# Ark

```
intrp    /lib64/ld-linux-x86-64.so.2
bintype  elf
class    ELF64
lang     c
arch     x86
bits     64
machine  AMD x86-64 architecture
os       linux
minopsz  1
maxopsz  16
pcalign  0
subsys   linux
endian   little
stripped true
static   false
linenum  false
lsyms    false
relocs   false
rpath    NONE
binsz    29504
```

```
[0x00401330 15% 87 /bin/true]> pd $r @ main
            ;-- section_end..plt:
            ;-- section..text:
/ (fcn) main 160
|           ; DATA XREF from 0x004013ed (entry0)
|           0x00401330      83ff02         cmp edi, 2                  ; [13] va=0x00401330 pa=0x00001330 sz=12601 vsz=
|       ,=< 0x00401333      7403           je 0x401338                 ;[1]
|       |   0x00401335      31c0           xor eax, eax
|       |   0x00401337      c3             ret
|       |   ; JMP XREF from 0x00401333 (main)
|       `-> 0x00401338      53             push rbx
|           0x00401339      488b3e         mov rdi, qword [rsi]
|           0x0040133c      4889f3         mov rbx, rsi
|           0x0040133f      e84c050000     call sub.fwrite_890         ;[2]
|           0x00401344      be8d444000     mov esi, 0x40448d
|           0x00401349      bf06000000     mov edi, 6
```

```
|Usage: px[afoswqWqQ][f] # Print heXadecimal
| px         show hexdump
| px/        same as x/ in gdb (help x)
| pxa        show annotated hexdump
| pxA        show op analysis color map
| pxb        dump bits in hexdump form
| pxd[124]   signed integer dump (1 byte, 2 and 4)
| pxe        emoji hexdump! :)
| pxi        HexII compact binary representation
| pxf        show hexdump of current function
| pxh        show hexadecimal half-words dump (16bit
)
| pxH        same as above, but one per line
| pxl        display N lines (rows) of hexdump
| pxo        show octal dump
| pxq        show hexadecimal quad-words dump (64bit
)
| pxQ        same as above, but one per line
| pxr[j]     show words with references to flags and
 code
```

```
[0x004013d0 16% 320 /bin/true]> x @ entry0
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E
0x004013d0  31ed 4989 d15e 4889 e248 83e4 f050 544
0x004013e0  c7c0 3044 4000 48c7 c1c0 4340 0048 c7c
0x004013f0  3013 4000 ff15 f65b 2000 f40f 1f44 000
0x00401400  b81f 7260 0055 482d 1872 6000 4883 f80
0x00401410  4889 e576 1bb8 0000 0000 4885 c074 115
0x00401420  bf18 7260 00ff e066 0f1f 8400 0000 000
0x00401430  5dc3 0f1f 4000 662e 0f1f 8400 0000 000
0x00401440  be18 7260 0055 4881 ee18 7260 0048 c1f
0x00401450  0348 89e5 4889 f048 c1e8 3f48 01c6 48d
0x00401460  fe74 15b8 0000 0000 4885 c074 0b5d bf1
0x00401470  7260 00ff e00f 1f00 5dc3 660f 1f44 000
0x00401480  803d c15d 2000 0075 1155 4889 e5e8 6ef
0x00401490  ffff 5dc6 05ae 5d20 0001 f3c3 0f1f 400
0x004014a0  bf18 6e60 0048 833f 0075 05eb 930f 1f0
0x004014b0  b800 0000 0048 85c0 74f1 5548 89e5 ffd
0x004014c0  5de9 7aff ffff 662e 0f1f 8400 0000 000
0x004014d0  4154 55ba 0500 0000 53be 4845 4000 89f
0x004014e0  31ff 4883 c480 488b 2d73 5d20 0064 488
```

