how to build a private dex

henry de valence // penumbra @ devcon vi, bogota // 12 october 2022

penumbra is {

penumbra is {

- private proof-of-stake L1

penumbra is {

private proof-of-stake L1

interchain shielded pool

penumbra is

- private proof-of-stake L1
- interchain shielded pool
- private dex

why build a private dex?

why build a private dex?

because every market is a market in information

why build a private dex?

because every market is a market in information

…so information leaks are value leaks

why build a private dex?

because every market is a market in information

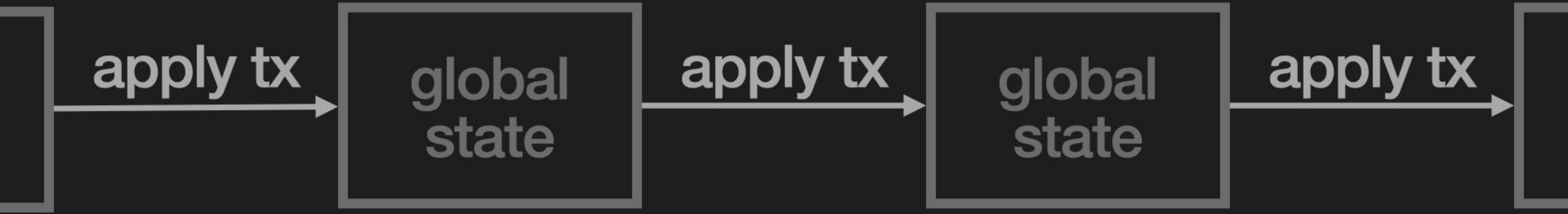
…so information leaks are value leaks

…so privacy unlocks capital efficiency

first challenge: state model

