Understanding Ark

a new Bitcoin layer 2 protocol

Who am I?

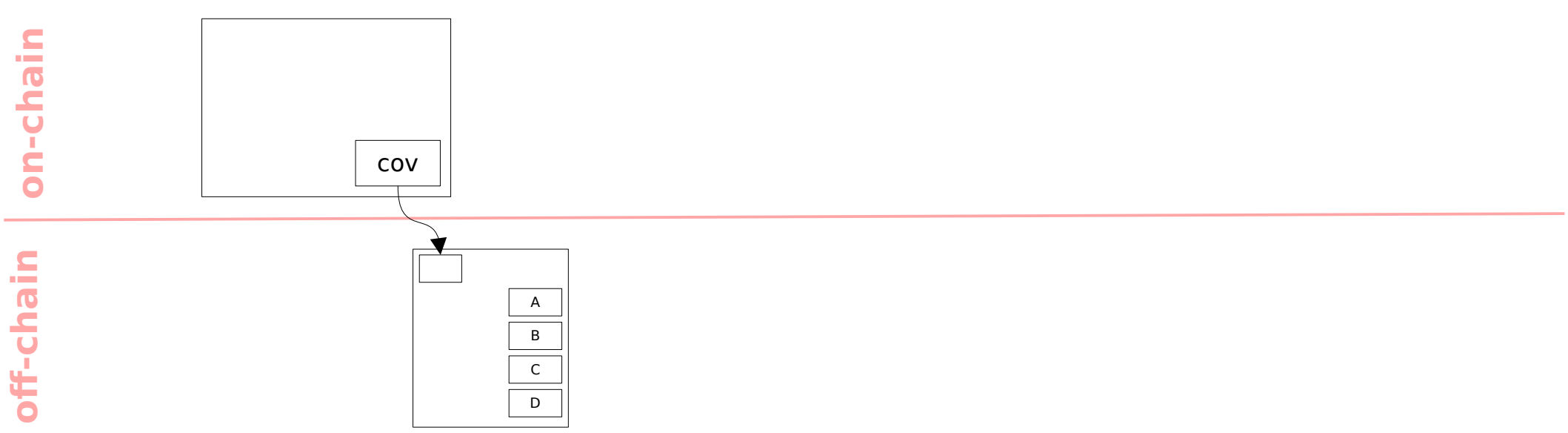
- Steven Roose
- Bitcoin dev for over 10 years
- Liquid team @ Blockstream
- rust-bitcoin

Understanding Ark

- no sales talk, just technical explanation
- from the ground up
  → might slightly differ from other explanations
- assume covenants*

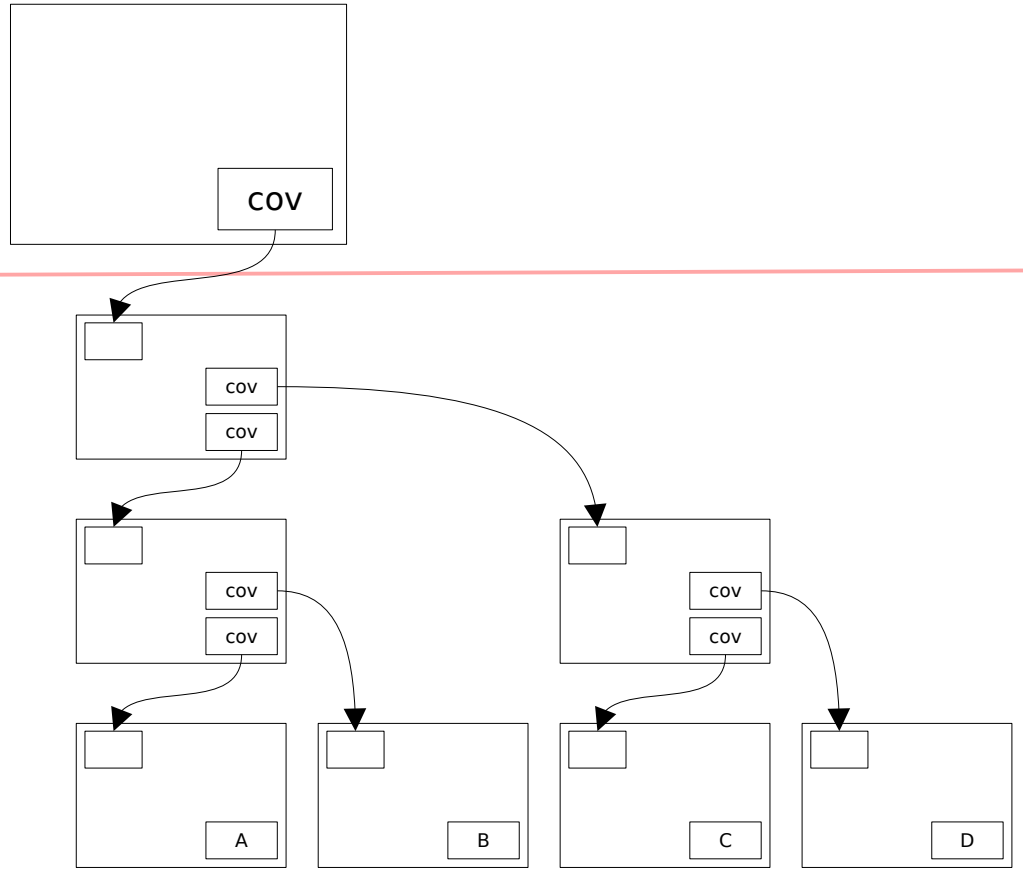
Covenants

- restriction on where the money in a UTXO can go
- for now: an output that can only be spent using a single pre-specified transaction