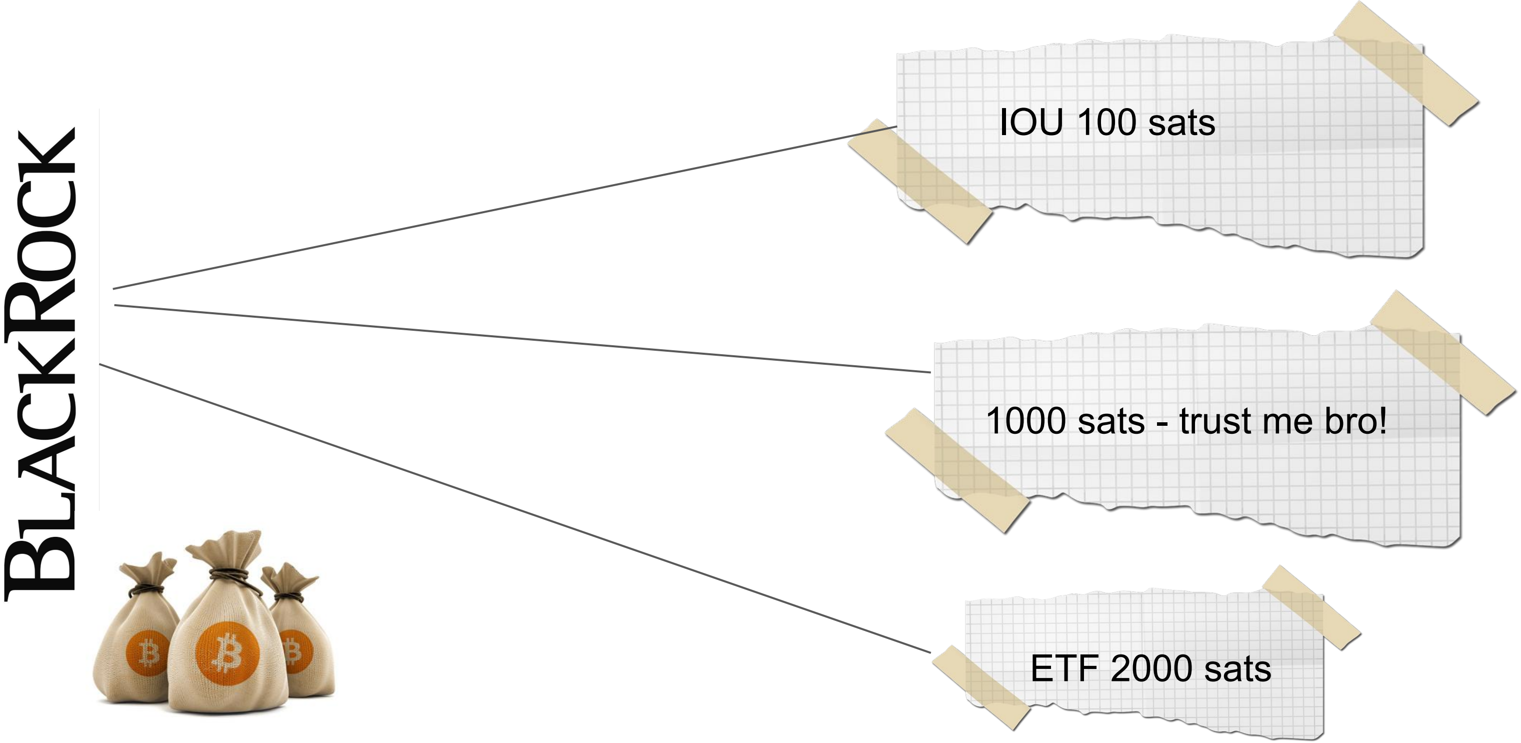
```
[0x004013d0 16% 320 /bin/true]> pxw @ entry0
0x004013d0  0x8949ed31 0x89485ed1 0xe48348e2 0x495
0x004013e0  0x4430c0c7 0xc7480040 0x4043c0c1 0xc7c
0x004013f0  0x00401330 0x5bf615ff 0x0ff40020 0x000
0x00401400  0x60721fb8 0x2d485500 0x00607218 0x0ef
0x00401410  0x76e58948 0x0000b81b 0x85480000 0x5d10
0x00401420  0x607218bf 0x66e0ff00 0x00841f0f 0x000
0x00401430  0x1f0fc35d 0x2e660040 0x00841f0f 0x000x
0x00401440  0x607218be 0x81485500 0x607218ee 0xfecp
0x00401450  0xe5894803 0x48f08948 0x483fe8c1 0xd14
0x00401460  0xb81574fe 0x00000000 0x74c08548 0x18b
0x00401470  0xff006072 0x001f0fe0 0x0f66c35d 0x000r
0x00401480  0x5dc13d80 0x75000020 0x89485511 0xff6
0x00401490  0xc65dffff 0x205dae05 0xc3f30100 0x004
0x004014a0  0x606e18bf 0x3f834800 0xeb057500 0x001.
0x004014b0  0x000000b8 0xc0854800 0x4855f174 0xd0f
0x004014c0  0xff7ae95d 0x2e66ffff 0x00841f0f 0x000
0x004014d0  0xba555441 0x00000005 0x4548be53 0xfb8
0x004014e0  0x8348ff31 0x8b4880c4 0x205d732d 0x8b4x
0x004014f0  0x00282504 0x89480000 0x31782444 0xfbfp
```

```
0x00405100 6 str.POSIX
0x00405106 6 str.ASCII
0x0040510c 9 str._usr_lib
0x00405115 16 str.CHARSETALIASDIR
0x00405125 10 str._50s__50s
0x00401330 256 main
0x004013d0 1 entry0
0x00000000 0 section.
0x00000000 0 section_end.
0x00400238 28 section..interp
0x00400254 0 section_end..interp
0x00400254 32 section..note.ABI_tag
0x00400274 0 section_end..note.ABI_tag
0x00400274 36 section..note.gnu.build_id