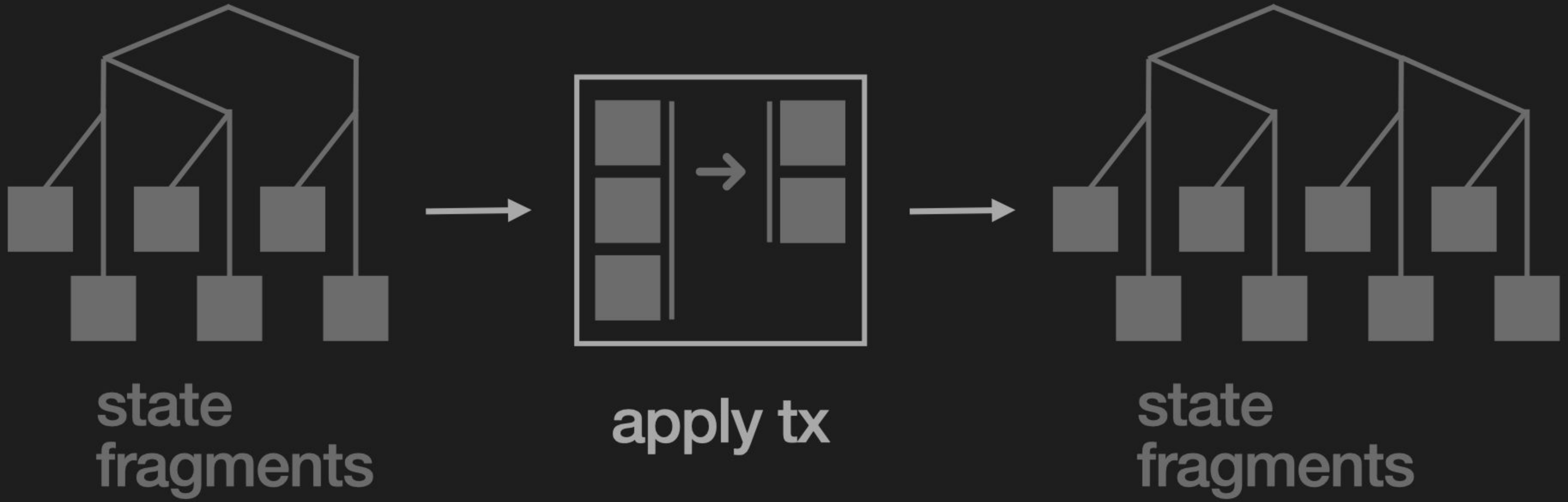
transparent blockchains use

mutable state



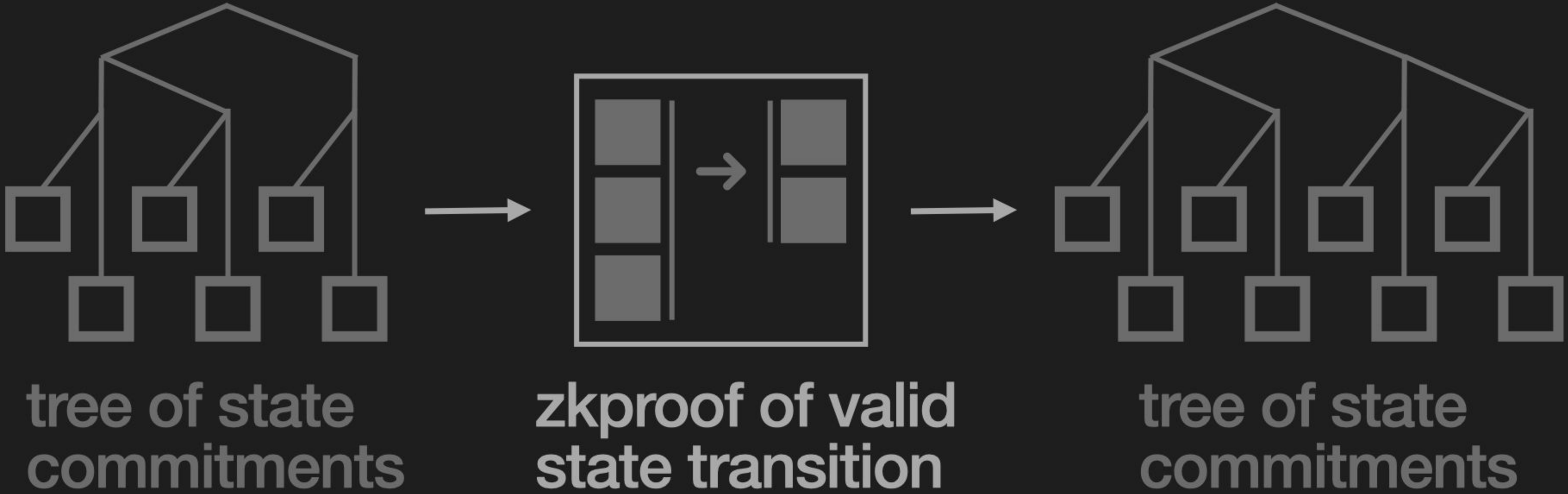
shielded blockchains need

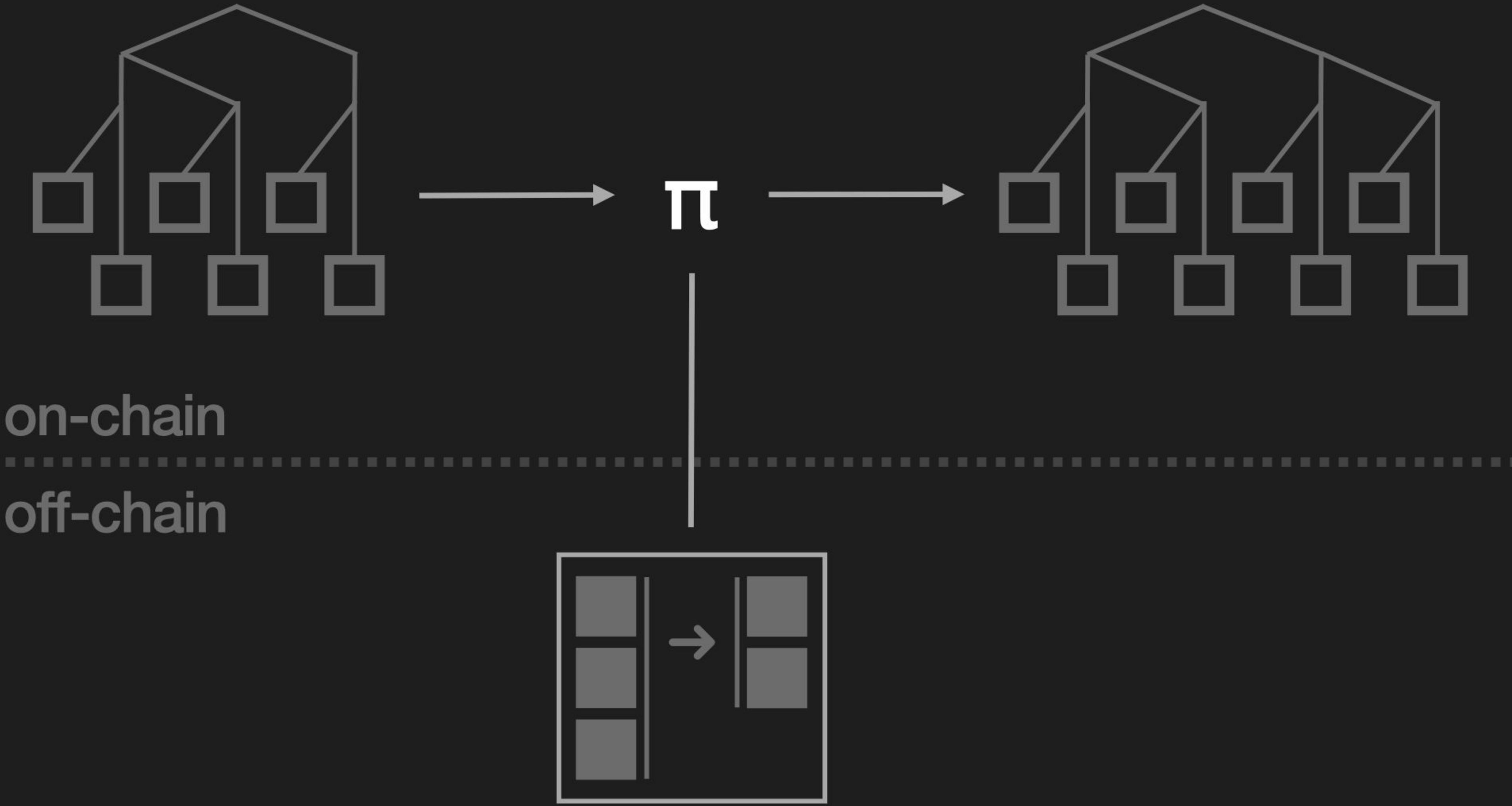
composable state



...so that **state transitions**

can be **private**

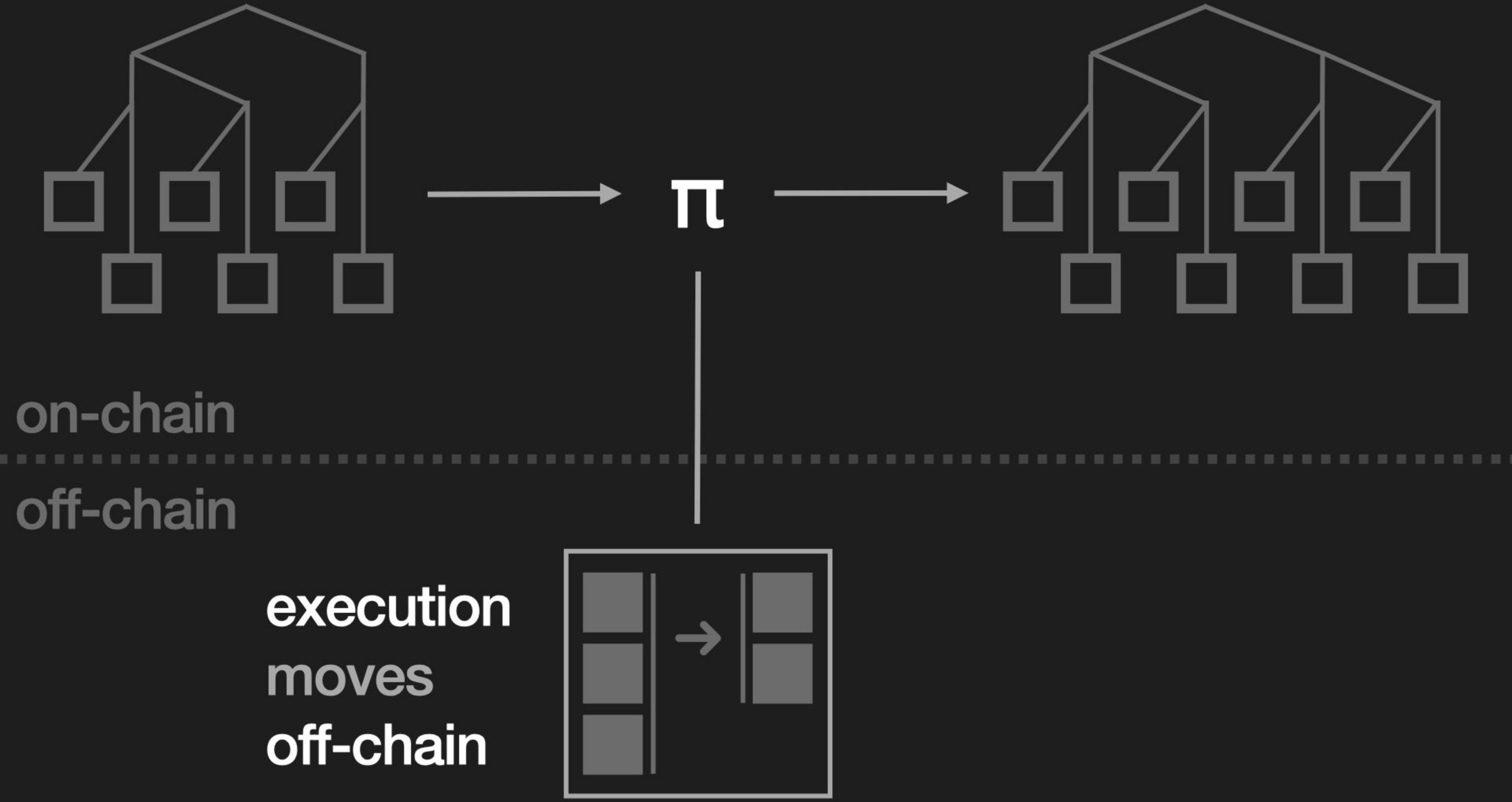

tree of state commitments
zkproof of valid state transition
tree of state commitments