on-chain
COV
off-chain
A
B
C
D

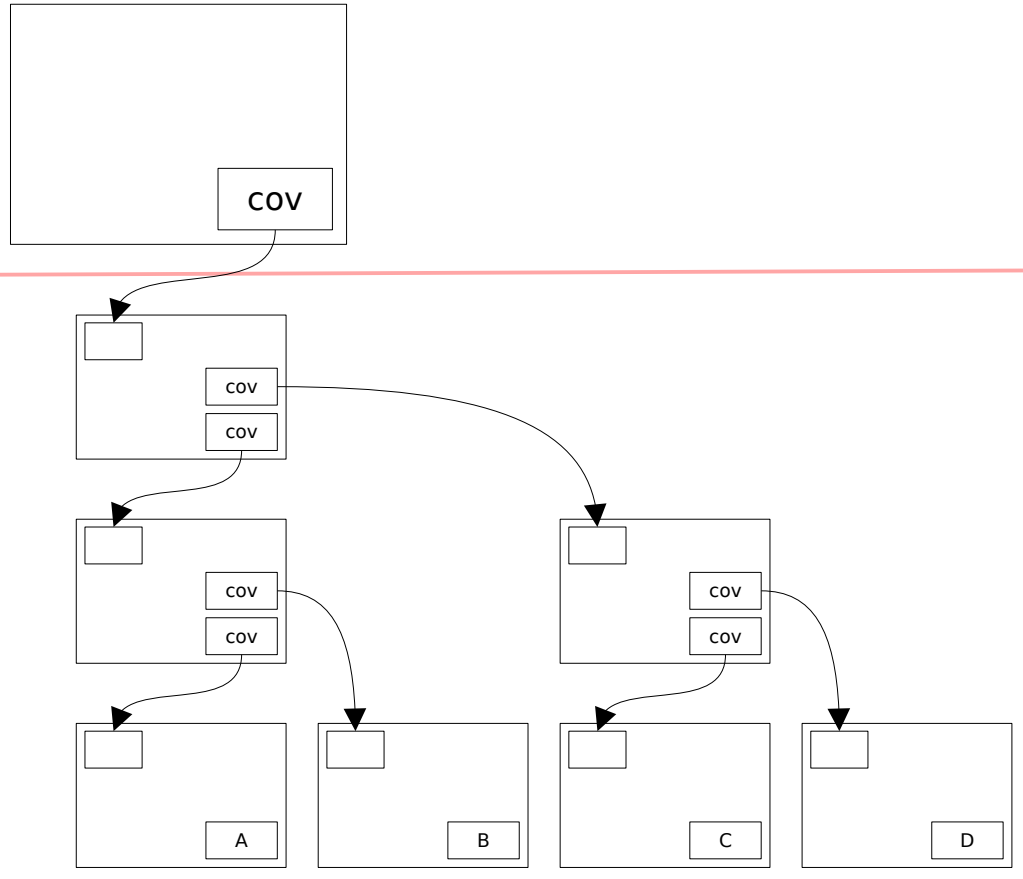
on-chain

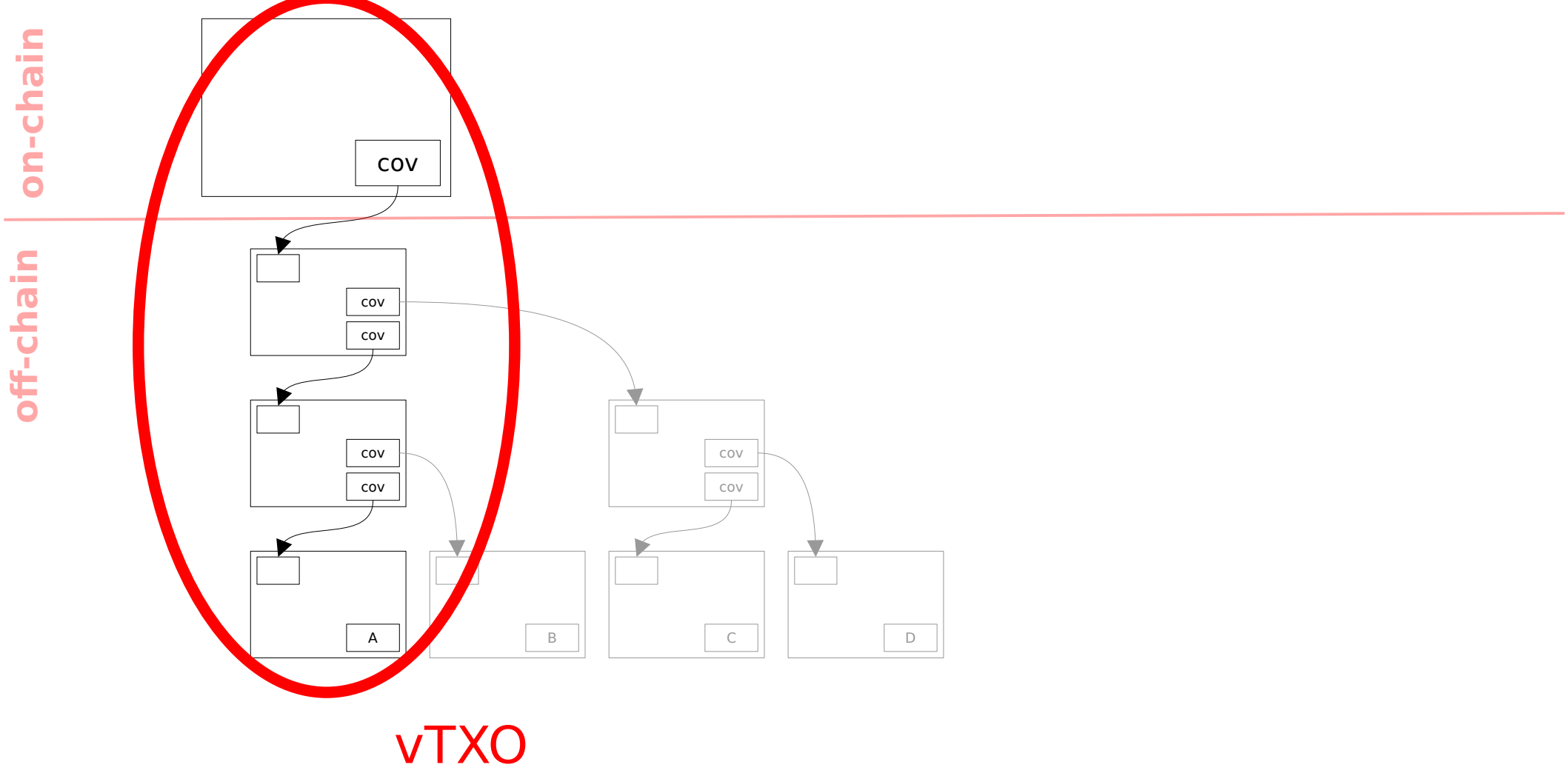
off-chain

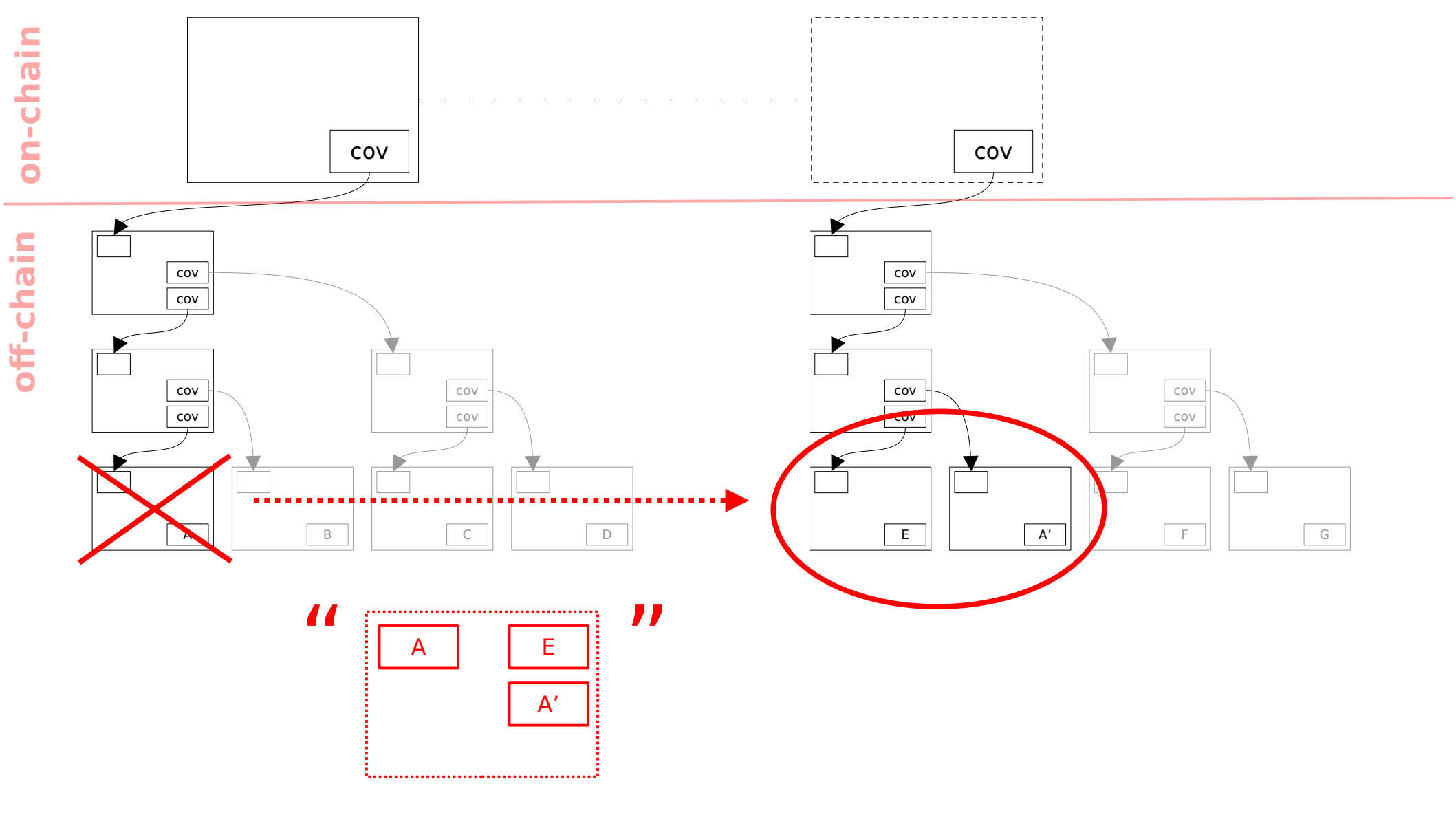


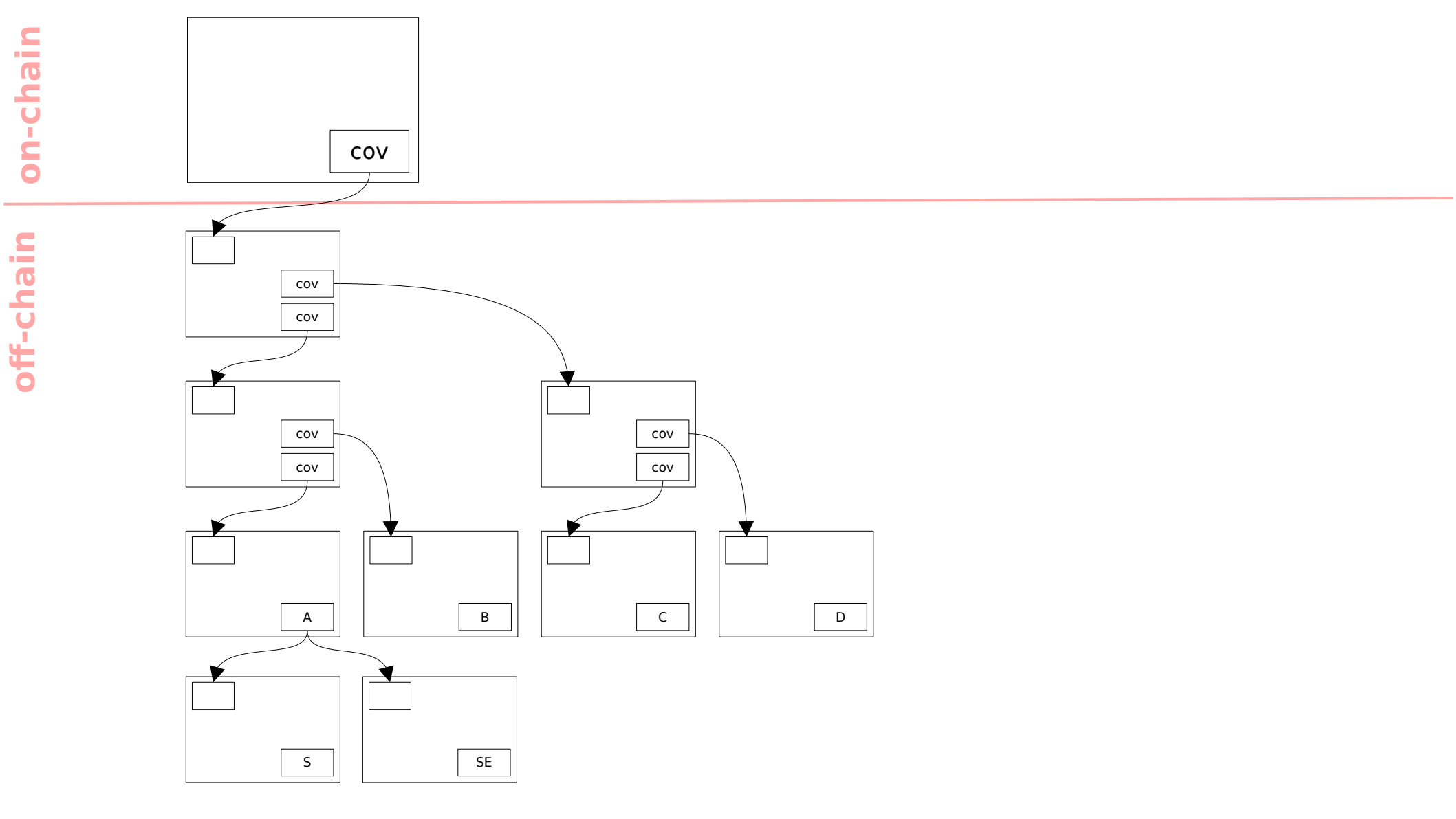
on-chain

off-chain



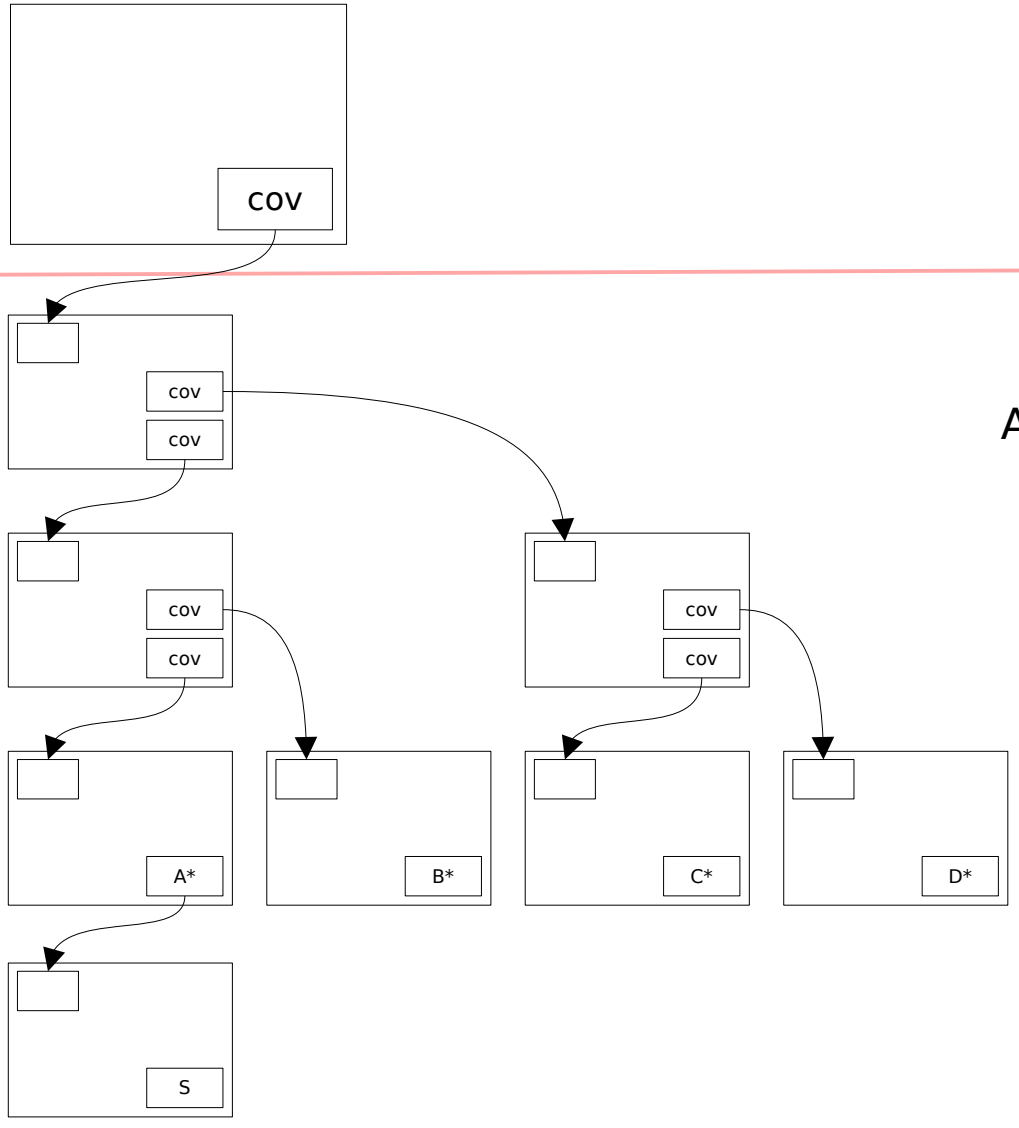

on-chain
off-chain
cov
cov
cov
cov
cov
cov
cov
cov
A
B
C
D
vTXO


on-chain
off-chain
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
A
B
C
D
E
A'
F
G
"
A
E
A'
"