0x00400298 0 section_end..note.gnu.build_id
0x00400298 92 section..gnu.hash
0x004002f4 0 section_end..gnu.hash
0x004002f8 1320 section..dynsym
0x00400820 0 section_end..dynsym
0x00400820 587 section..dynstr
```

```
[0x004013d0 16% 256 /bin/true]> pd $r @ entry0
            ;-- entry0:
            0x004013d0      31ed           xor ebp, ebp
            0x004013d2      4989d1         mov r9, rdx
            0x004013d5      5e             pop rsi
            0x004013d6      4889e2         mov rdx, rsp
            0x004013d9      4883e4f0       and rsp, 0xfffffffffffffff0
            0x004013dd      50             push rax
            0x004013de      54             push rsp
            0x004013df      49c7c0304440.  mov r8, 0x404430
            0x004013e6      48c7c1c04340.  mov rcx, 0x4043c0
            0x004013ed      48c7c7301340.  mov rdi, 0x401330           ; section..text
            0x004013f4      ff15f65b2000   call qword [rip + 0x205bf6] ; [0x606ff0:8]=0 LEA reloc.__libc_start_main_240
            0x004013fa      f4             hlt
            0x004013fb      0f1f440000     nop dword [rax + rax]
            0x00401400      b81f726000     mov eax, 0x60721f           ; "6.1.1 20160802" @ 0x60721f
```

```
[0x004013d0 16% 2560 /bin/true]> pxA @ entry0
0x004013d0
0x00401446
0x004014b0
0x00401528
0x004015b8
0x0040166e
0x004016f0
0x0040176f
0x004017f4
0x00401863
0x004018d7
0x00401952
0x004019b6
0x00401a0f
0x00401ab7
0x00401b47
0x00401bdd
0x00401c52
0x00401cec
```

