 BTC 上发行 RGB++ FT 资产，包括发行总量、资产名称、小数位等信息（不可增发）

Example Code

BTC 交易示例

CKB 交易示例

```
# BTC TX
utxo#1 => commitment, utxo#2

# CKB TX
input:
    rgbpp_cell:
        lock:
            RGBPP_lock, utxo#1

output:
    rgbpp_cell:
        amount: 21,000,000 * 10^8 (decimal=8)
        type: xudt
        lock: RGBPP_lock, utxo#2

    rgbpp_cell:
        type: unique type
        data: xudt-info (name, symbol, decimal, etc.)
        lock:
            unlockable lock
```

RGB++ 资产在 BTC 上转让

Example Code

BTC 交易示例

CKB 交易示例

```
# BTC_TX
input:
    btc_utxo_1  # =(previous_btc_tx | out_index)
    ...
output:
    OP_RETURN: commitment
    btc_utxo_3  # =(new_bitcoin_tx | out_index)
    btc_utxo_4  # =(new_bitcoin_tx | out_index)

# CKB_TX
input:
    rgbpp_cell:
        data: 2000 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: btc_utxo_1 = (out_index | previous_btc_tx_id)

output:
    rgbpp_cell:
        data: 1500 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: out_index = 1 | new_bitcoin_tx_id

    rgbpp_cell:
        data: 500 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: out_index = 2 | new_bitcoin_tx_id
```

RGB++ 资产在 BTC 上分发

Example Code

BTC 交易示例

CKB 交易示例

```
# BTC TX
utxo#2 => commitment, utxo#3, utxo#4, utxo#5, utxo#6, ...

# CKB TX
input:
    rgbpp_cell:
        amount: 21,000,000
        type: xudt
        lock: RGBPP_lock, utxo#2

output:
    rgbpp_cell (utxo#3), # <= change cell

    rgbpp_cell:
        amount: 1000
        type: xudt
        lock: RGBPP_lock, utxo#3
    rgbpp_cell:
        amount: 1000
        type: xudt
        lock: RGBPP_lock, utxo#4
    ...
```

RGB++ 资产 Leap (BTC -> CKB)

Example Code

BTC 交易示例

CKB 交易示例

```
# BTC_TX
input:
    btc_utxo_1
output:
    OP_RETURN: commitment
    btc_utxo_2

# CKB_TX
input:
    rgbpp_cell:
        data: 2000 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: out_index | source_tx

output:
    rgbpp_cell:
        data: 800 * 10 ^ 8
        type: xudt
        lock:
            code: BTC_TIME_lock
            args: lock_script | after | new_bitcoin_tx_id

    rgbpp_cell:
        data: 1200 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: btc_utxo_2 => out_index=1 | new_bitcoin_tx_id
```

解锁 BTC Time cells

Example Code

CKB 交易示例

```
# CKB_TX
input:
    btc_time_cell:
        type: xudt
        lock:
            code: BTC_TIME_lock
            args: lock_script | 6 | btx_tx_id

output:
    xudt_cell:
        type: xudt
        lock:
            lock_script

witness:
    # proof of 6 confirmations after btx_tx_id
```

RGB++ 资产 Leap (CKB -> BTC)

Example Code

CKB 交易示例

```
# CKB TX
input:
    xudt:
        data: 2000 * 10 ^ 8
        type: xudt
        lock:
            ckb_address1

output:
    xudt:
        data: 1000 * 10 ^ 8
        type: xudt
        lock:
            ckb_address2

    rgb_xudt:
        data: 1000 * 10 ^ 8
        type: xudt
        lock:
            code: RGBPP_lock
            args: out_index | btc_tx_id
```

Thanks & QA