on-chain
COV
off-chain
cov
cov
cov
cov
cov
cov
A
B
C
D
S
SE

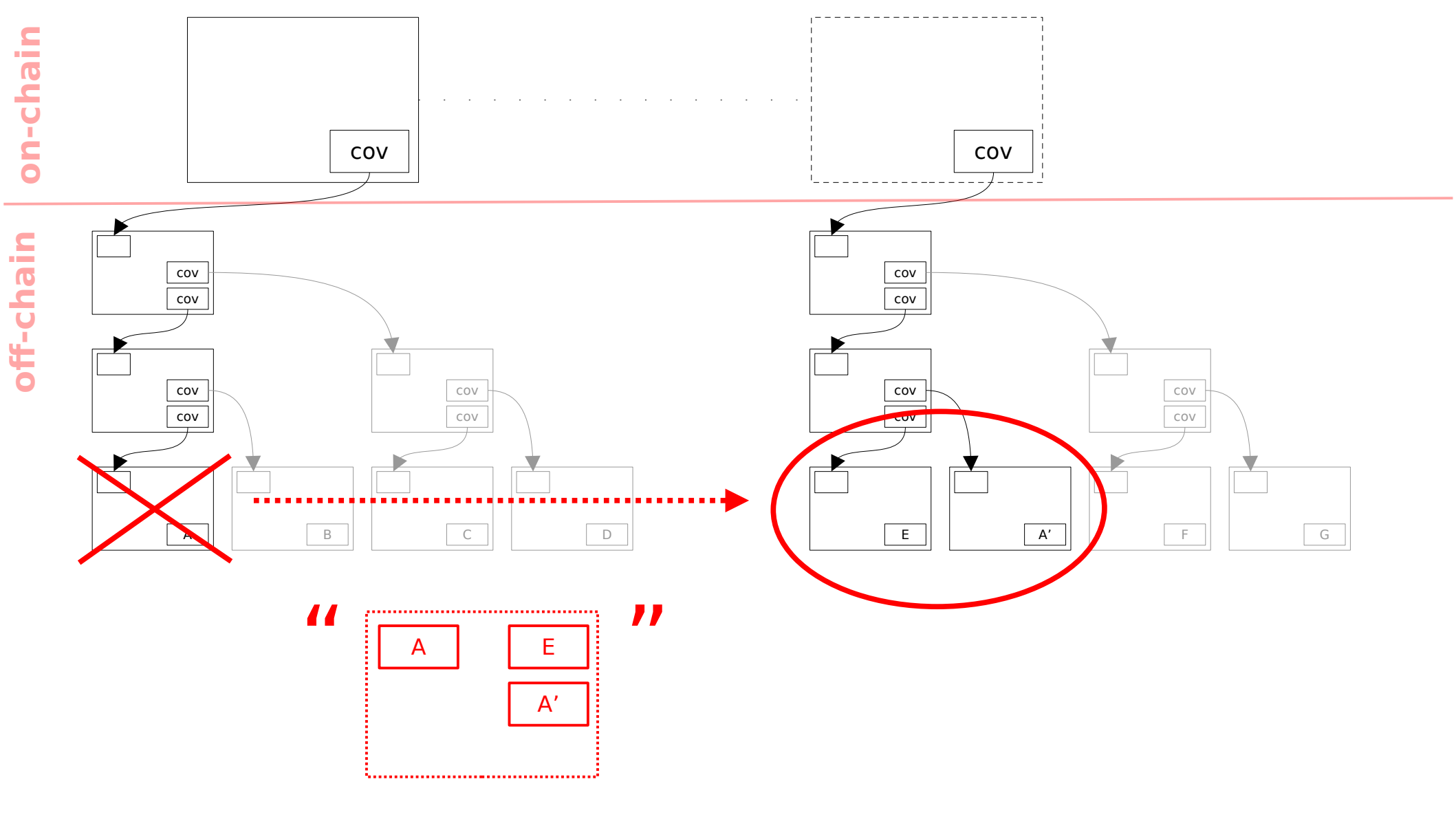
on-chain

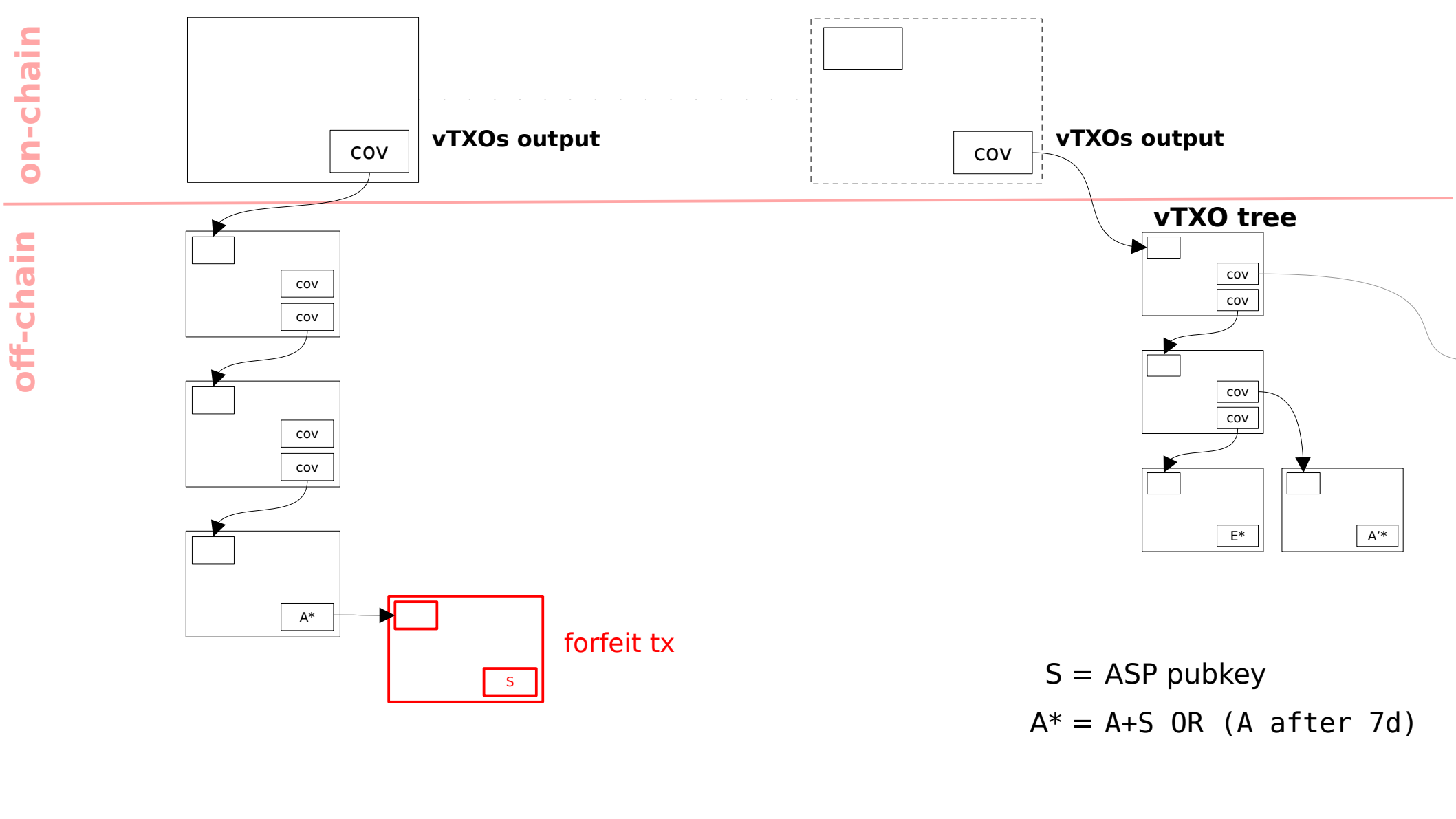
off-chain

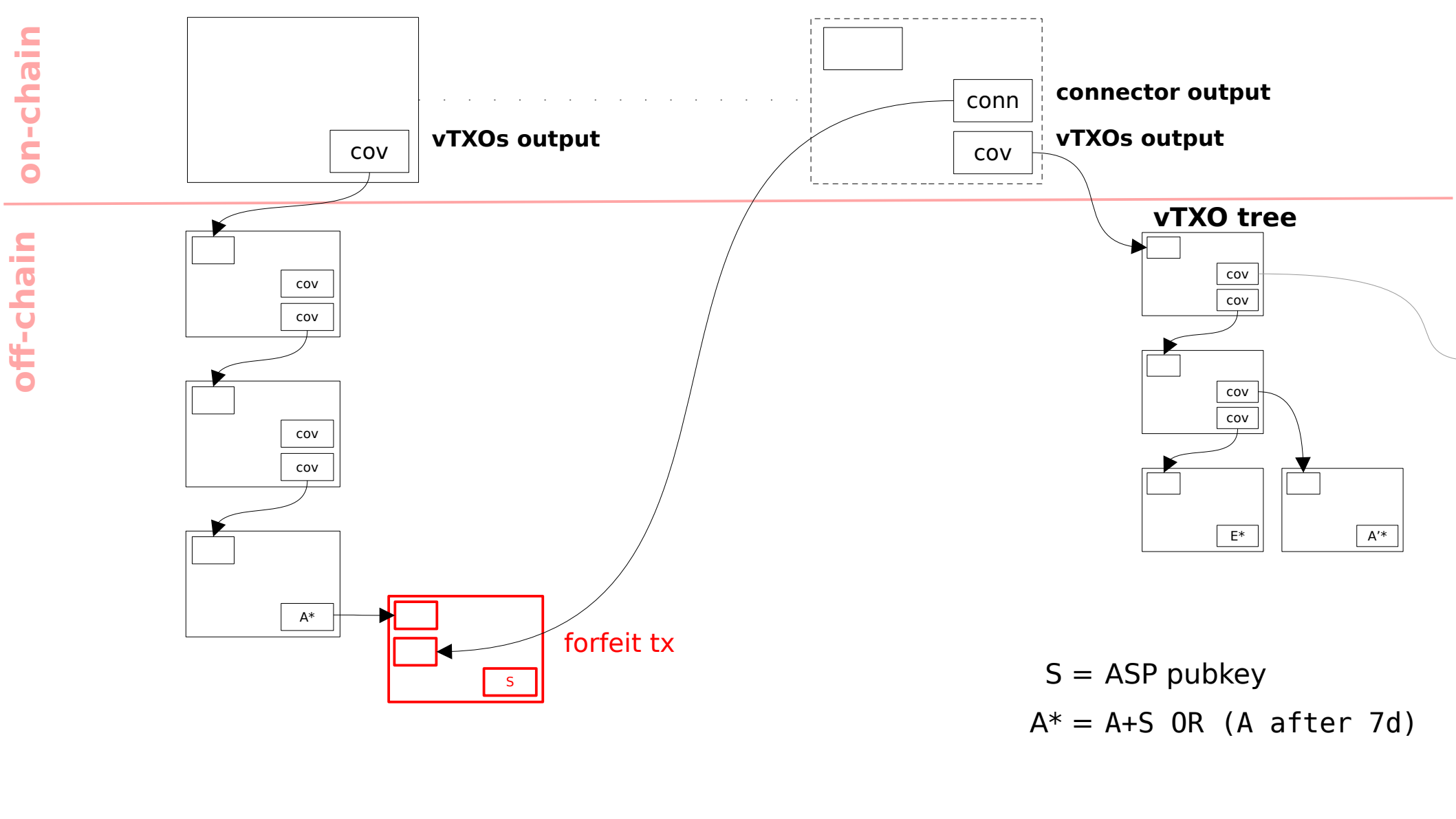


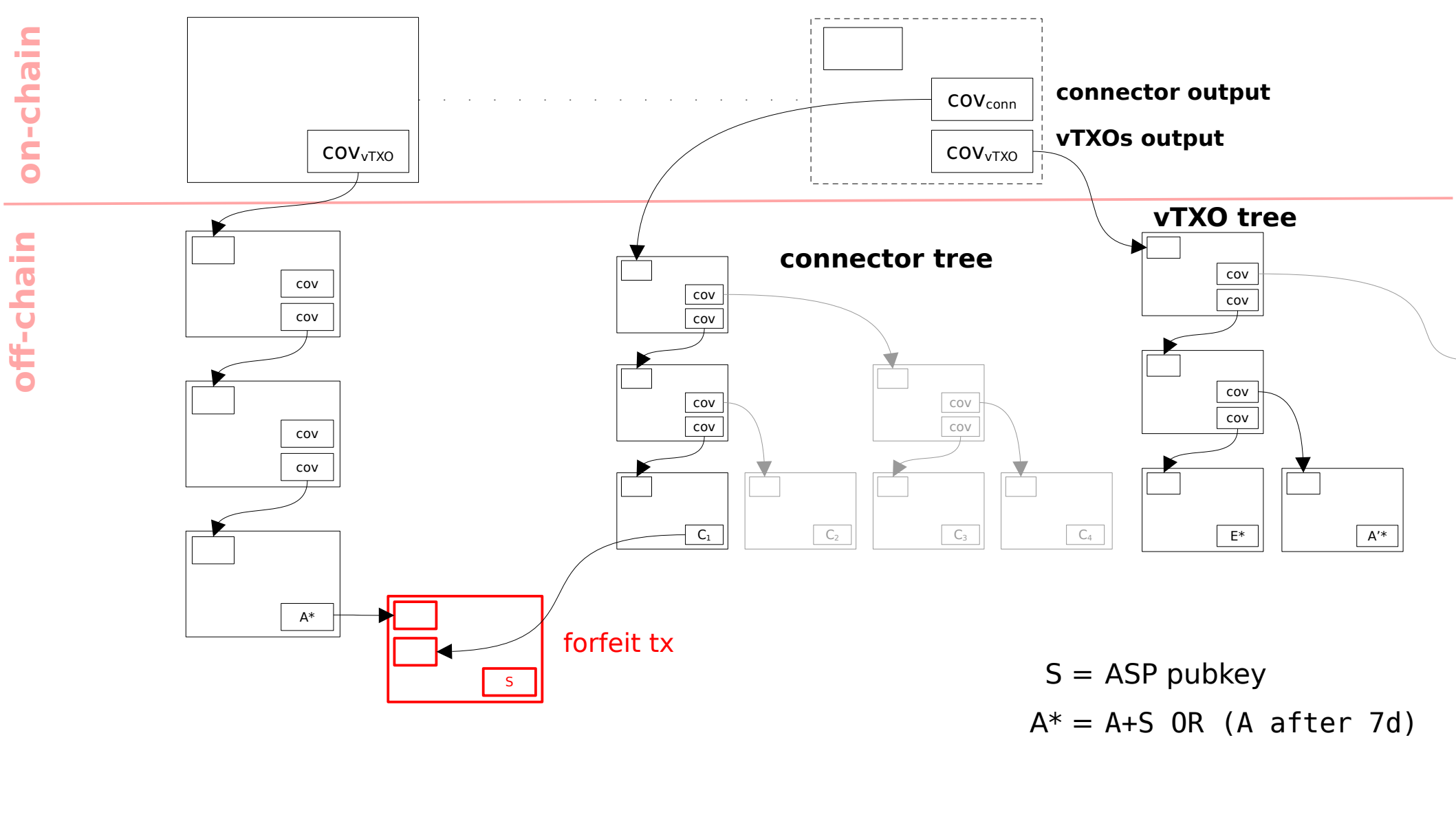
S = ASP pubkey

A* = A+S OR (A after 7d)


on-chain
off-chain
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
cov
A
B
C
D
E
A'
F
G
"
A
E
A'
"

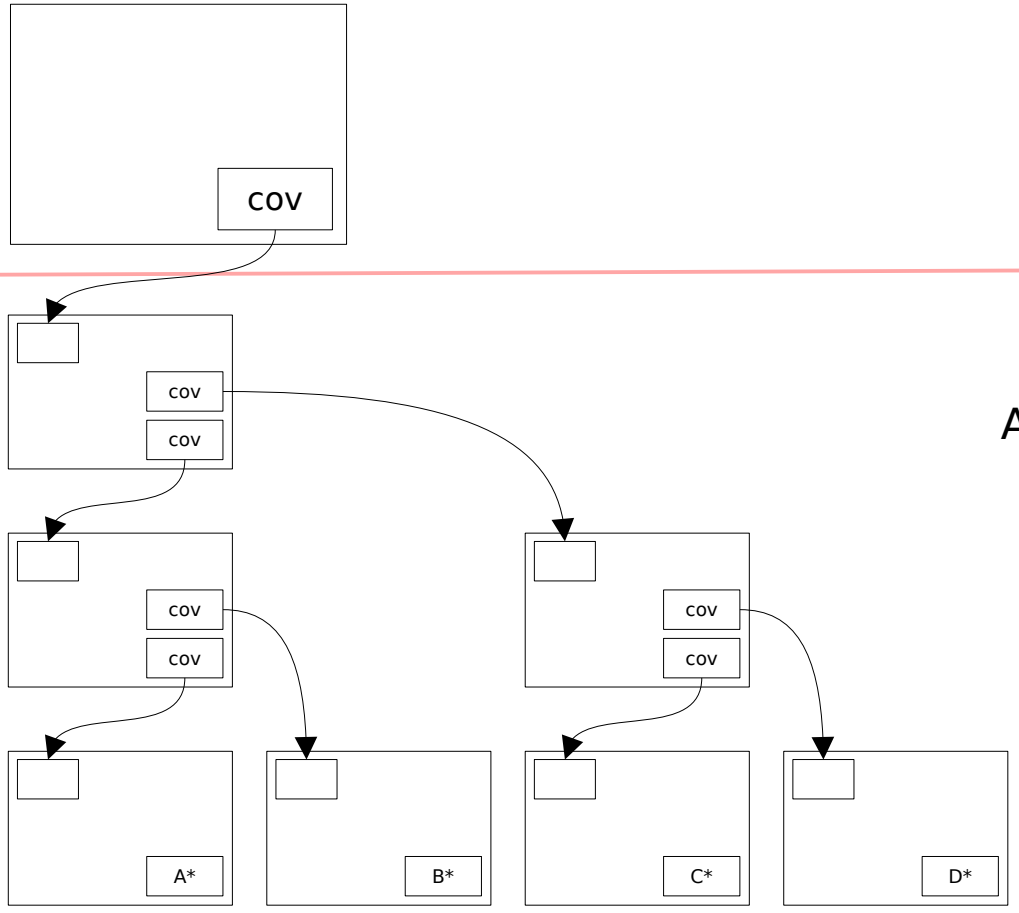

on-chain
off-chain
COV
vTXOs output
COV
vTXOs output
vTXO tree
COV
COV
COV
COV
COV
COV
COV
COV
A*
E*
A'*
forfeit tx
S
S = ASP pubkey
A* = A+S OR (A after 7d)