# Solution?



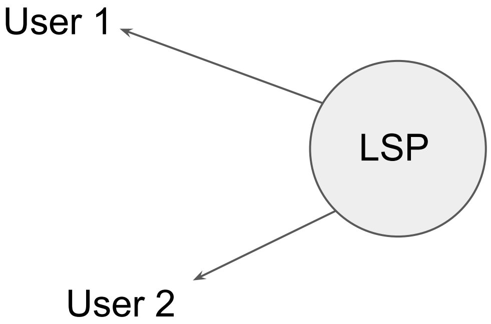
IOU 100 sats

1000 sats - trust me bro!

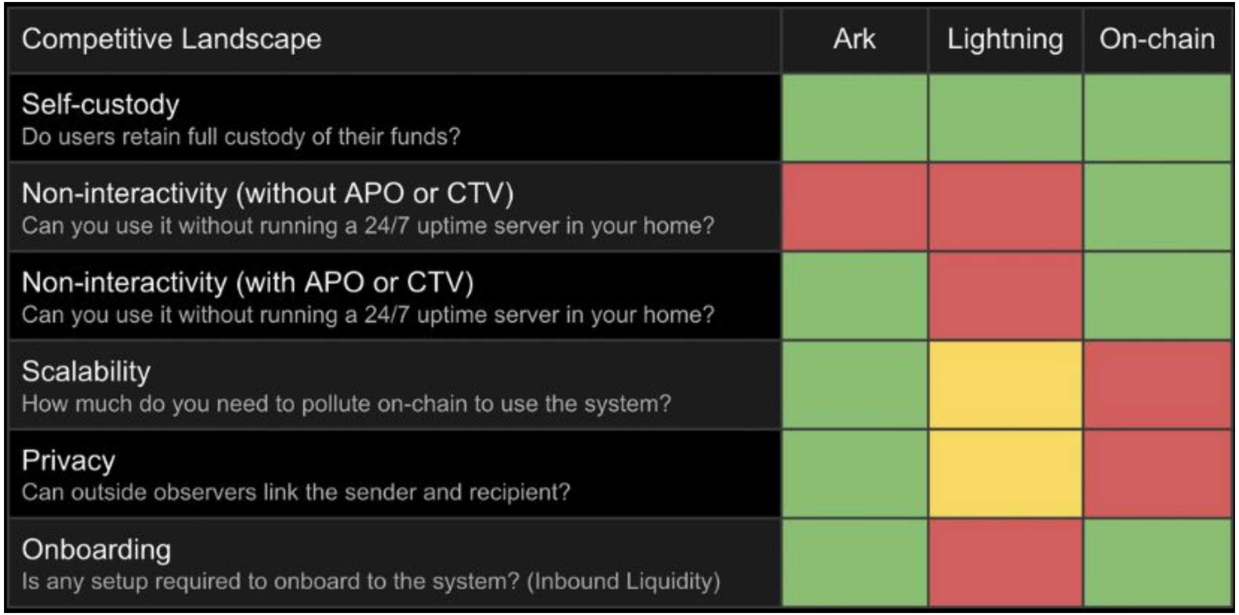
ETF 2000 sats

# Lightning?

- Lightning node always-on, BUT we want to avoid custodial solutions
- Intermediate states are offchain - funding tx needs to hit chain
- **Inbound liquidity issue**
  - bad UX
  - pay somebody to open channel with you - but how much do you need?
  - main reason **Burak Keceli** published Ark idea (first called TBDXXX)