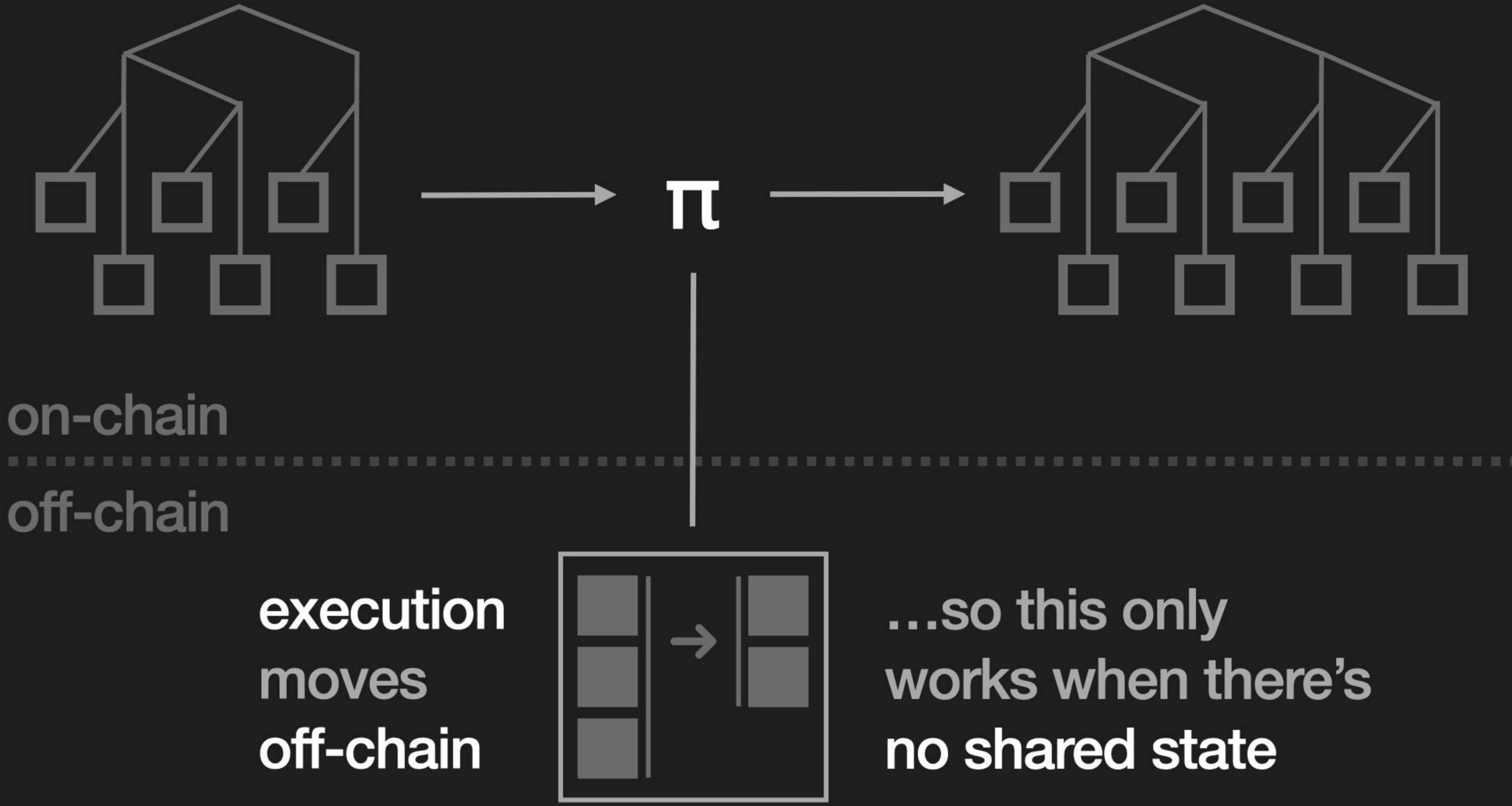
π

on-chain

off-chain



execution
moves
off-chain



on-chain

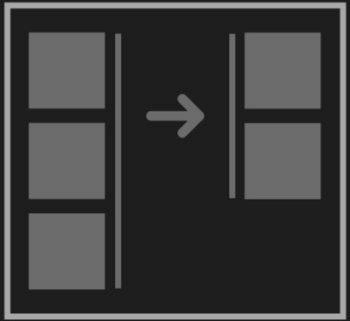
off-chain

execution moves **off-chain**

…so this only works when there’s **no shared state**

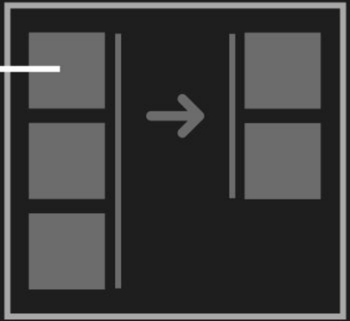
how do we recover late binding?

how do we recover late binding?



what we have:
early binding

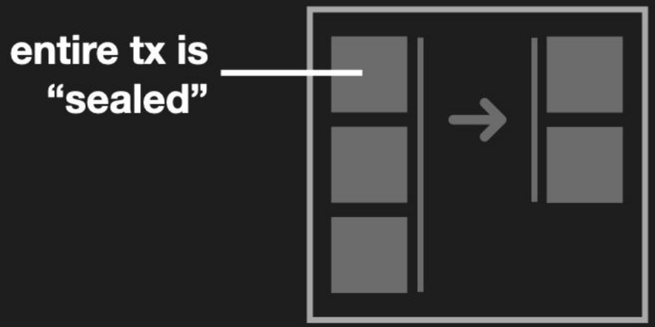
how do we recover late binding?



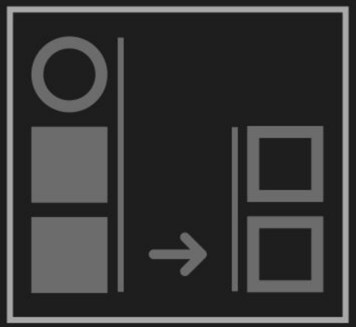
entire tx is "sealed"

what we have:
early binding

how do we recover late binding?



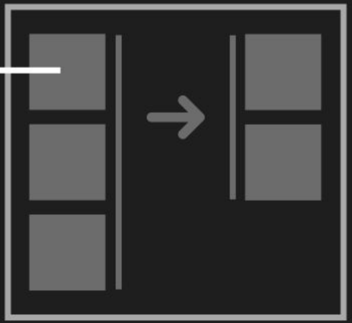
what we have:
early binding



what we want:
late binding

how do we recover **late binding?**

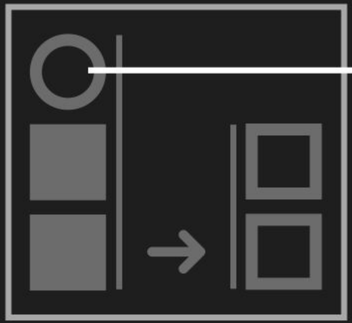
entire tx is
“sealed”



what we have:

early binding

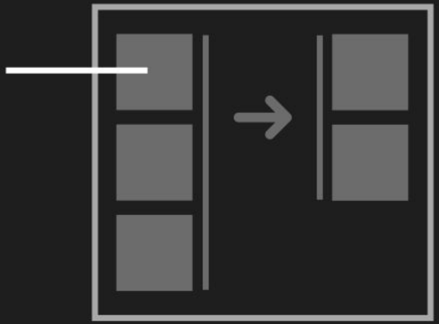
when tx is
executed,
**shared state
goes here**



what we want:

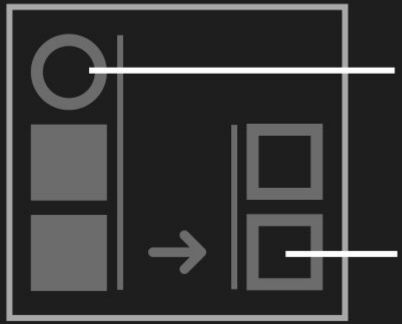
late binding

how do we recover late binding?



entire tx is
“sealed”

what we have:
early binding



when tx is
executed,
shared state
goes here

and determines
these outputs

what we want:
late binding

we need a better

concurrency model

for **shared state**

what if we model concurrency
with message passing
instead of locking?

an actor model for blockchains

an actor model for blockchains

transactions pass messages to contracts

an actor model for blockchains

transactions pass messages to contracts

each contract executes once per block, on all messages, allowing batch processing

an actor model for blockchains

transactions pass messages to contracts

each contract executes once per block, on all messages, allowing batch processing

user state executes async, off-chain, in zk

an actor model for blockchains

transactions pass messages to contracts

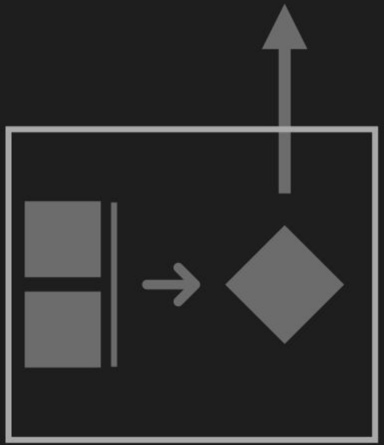
each contract executes once per block, on all messages, allowing batch processing

user state executes async, off-chain, in zk

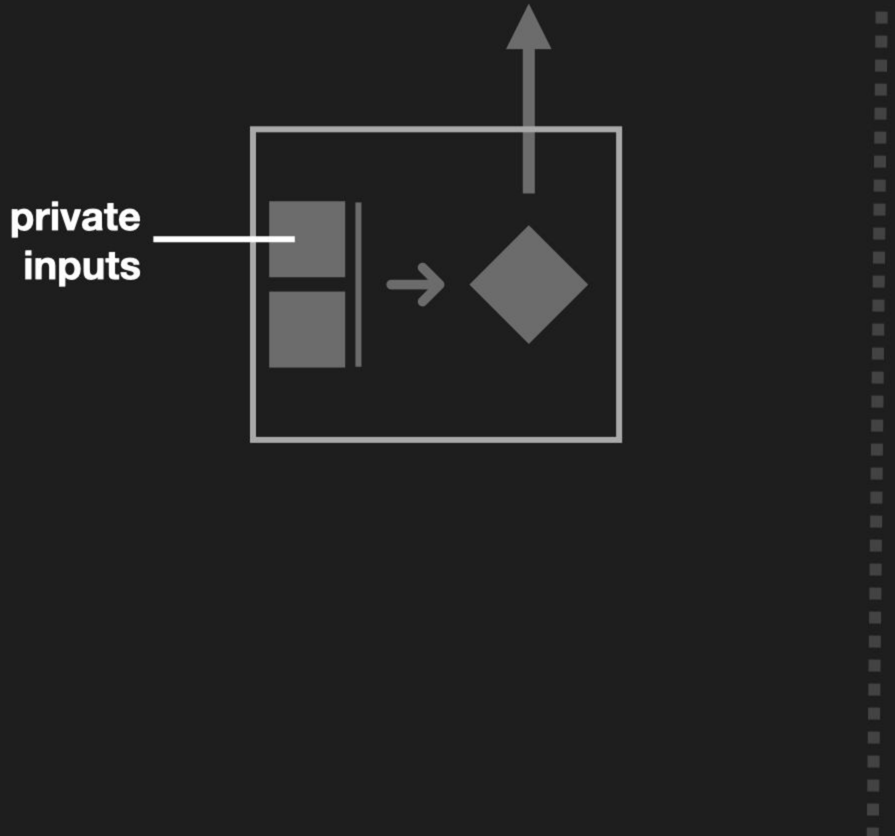
unlocks scalability *and* privacy!