on-chain
off-chain
cov
vTXOs output
conn
connector output
cov
vTXOs output
vTXO tree
cov
cov
cov
cov
cov
cov
cov
cov
A*
E*
A'*
forfeit tx
S
S = ASP pubkey
A* = A+S OR (A after 7d)


on-chain
off-chain
COV_vTXO
COV_conn
connector output
vTXOs output
vTXO tree
connector tree
cov
C1
C2
C3
C4
E*
A'*
A*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)

on-chain

off-chain

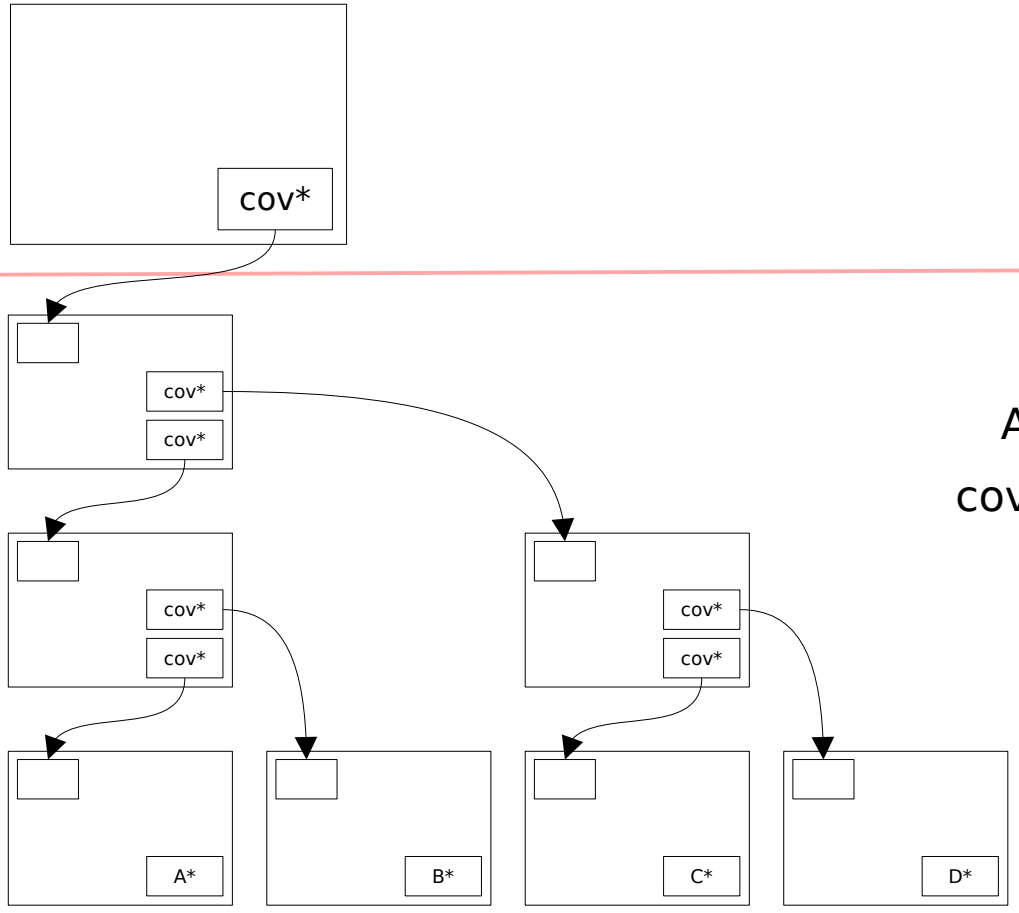


S = ASP pubkey

A* = A+S OR (A after 7d)

on-chain

off-chain

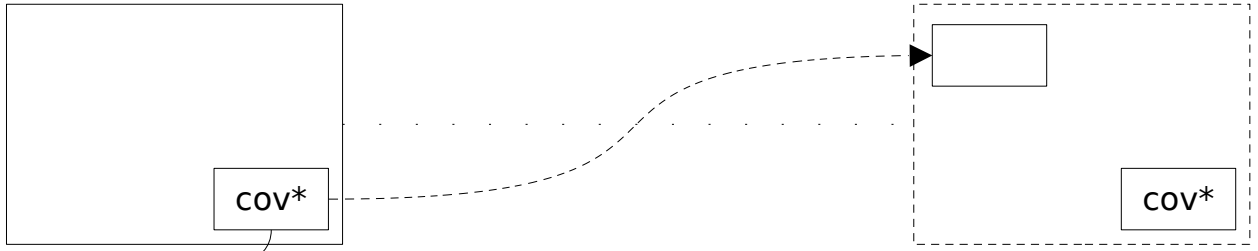


S = ASP pubkey

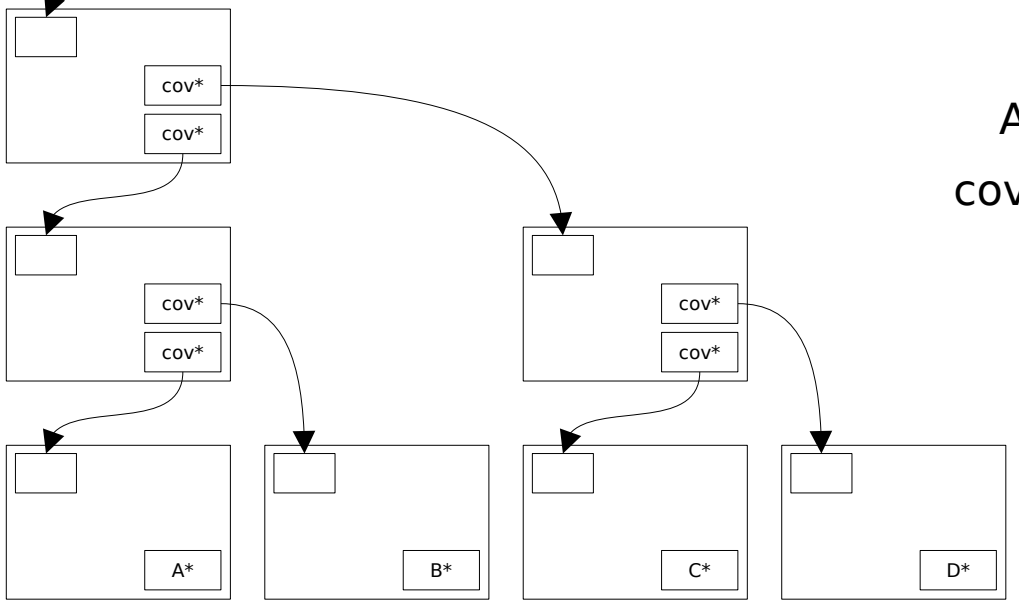
A* = A+S OR (A after 7d)

cov* = cov OR (S after 14d)

on-chain



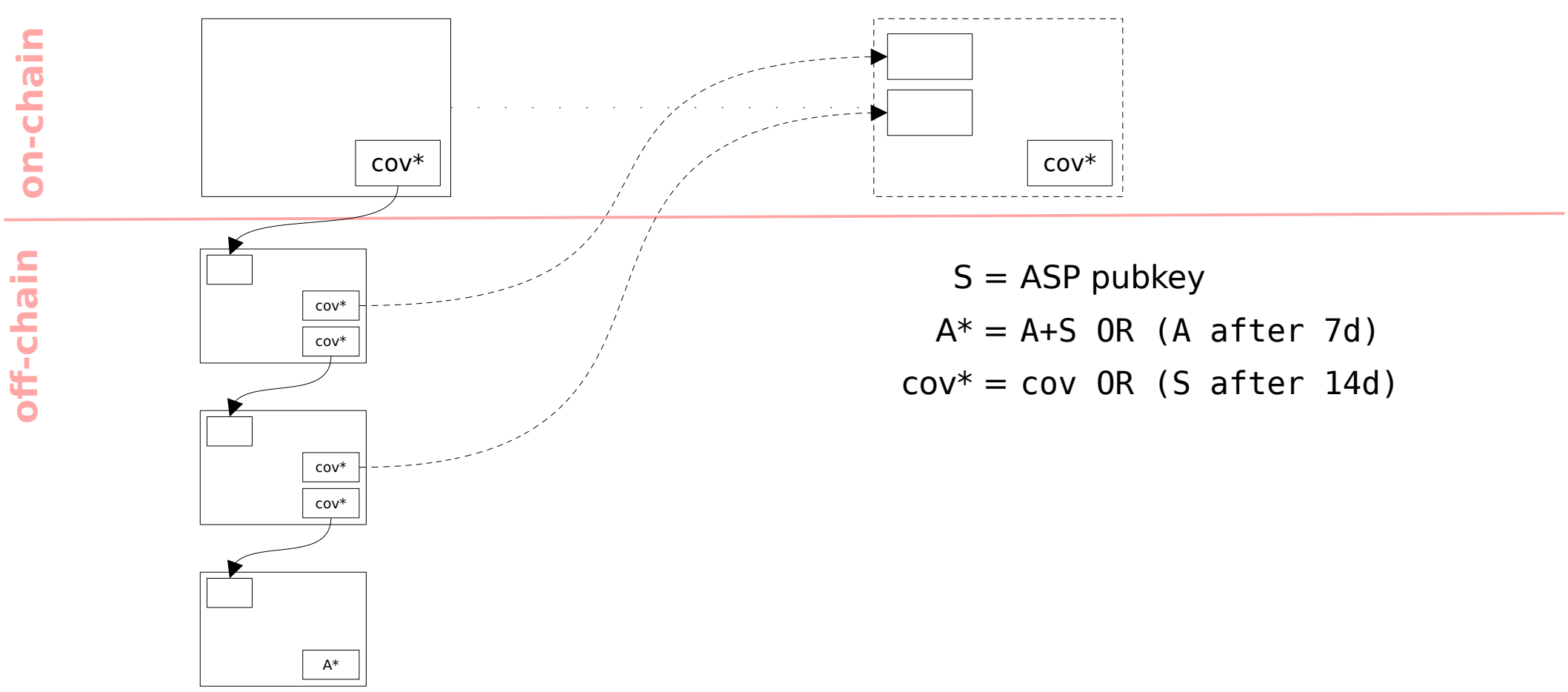
off-chain



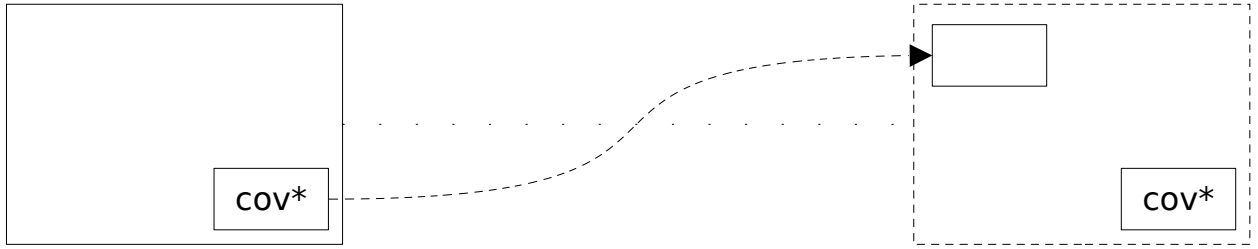
S = ASP pubkey

A* = A+S OR (A after 7d)

cov* = cov OR (S after 14d)


on-chain
off-chain
cov*
cov*
cov*
cov*
cov*
cov*
A*
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