# Scaling bitcoin

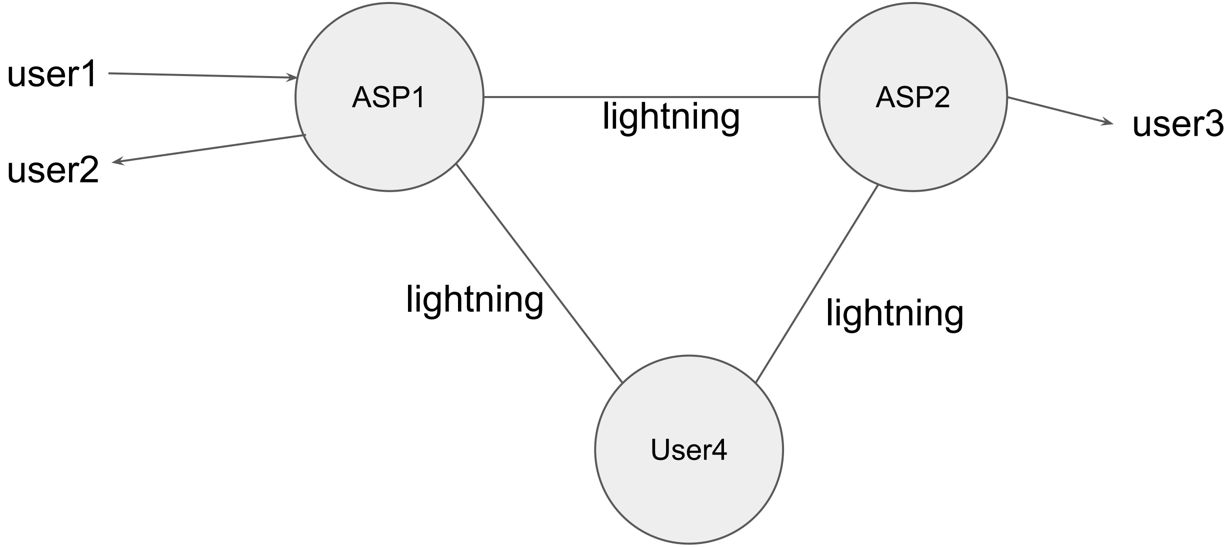


| Competitive Landscape | Ark | Lightning | On-chain |
|---|---|---|---|
| Self-custody<br>Do users retain full custody of their funds? | 🟩 | 🟩 | 🟩 |
| Non-interactivity (without APO or CTV)<br>Can you use it without running a 24/7 uptime server in your home? | 🟥 | 🟥 | 🟩 |
| Non-interactivity (with APO or CTV)<br>Can you use it without running a 24/7 uptime server in your home? | 🟩 | 🟥 | 🟩 |
| Scalability<br>How much do you need to pollute on-chain to use the system? | 🟩 | 🟨 | 🟥 |
| Privacy<br>Can outside observers link the sender and recipient? | 🟩 | 🟨 | 🟥 |
| Onboarding<br>Is any setup required to onboard to the system? (Inbound Liquidity) | 🟩 | 🟥 | 🟩 |

Source: arkpill.me

# Ark overview

- Ark Service Provider (ASP)
- ASP is a LSP so it has lightning, users can instruct it to pay an invoice
- System in **permissioned** but you don’t need to trust ASPs



# Ark terms

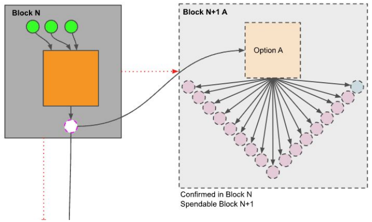
- UTXO -> VTXO
  - VTXO = unpublished UTXO, but validity is 4 weeks in Ark
- Hypothetical OP_TXHASH
  - does this TX (X) exist in UTXO set
  - we can emulate it with “connectors”
  - connector is an output with 450 sats (> dust value)
  - other TX (Y) uses connector as input to make sure X is confirmed
- Buzzword “ATLC” (ala HTLC)

# Covenants

- Restrict how coins can be spent further than just requiring valid sig
- A scaling solution on it’s own
- Check https://utxos.org/uses/

- OP_CTV - BIP 119 soft-fork required
- Check Template Verify (possible tx ids spending)
- One level ahead
- Could be emulated with APO (BIP 118)

Congestion Controlled Transactions



# ASP

- tick-tock next ASP tx
- every 5 s for good UX
  - -> one ASP uses 4% of blockspace this way