async zk execution via message passing

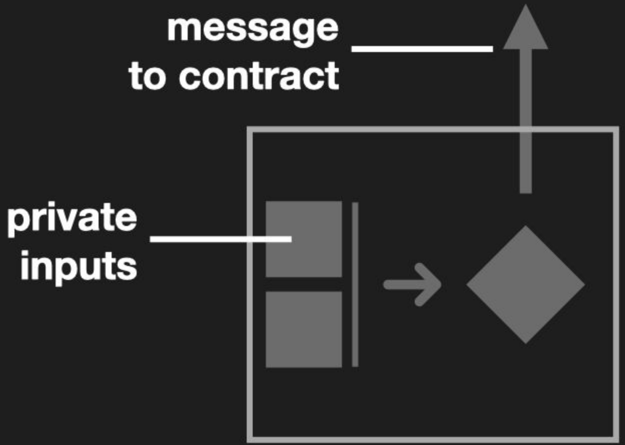
async zk execution via message passing



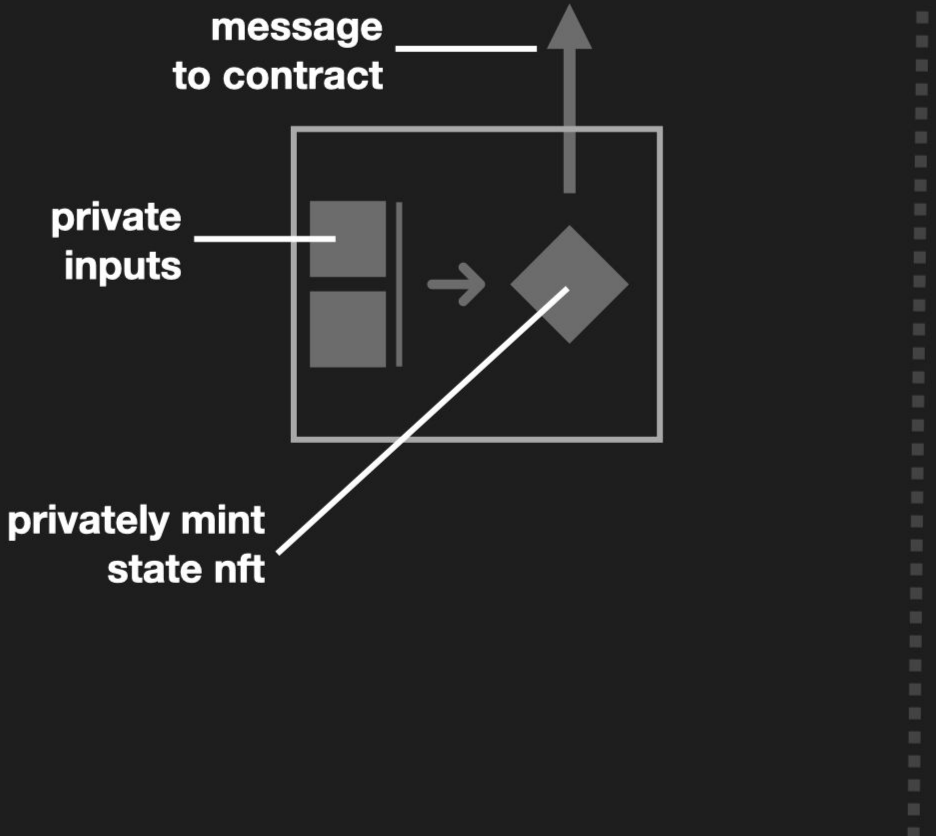
async zk execution via message passing