on-chain

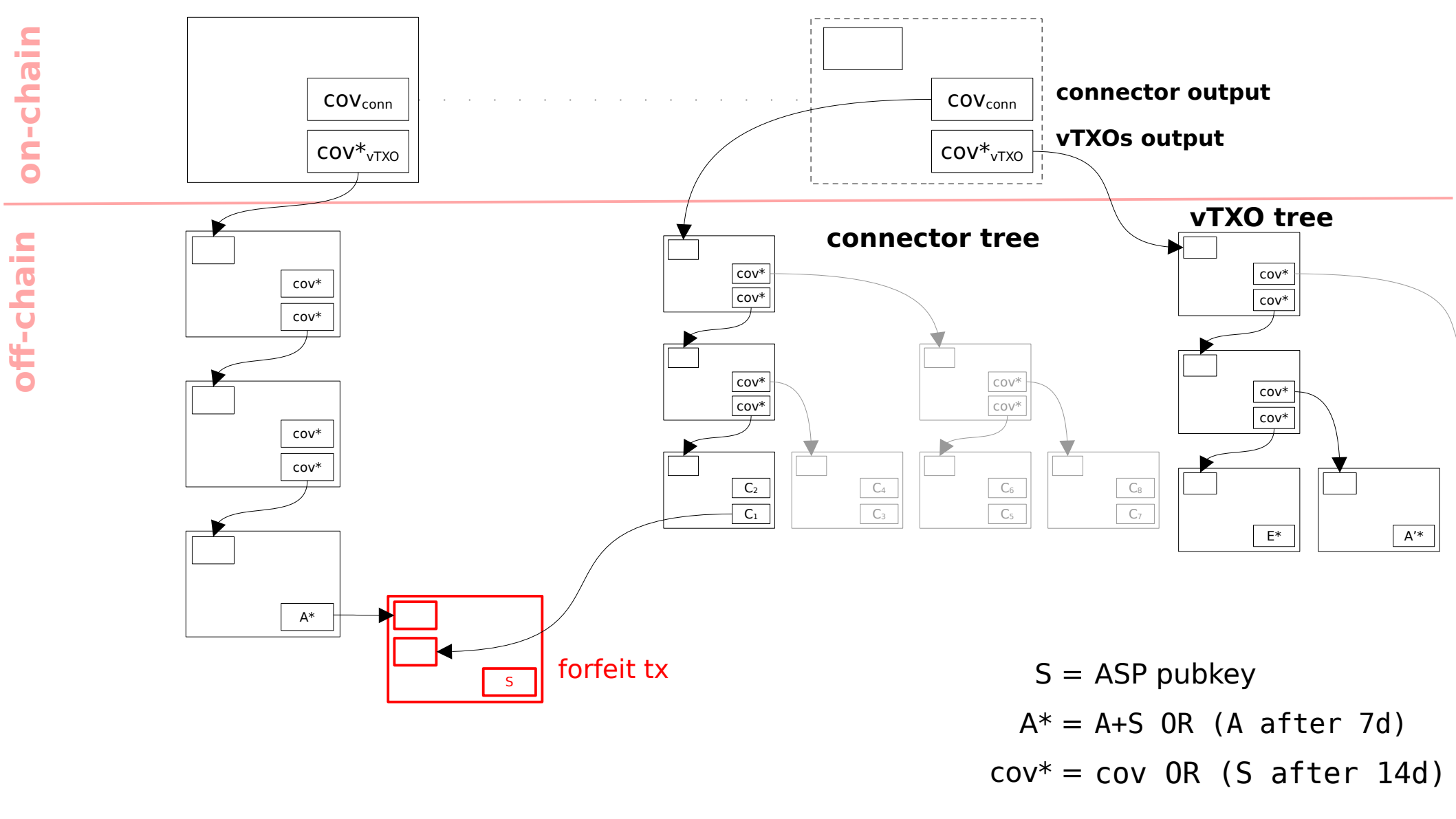


off-chain

S = ASP pubkey

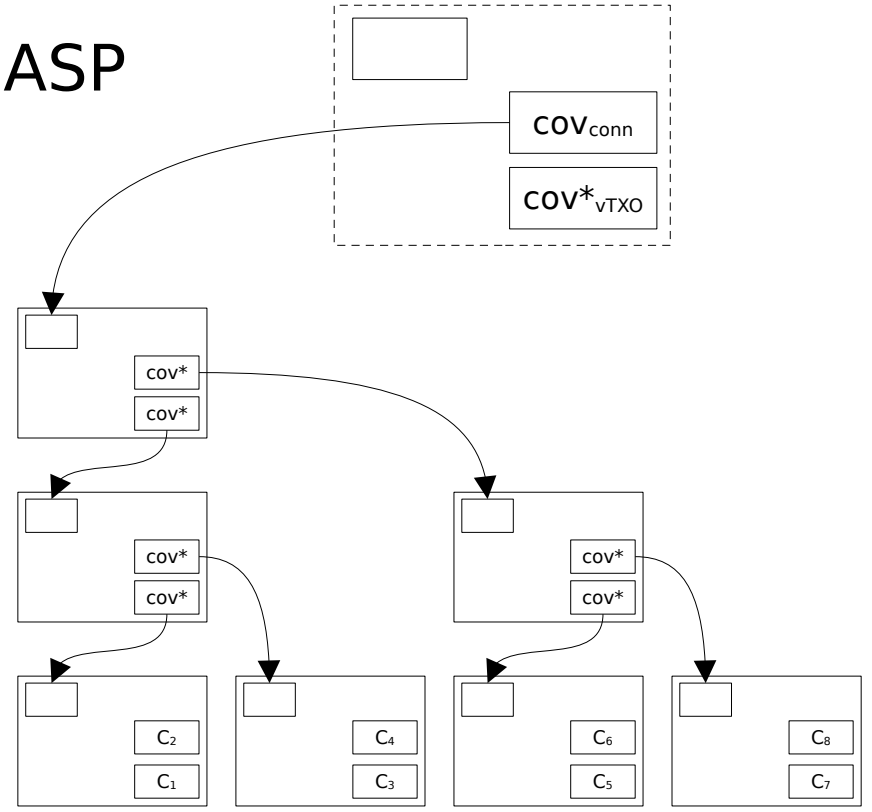
A* = A+S OR (A after 7d)

cov* = cov OR (S after 14d)


on-chain
off-chain
COV_conn
COV*_vTXO
COV_conn
COV*_vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
C2
C1
C4
C3
C6
C5
C8
C7
E*
A'*
A*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

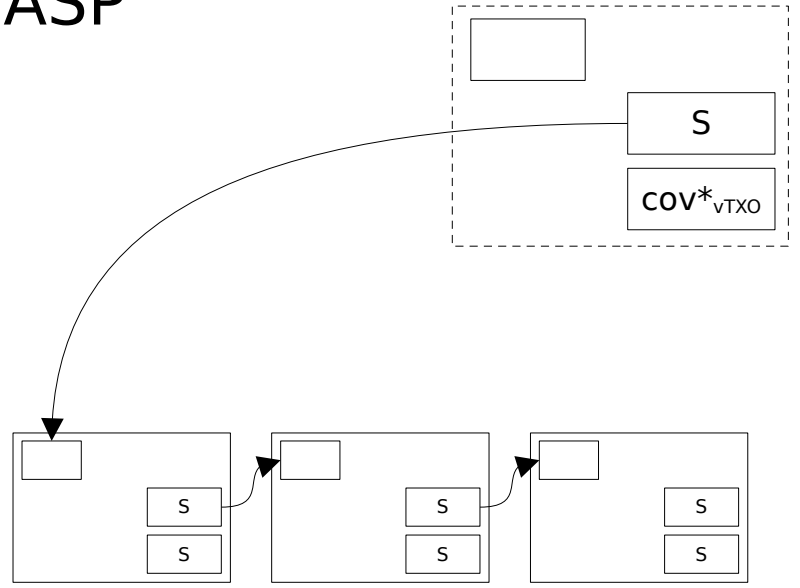
Connectors

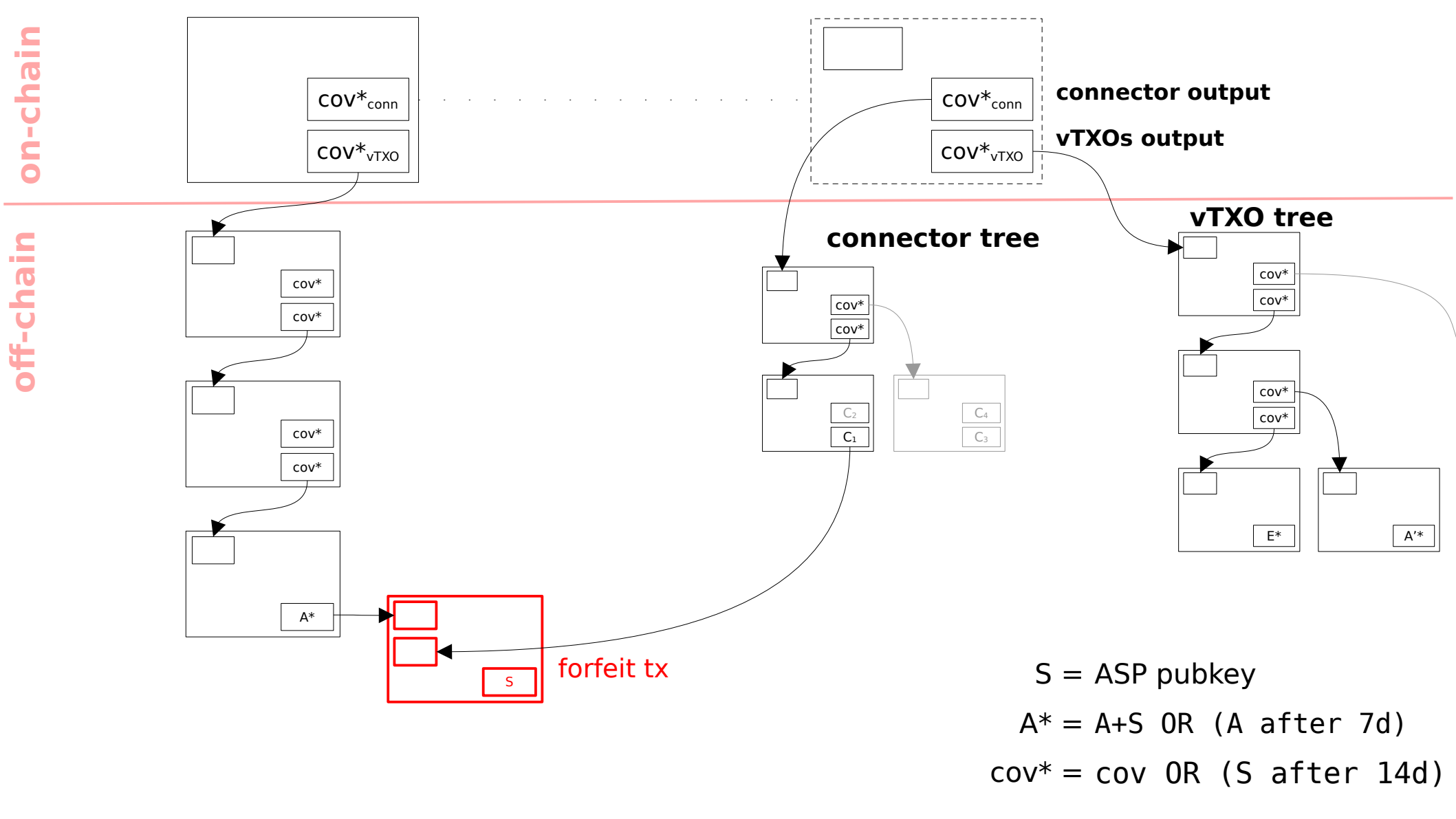
- users only care about tx dep. chain
- simple 1-of-1 outputs owned by ASP
- ideally 0-value
  - → rely on CPFP & package relay



Connectors

- users only care about tx dep. chain
- simple 1-of-1 outputs owned by ASP
- ideally 0-value
- alternatively single chain
  - → users sign multiple forfeit txs




on-chain
off-chain
cov*conn
cov*vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
C2
C1
C4
C3
E*
A'*
A*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

What is (an) Ark?

- series of “Ark rounds”
  - → atomic spending vTXOs to create new ones
  - → one on-chain Ark tx with vTXO & connector outputs
- single service provider: “ASP”
  - → coordinates & provides liquidity*
  - → users always 100% in control of money

What is (an) Ark?

- efficient UTXO-style off-chain txs
- only client-server interactions needed, no p2p
- anyone can receive (no liquidity required)
- flexible round times
  - → confirmation when Ark tx confirms

What is (an) Ark?

- efficient UTXO-style off-chain txs
- only client-server interactions needed, no p2p
- anyone can receive (no liquidity required)
- flexible round times
- vTXO expiry
  - → watchtower-based automatic vTXO refresh?

Ark round flow

- ASP announces start of new round
- spenders indicate input & output vTXOs
- ASP assembles an Ark tx and sends it to spenders
  - → vTXO tree output
  - → connector tree output
  - → input liquidity & potential change output (can be a vTXO too!)
- spenders sign their forfeit txs
  - → using individually assigned connector output
- outputs of spenders that refuse signing are dropped
  - → ASP creates new Ark tx and spenders sign again, etc..*