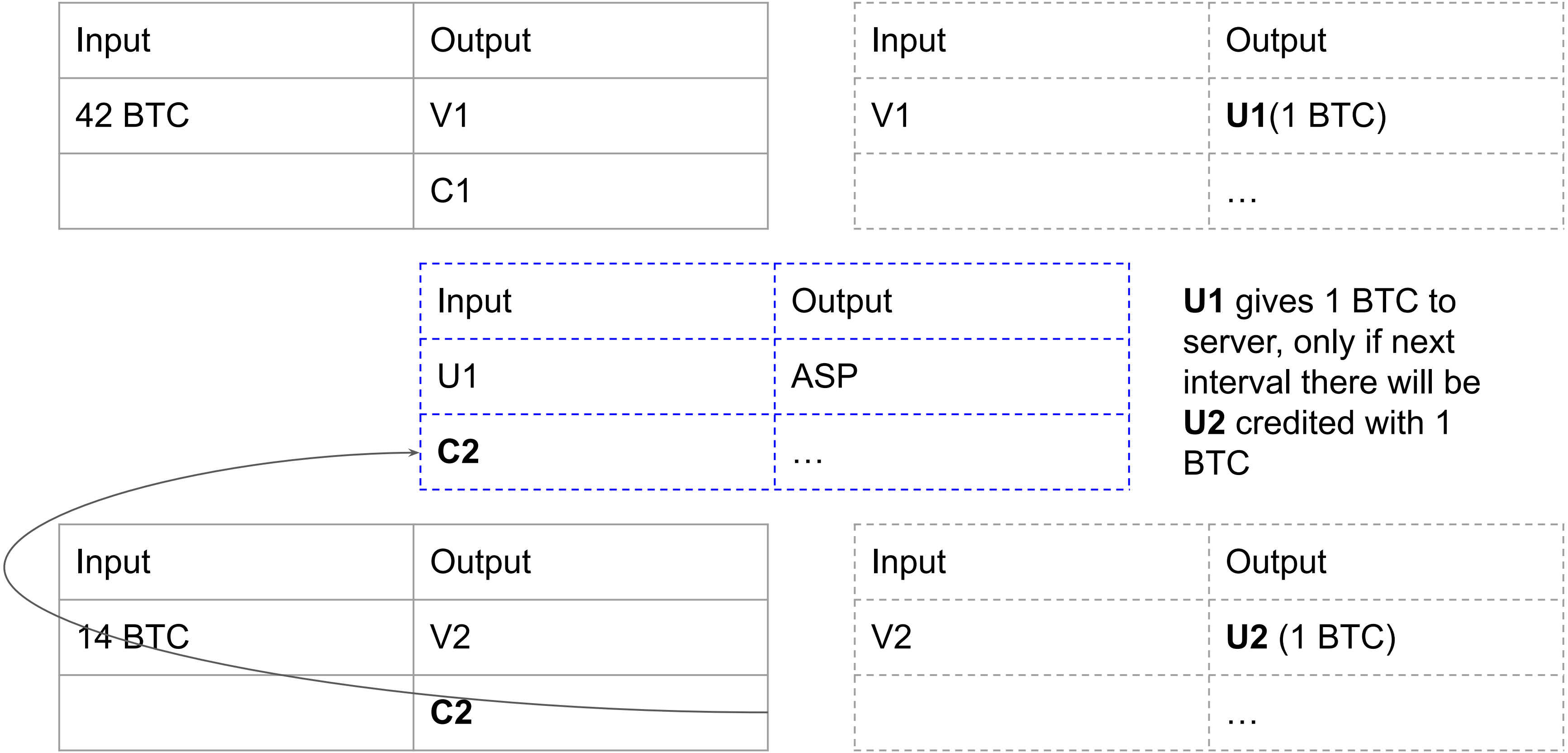
| Input | Output |
|---|---|
| 42 BTC | V1 |
|  | connectors… |

| Input | Output |
|---|---|
| V1 | U1(1 BTC) * |
|  | U2 (1 BTC) |
|  | U3 (1 BTC) |
|  | … |

42/42 multisig (or CTV)
U1, U2 are VTXOs (also unpublished)
Instead of U1 it could also be HTLC for lightning

* actually it is (U1 && ASP) || (U1 && timelock)

# ASP tx



| Input | Output |
|---|---|
| 42 BTC | V1 |
| | C1 |

| Input | Output |
|---|---|
| V1 | **U1**(1 BTC) |
| | … |

| Input | Output |
|---|---|
| U1 | ASP |
| **C2** | … |

| Input | Output |
|---|---|
| 14 BTC | V2 |
| | **C2** |

| Input | Output |
|---|---|
| V2 | **U2** (1 BTC) |
| | … |

**U1** gives 1 BTC to server, only if next interval there will be **U2** credited with 1 BTC

# Cheating

| Input | Output |
|---|---|
| 42 BTC | V1 |
|  | C1 |

| Input | Output |
|---|---|
| V1 | **U1**(1 BTC) |
|  | … |

- When you need to exit direct ASP to pay via lightning
- If ASP is uncooperative -> publish the huge tx (U1 VTXO -> U1 UTXO)
- There is always a commitment on-chain
- If user cheats ASP uses “forfeit tx” (which U1 signed)