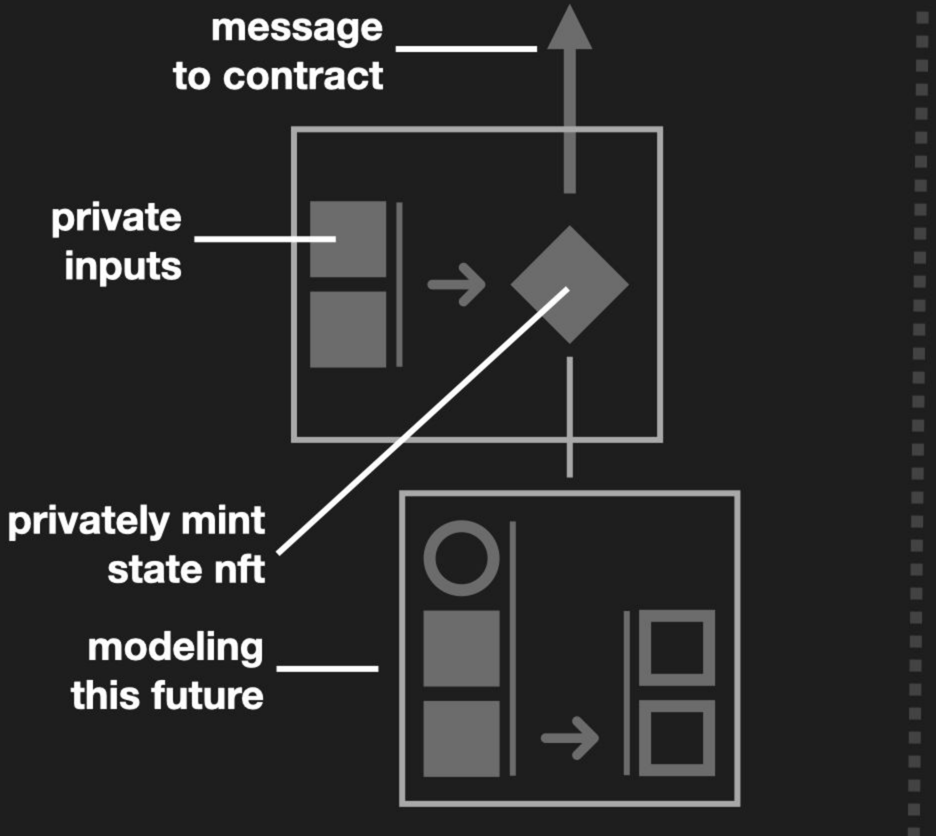
async zk execution via message passing



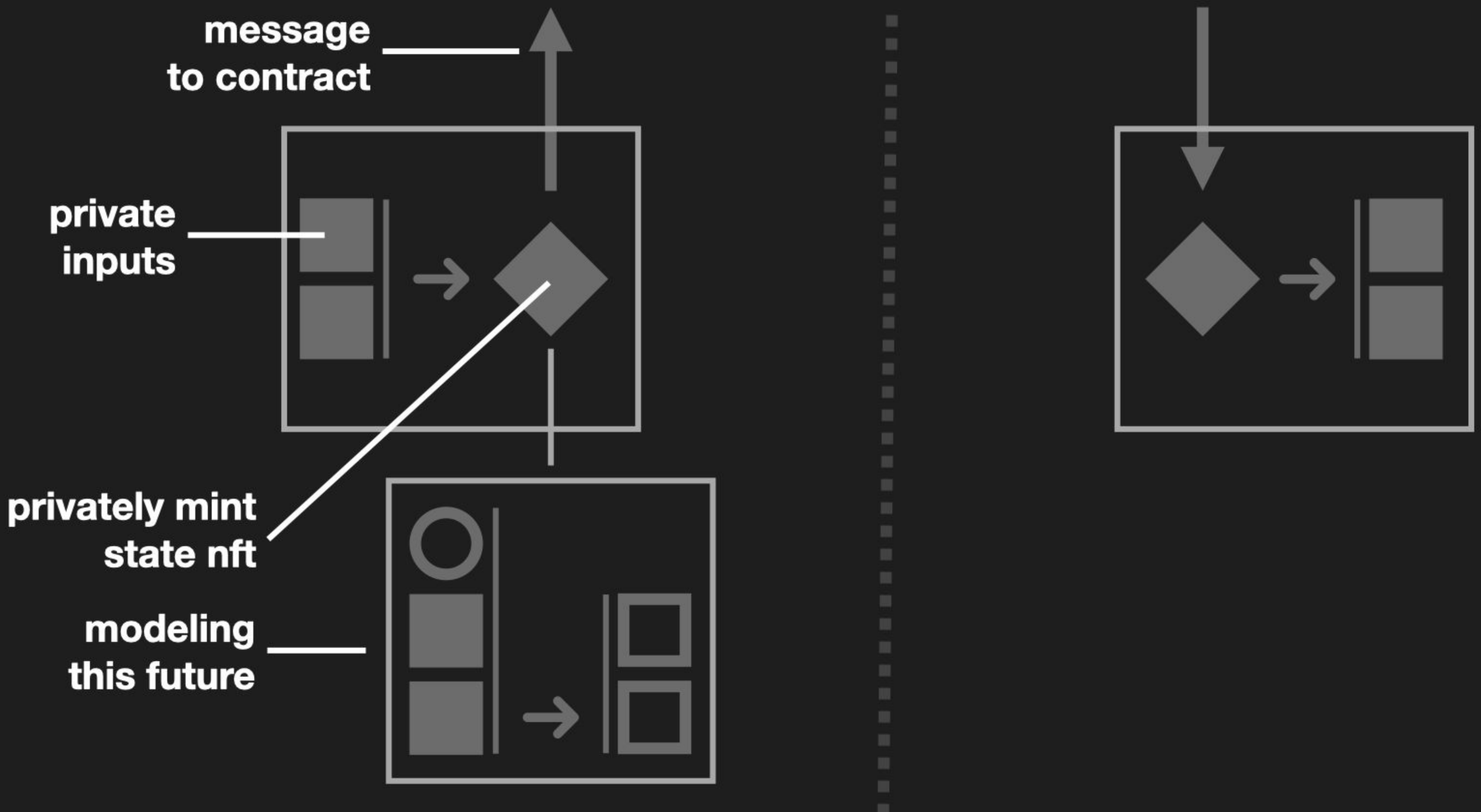
async zk execution via message passing



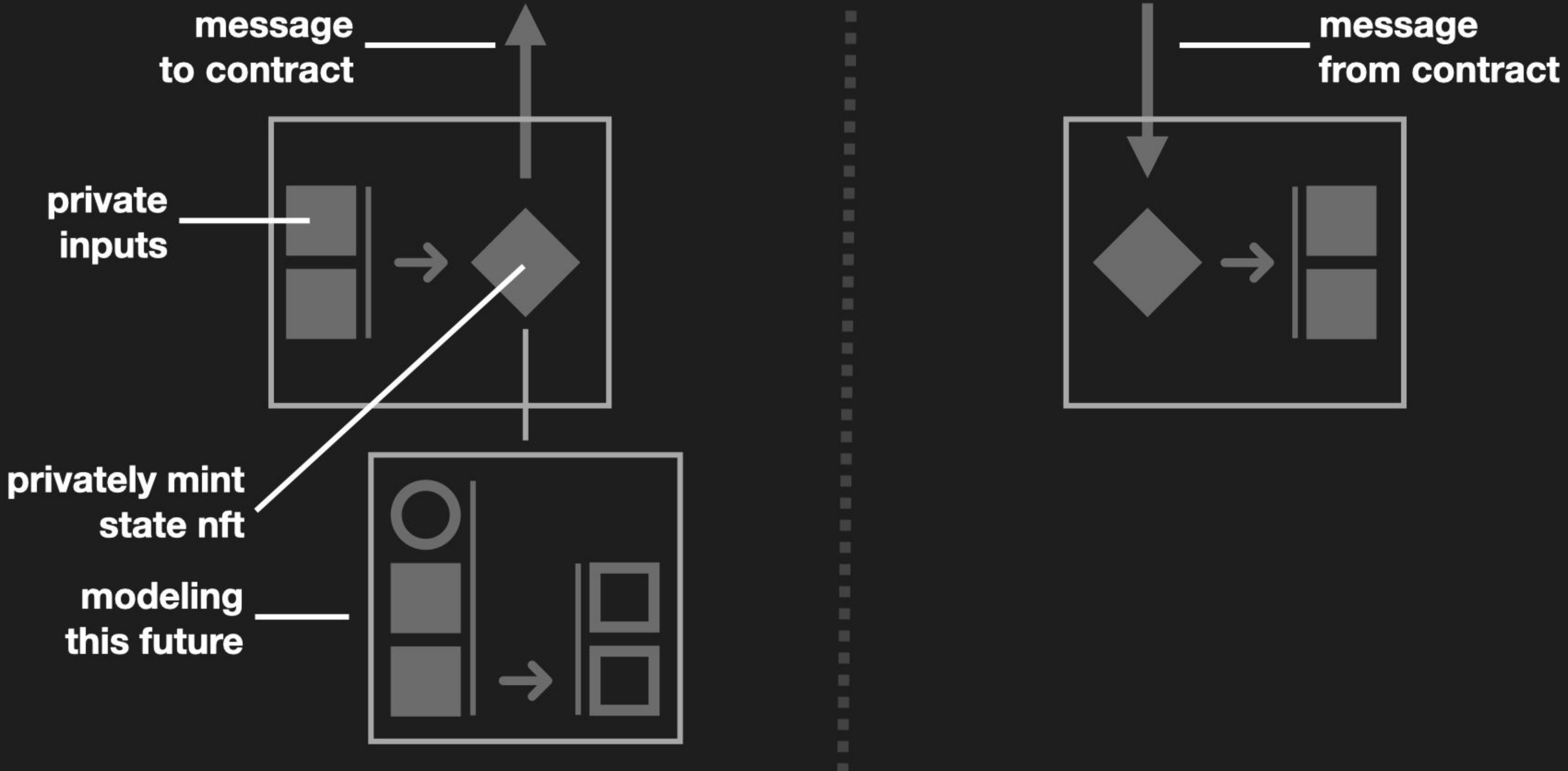
async zk execution via message passing