Let’s dig a little deeper

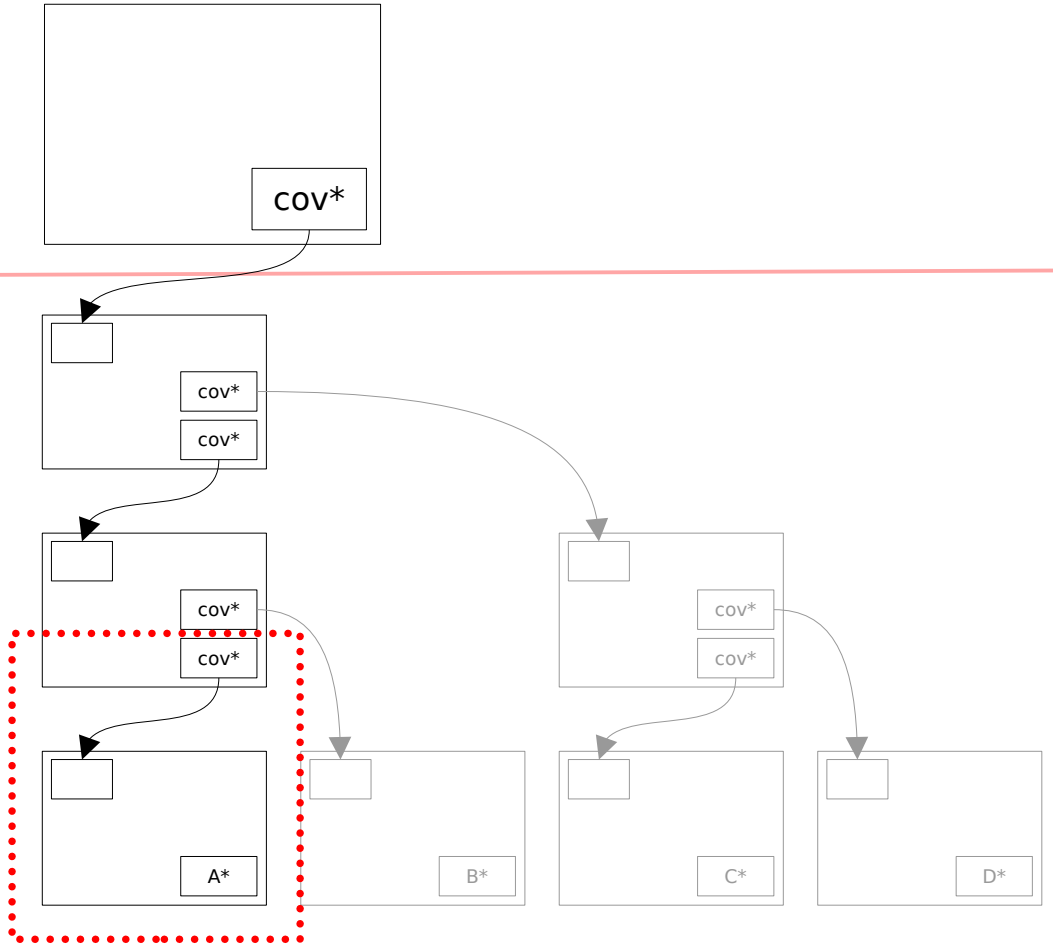
Lifting

- mechanism to enter and exit the Ark (boarding?)
- non-interactive lift-in

  → straight from on-chain to vTXO
- interactive lift-out

on-chain

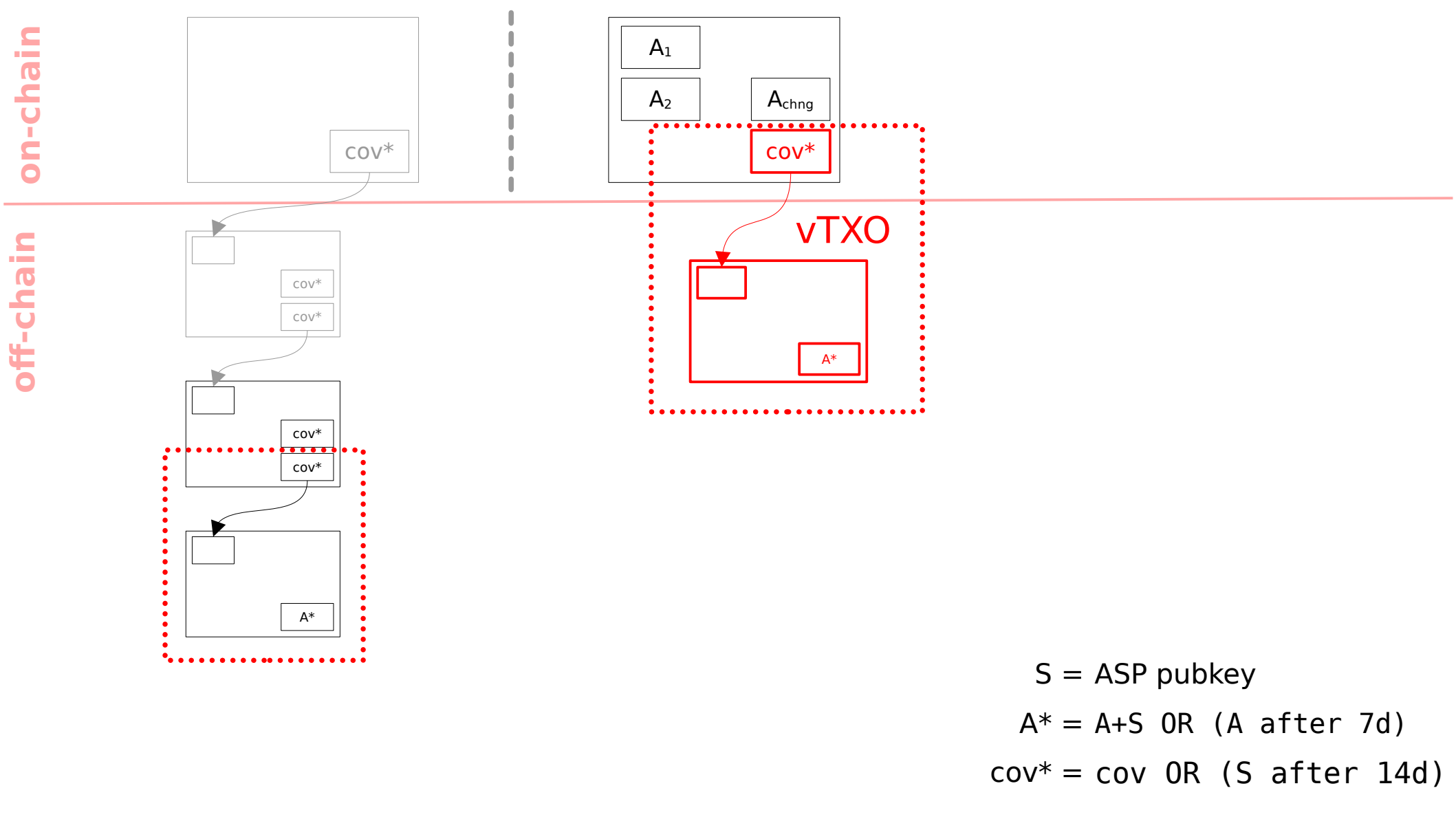
off-chain

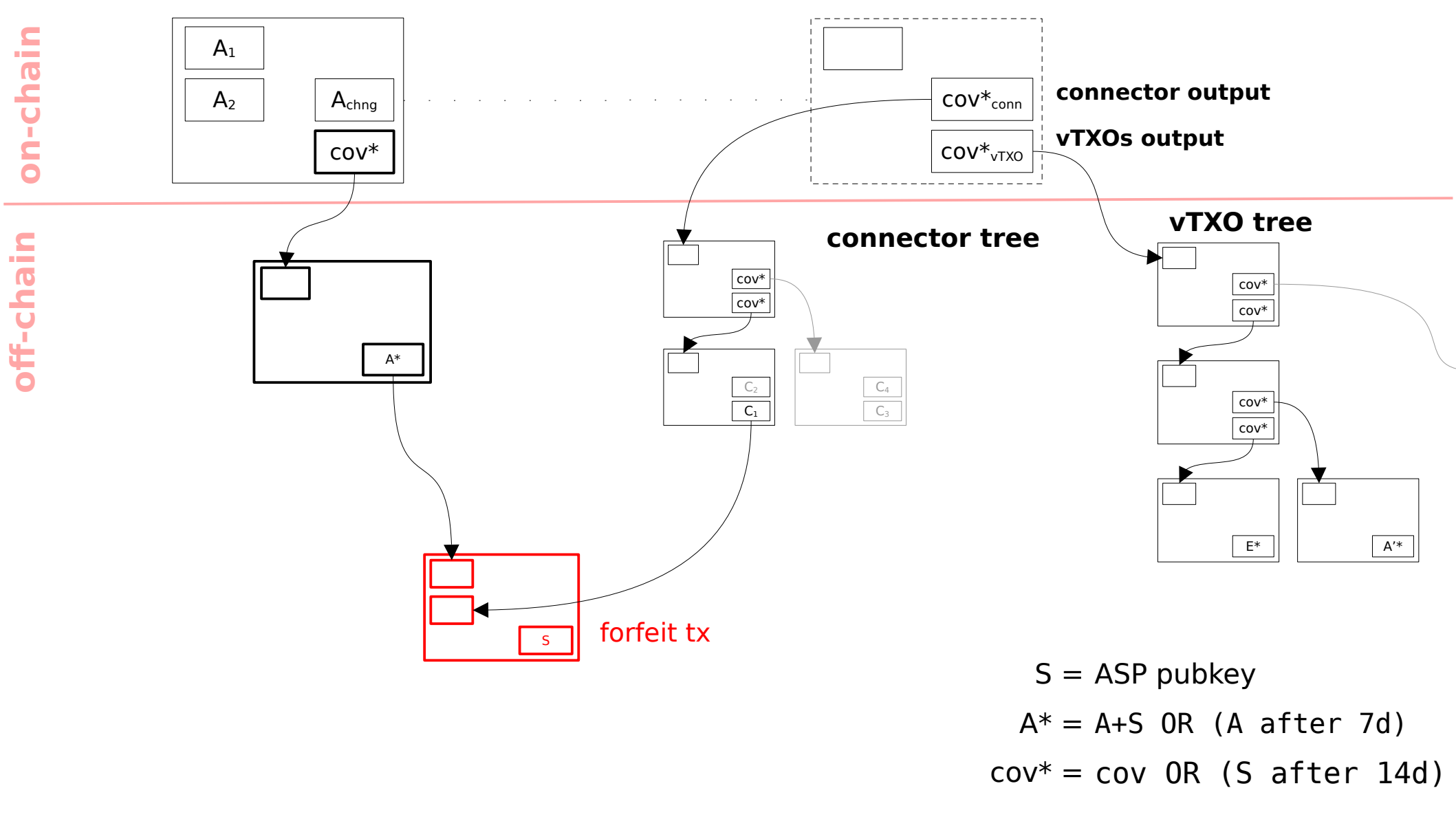


S = ASP pubkey

A* = A+S OR (A after 7d)

cov* = cov OR (S after 14d)


on-chain
off-chain
cov*
A1
A2
Achng
cov*
vTXO
A*
cov*
cov*
cov*
cov*
A*
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

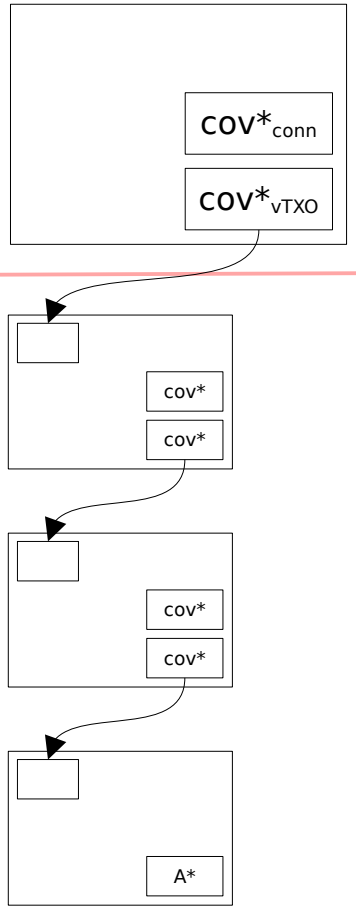

on-chain
off-chain
A1
A2
Achng
cov*
cov*conn
connector output
cov*vTXO
vTXOs output
connector tree
vTXO tree
A*
C2
C1
C4
C3
E*
A'*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