# ASP onboarding (1/2)

- “Custodial”: 1000 sats

| Input | Output |
|---|---|
| 42 BTC | V1 |
|  | C1 |

| Input | Output |
|---|---|
| V1 | **U1**(1000 sats) |
|  | … |

- Lifting UTXOs (cooperation with ASP, it is atomic however)

| Input | Output |
|---|---|
| 4199999000 sats | V1 |
| **1000 sats** | C1 |

| Input | Output |
|---|---|
| V1 | **U1**(1000 sats) |
|  | … |

# ASP onboarding (2/2)

- Atomic swap your UTXO <-> someone's VTXO
- Instead of U2 you could have script (HTLC) inside the unpublished TX
- Lock time << 4 weeks (or else ASP can rug you)

# ASP GC

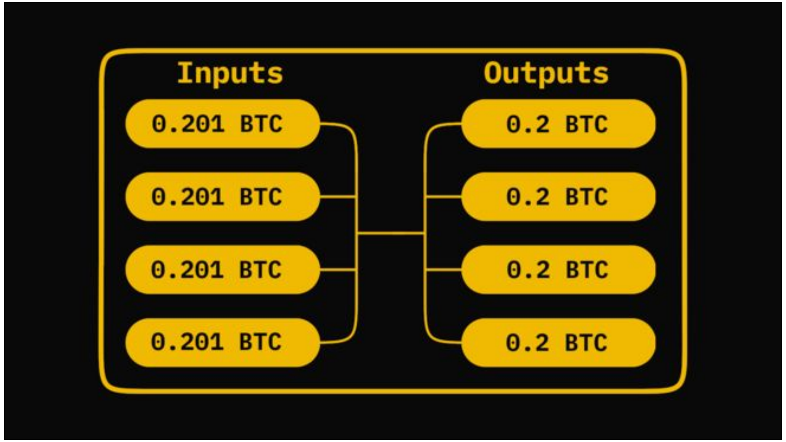
- ASP does not need to deal with every single TX
    - else it would still need to pay fees depending on # tx
- Special spending condition CLTV (4 weeks)
    - whole V is claimable by ASP after that time (used as input for new periodic V+n)
- You need to periodically “refresh” VTXOs
    - no need to be online all the time (fixed time requirement vs. dynamic)
    - transfer to self (might also improve anonymity)
    - ASP charges for transactions but it knows VTXO “age”
    - so in theory refresh could be free (but you could also transfer funds to somebody else)
    - simple “watchtowers”?

# Anonymity

- Lightning: sphinx onion routing makes sure node sees just what it needs (PTLC vs. HTLC)
- Ark has either
  - one intermediate hop (ASP)
  - or it is an (almost) normal lightning payment
- In naive way ASP could deduct that U1 pays U2
- However ASP is also **coinjoin** coordinator
- Need fixed amounts of “denominations” 1000 sats, 10k sats, 100k sats..

# Coinjoin

- Wabisabi protocol
  - Register inputs
  - Register outputs
  - Signing
- ASP is a (blinded) coinjoin coordinator
- Is 5 seconds too fast for good anonymity set?



# Discussion

- **Ark is just like a new fast L1 (without a shitcoin!)**
- Huge capital requirements for ASPs (but we can tweak the numbers, less available funds => more expensive transactions become)
- ASP needs fees for transfers to compensate for locked funds and on-chain fees
- Channel jamming -> liquidity draining (are fees enough?)
- Receiver needs to be online (unless we get covenants)
- **Mempool concern**
  - periodic transactions are small (and O(1) in terms of actual user txs)
  - more people start using it, the greater the savings
  - you could have tree like structure for multiple ASPs in one on-chain tx
- Crazy ideas
  - “lightning” over VTXOs
  - or even Ark over Ark

# Additional resources

?

Gregor Pogačnik @fiksn
https://arkpill.me
https://github.com/fiksn/awesome-ark

Bitcoin Ljubljana