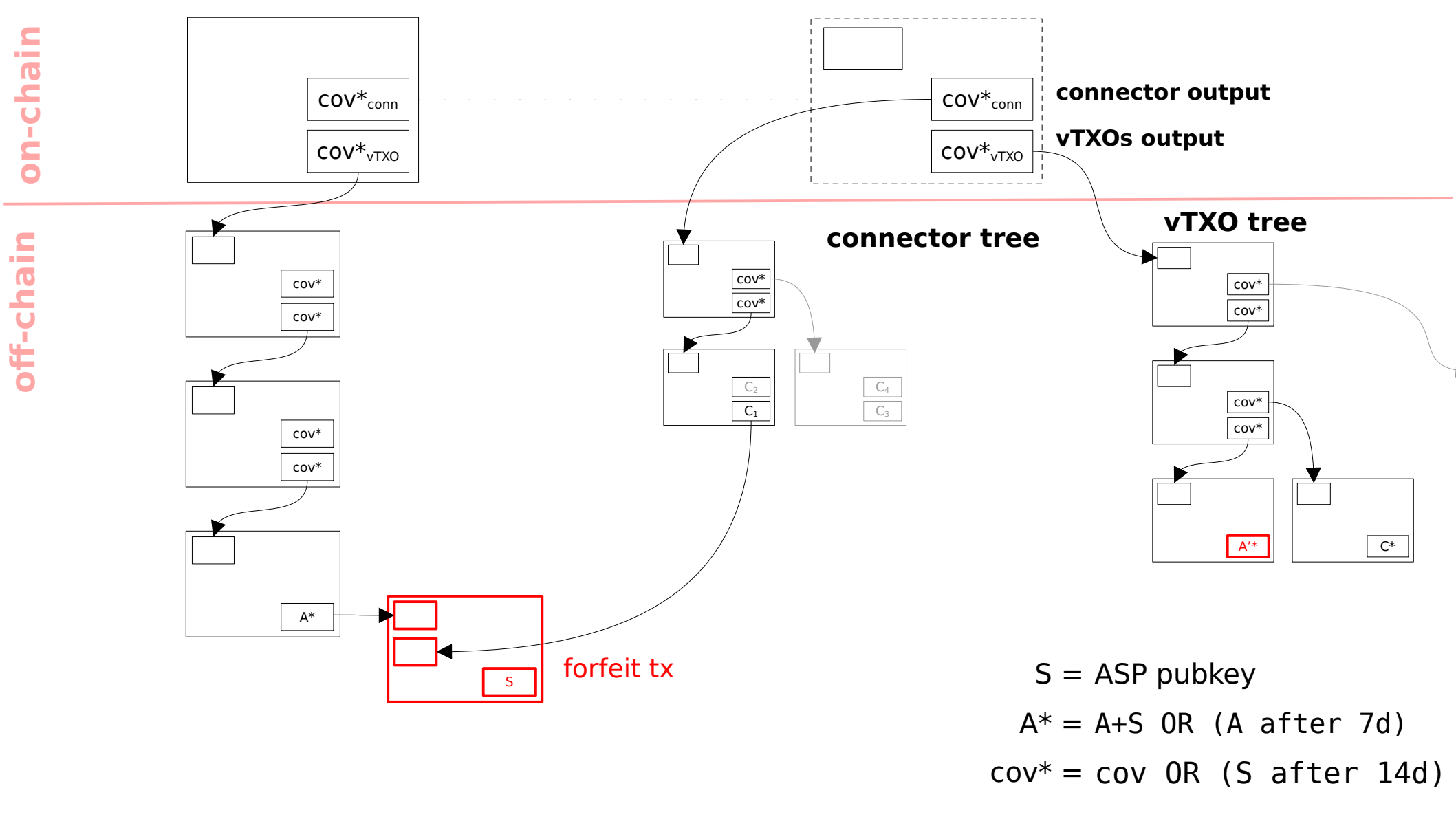
Lifting

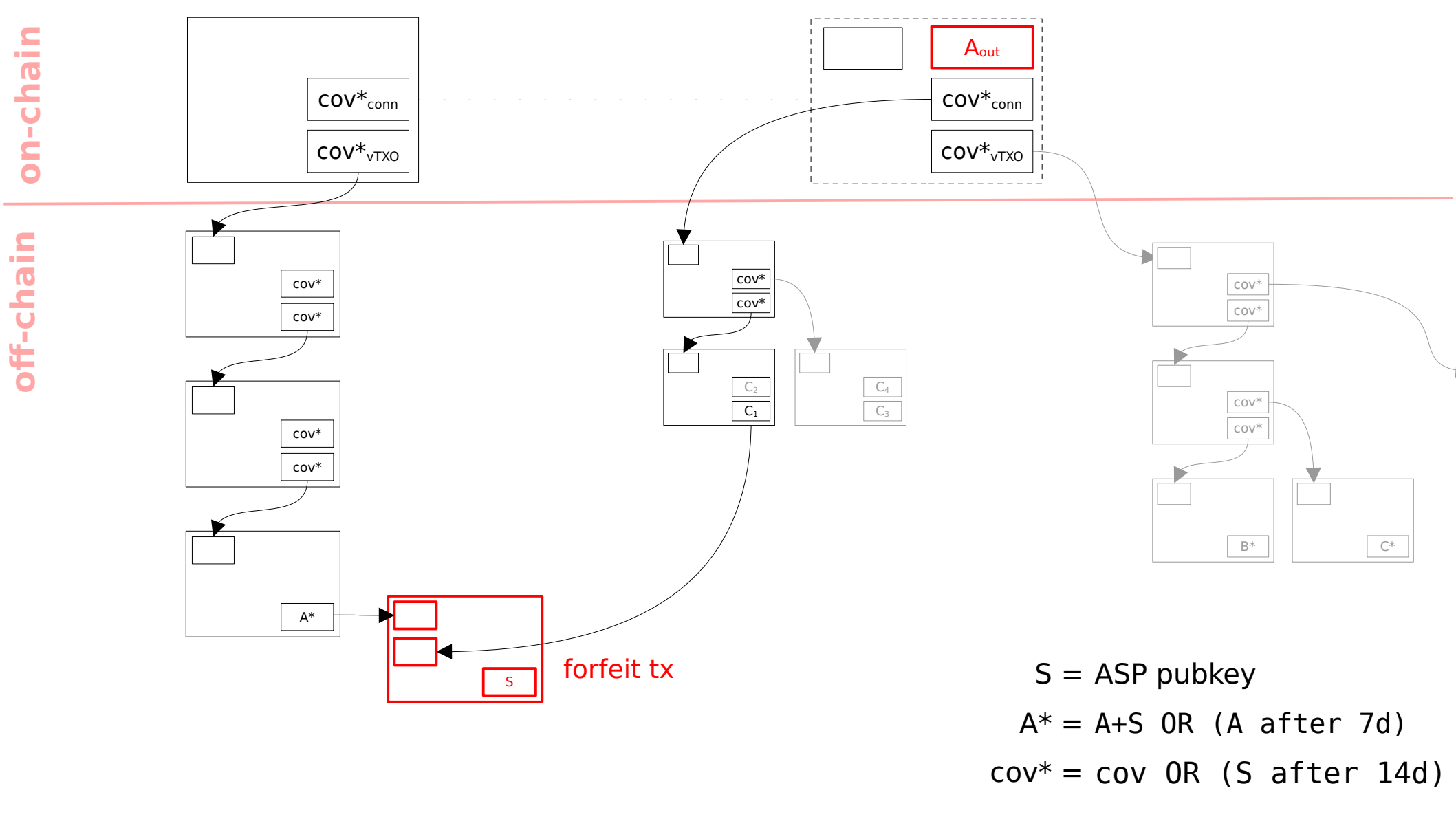
- mechanism to enter and exit the Ark
- non-interactive lift-in
  - → straight from on-chain to vTXO
- interactive lift-out
  - → non-interactive unilateral exit always available

on-chain

off-chain




on-chain
off-chain
cov*conn
cov*vTXO
cov*conn
cov*vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
cov*
cov*
cov*
cov*
cov*
cov*
cov*
cov*
cov*
C2
C1
C4
C3
A*
A'*
C*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)


on-chain
off-chain
cov*conn
cov*vTXO
Aout
cov*conn
cov*vTXO
cov*
cov*
cov*
cov*
A*
C2
C1
C4
C3
cov*
cov*
cov*
cov*
B*
C*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

Note on covenants

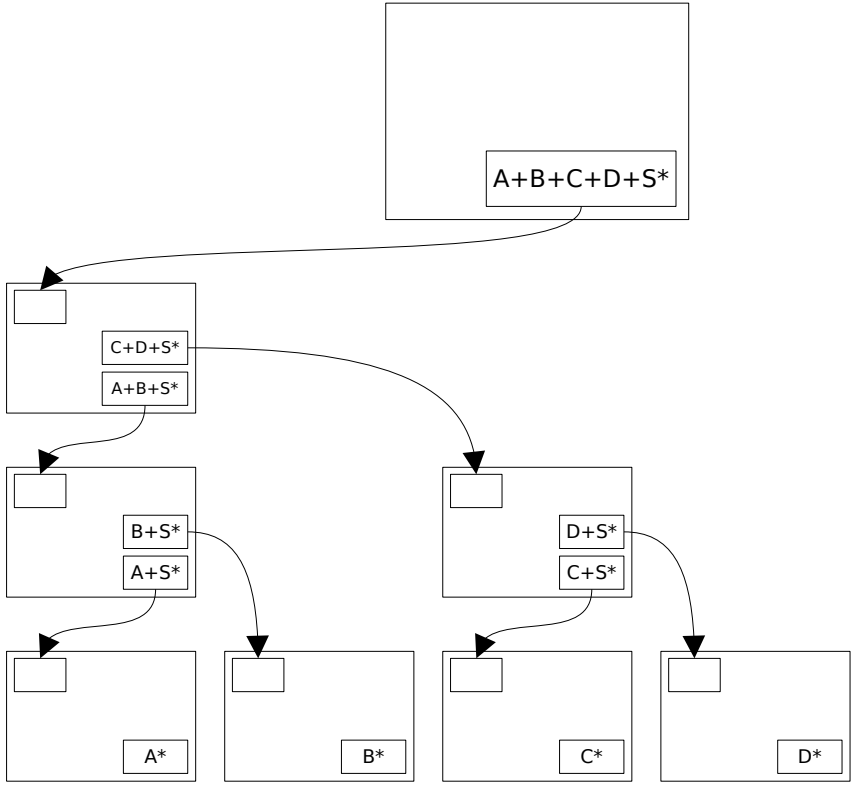
- covenant construction desired
  - OP_CHECKTEMPLATEVERIFY (CTV)
  - OP_TXHASH / OP_TX
  - OP_CHECKSIGFROMSTACK (on Liquid)
  - SIGHASH_ANYPREVOUT* (aka SIGHASH_NOINPUT)
- possible on Inquisition testnet or Liquid right now

Without covenants: clArk

- use multisigs instead of covenants

Without covenants: clArk

- use multisigs instead of covenants
  - → all receivers cosign with ASP
  - → requires receivers online



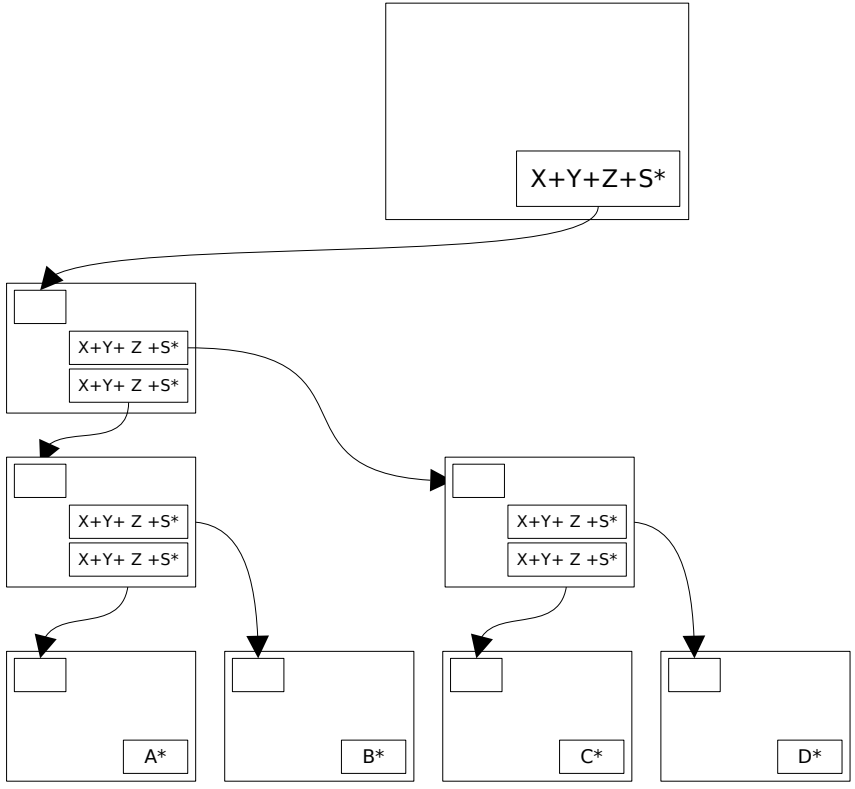
S = ASP pubkey

A+B+S* = A+B+S OR (S after 14d)

A* = A+S OR (A after 7d)

Without covenants: clArk

- use multisigs instead of covenants
  - → all receivers cosign with ASP
  - → requires receivers online
  - → (how about all senders?)
- possible on Bitcoin today



S = ASP pubkey

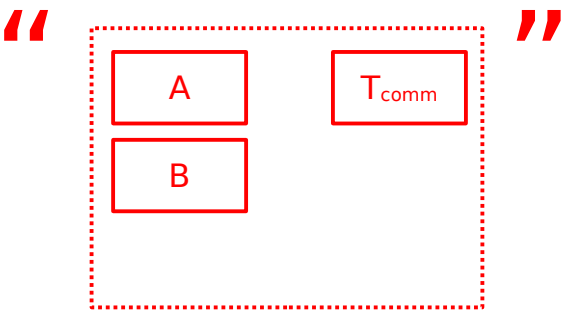
A+B+S* = A+B+S OR (S after 14d)

A* = A+S OR (A after 7d)

You said Lightning?

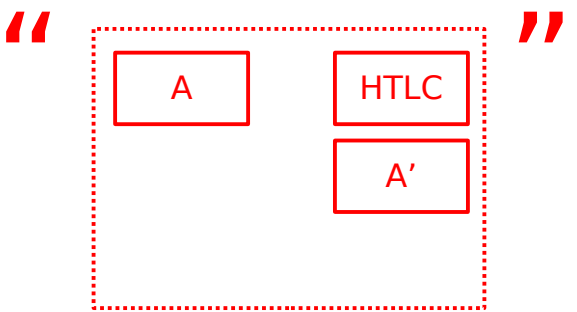
- Lightning channel commitment vTXOs
  - → Ark as “channel factory”
  - → requires periodic channel refresh

$T_{comm} = A+B$



You said Lightning?

- Lightning channel commitment vTXOs
- HTLCs can be added directly as vTXOs
  - → Lightning payment in Ark round
  - → ASP acts as LSP and forwards payment




```
HTLC = (S AND preimage) OR (A after 24h)
```

Addressing

- `A* = A+S OR (A after 7d)`
- in theory many Miniscript policies seem possible
  → `or(and(P, key(S)), after(7d, P))`
- ideally a single Schnorr (FROST?) pubkey
  → optimal for taproot keyspend
- receiver gives a pubkey/policy to sender

Privacy

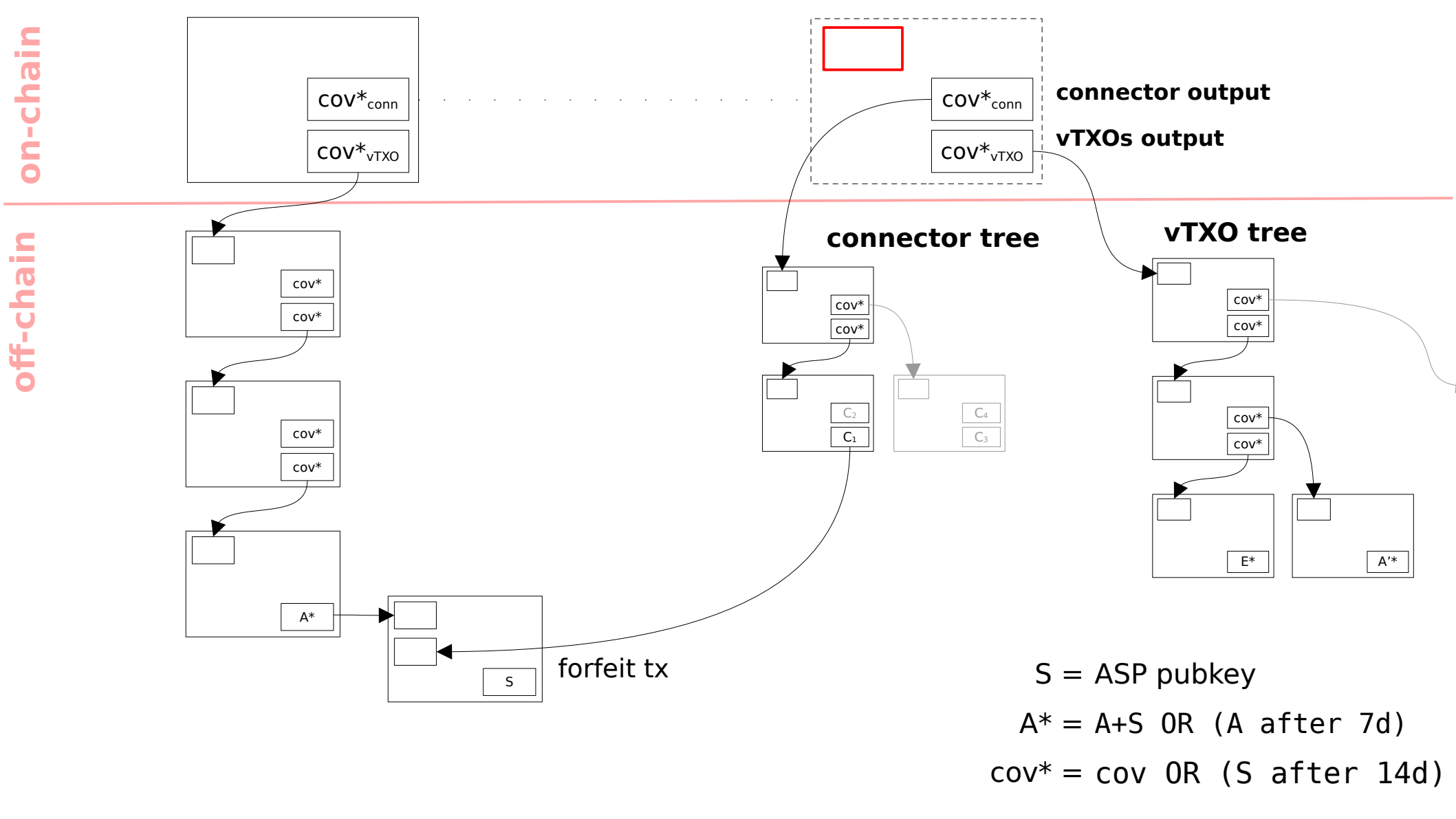
- currently the ASP has full insight in txs
- solution: blinded coinjoins “à la WabiSabi”
  - → spenders get blinded tokens for input vTXOs
  - → redeem blinded tokens for output vTXOs
  - → fixed denominations for vTXO values

But wait...

- vTXO output pubkey reuse is a problem
  - → ASP can deanonymise receivers & targetted senders
- need new pubkey each vTXO & round attempt
  - → paynym/greenaddress-like solution
    - out-of-band from sender to receiver? nostr?
    - using deterministic round-specific entropy?

Existing challenges

- ASP can double spend txs in mempool


on-chain
off-chain
cov*conn
cov*vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
C2
C1
C4
C3
E*
A'*
A*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

Existing challenges

- ASP can double spend txs in mempool
  - → no inherent incentive
  - → disincentive because of HTLCs
    - ↙ LN-on-Ark txs don’t care about confirmations
  - → double-spend prevention with bond?

Existing challenges

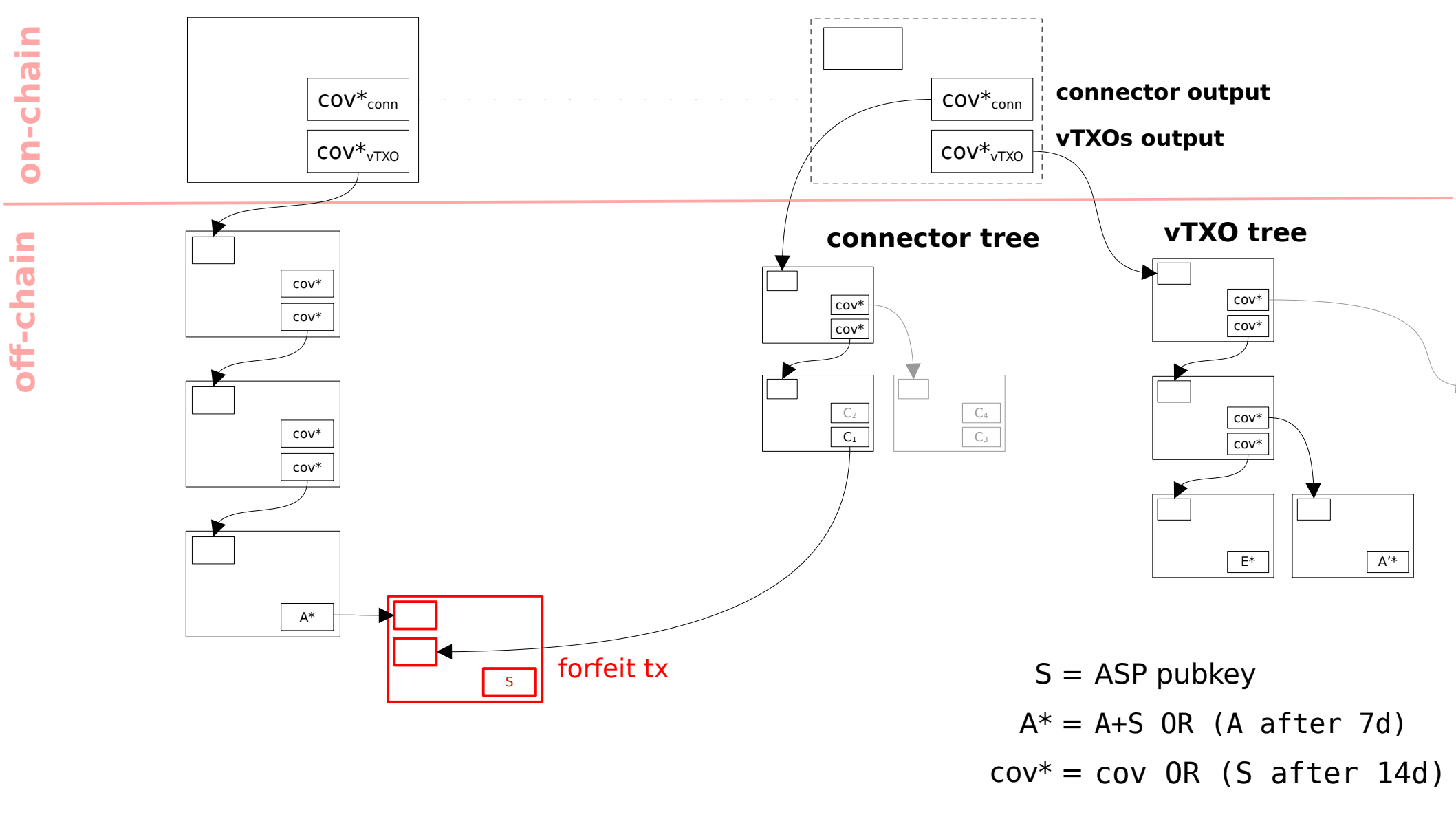
- ASP can double spend txs in mempool
- high liquidity requirement
  - → increases with vTXO velocity
  - → depends on vTXO expiration parameter
  - → ASP can charge fees based on vTXO age

Existing challenges

- ASP can double spend txs in mempool
- high liquidity requirement
- DoS by forcing many round restarts
  - → penalties for abandoning a round
  - → attack incentive is small with larger round times

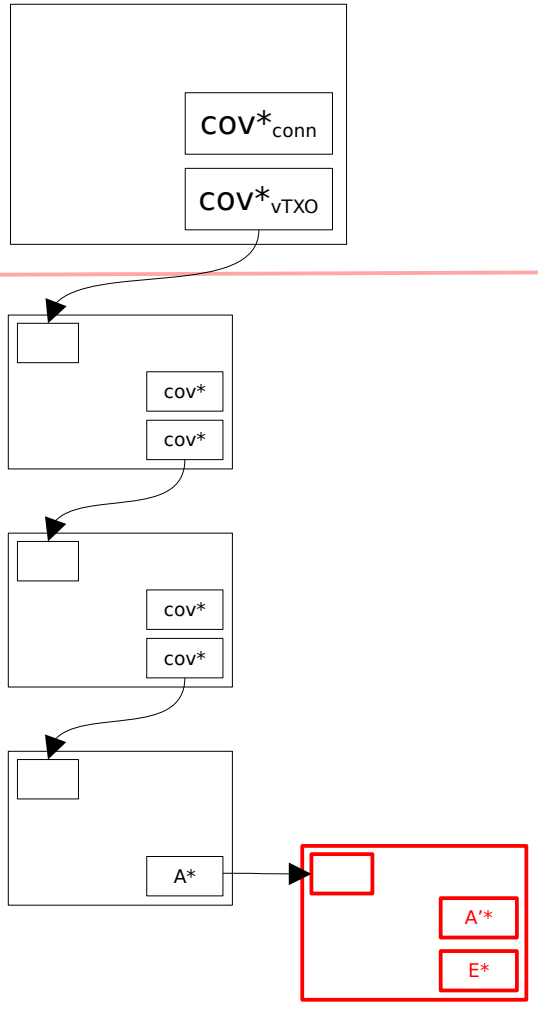
NEW: the Somsen Shortcut

- send vTXOs outside Ark round


on-chain
off-chain
cov*conn
cov*vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
C2
C1
C4
C3
E*
A'*
A*
S
forfeit tx
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

on-chain

off-chain



shortcut tx

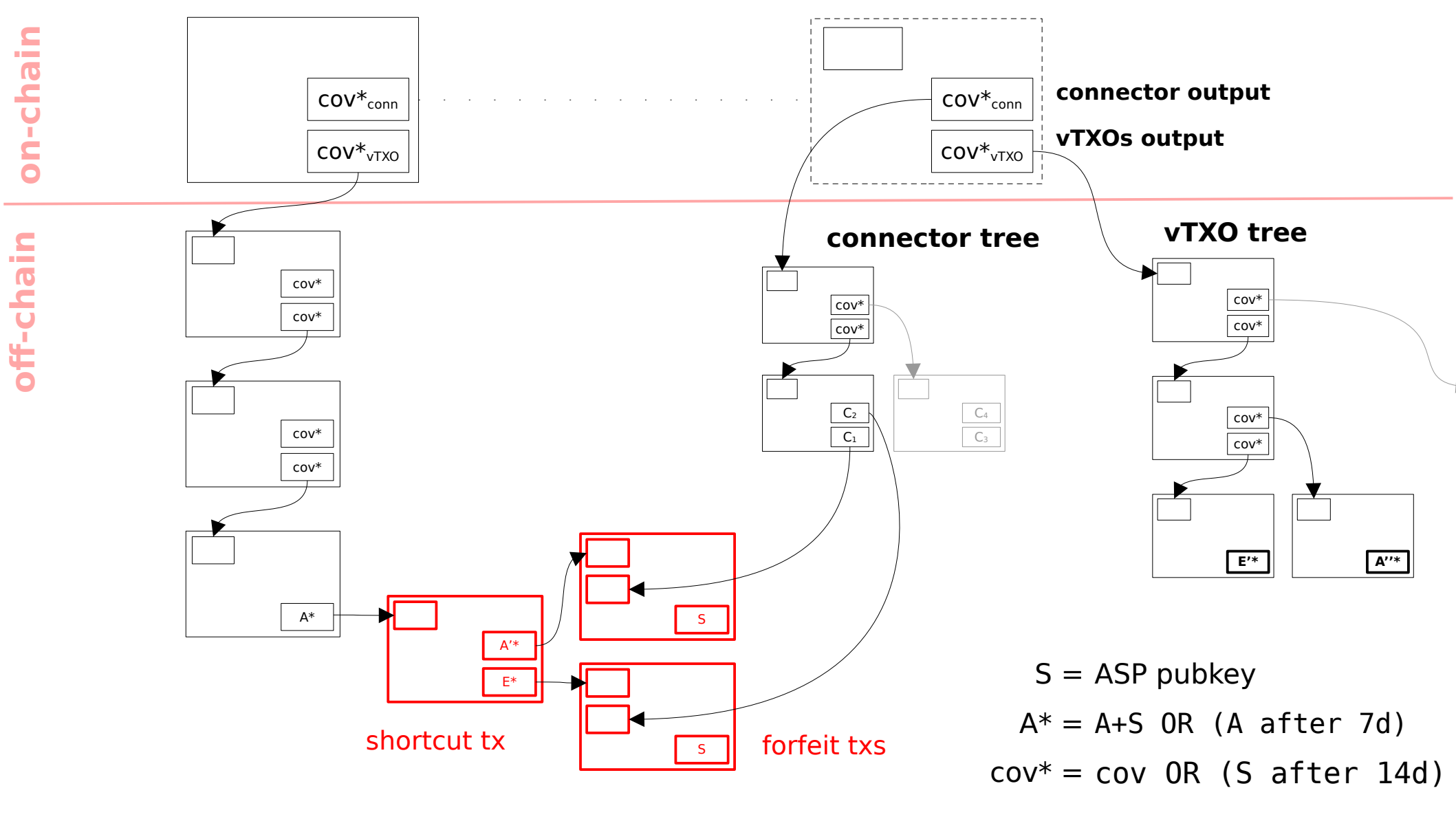
S = ASP pubkey

A* = A+S OR (A after 7d)

cov* = cov OR (S after 14d)

NEW: the Somsen Shortcut

- send vTXOs outside Ark round
  - → “building a state-chain from a vTXO”


on-chain
off-chain
cov*conn
cov*vTXO
connector output
vTXOs output
connector tree
vTXO tree
cov*
A*
C2
C1
C4
C3
E'*
A''*
A'*
E*
S
shortcut tx
forfeit txs
S = ASP pubkey
A* = A+S OR (A after 7d)
cov* = cov OR (S after 14d)

NEW: the Somsen Shortcut

- send vTXOs outside Ark round
  → “building a state-chain from a vTXO”
- makes clArk more feasible

Thanks

- https://roose.io/presentations
- https://arkpill.me/
- Questions?