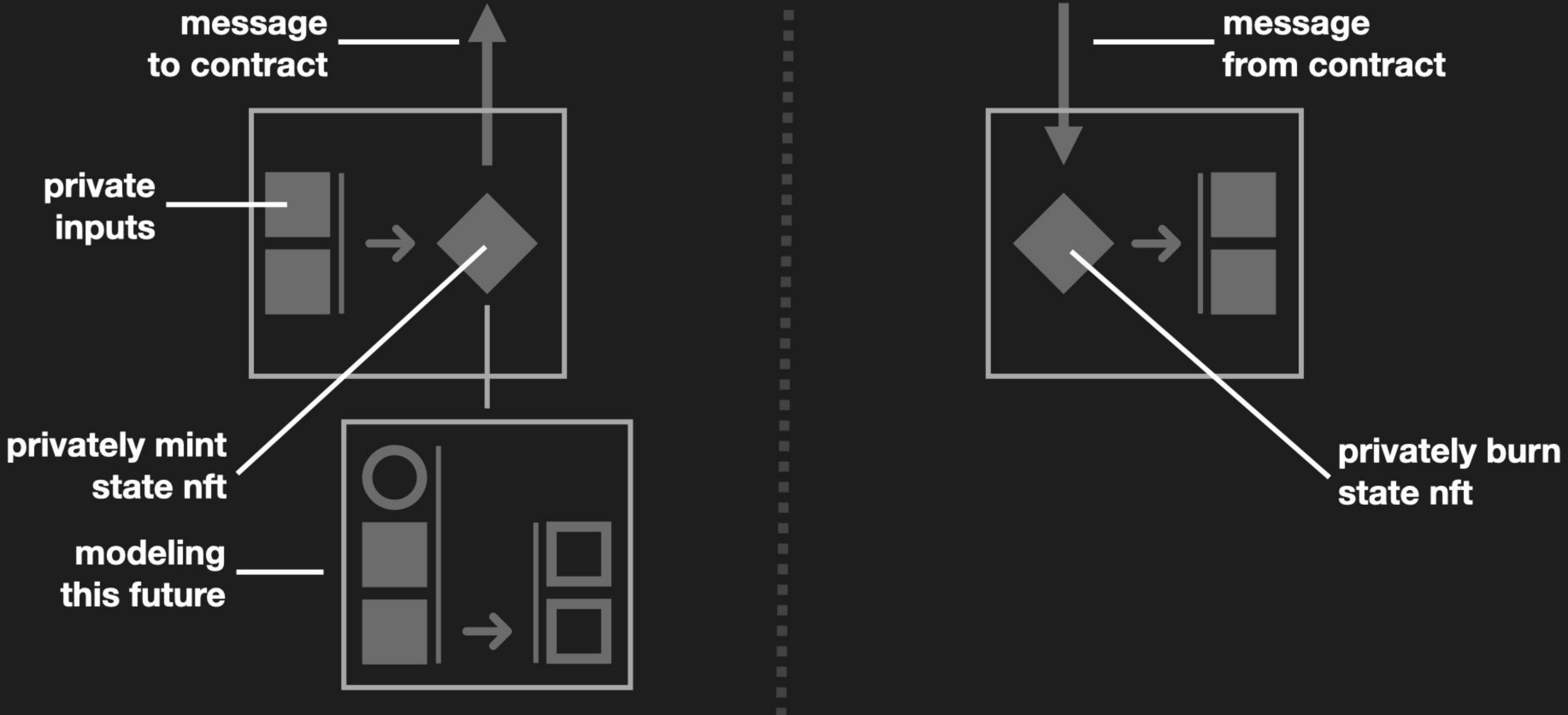
async zk execution via message passing



async zk execution via message passing



async zk execution via message passing



message to contract

private inputs

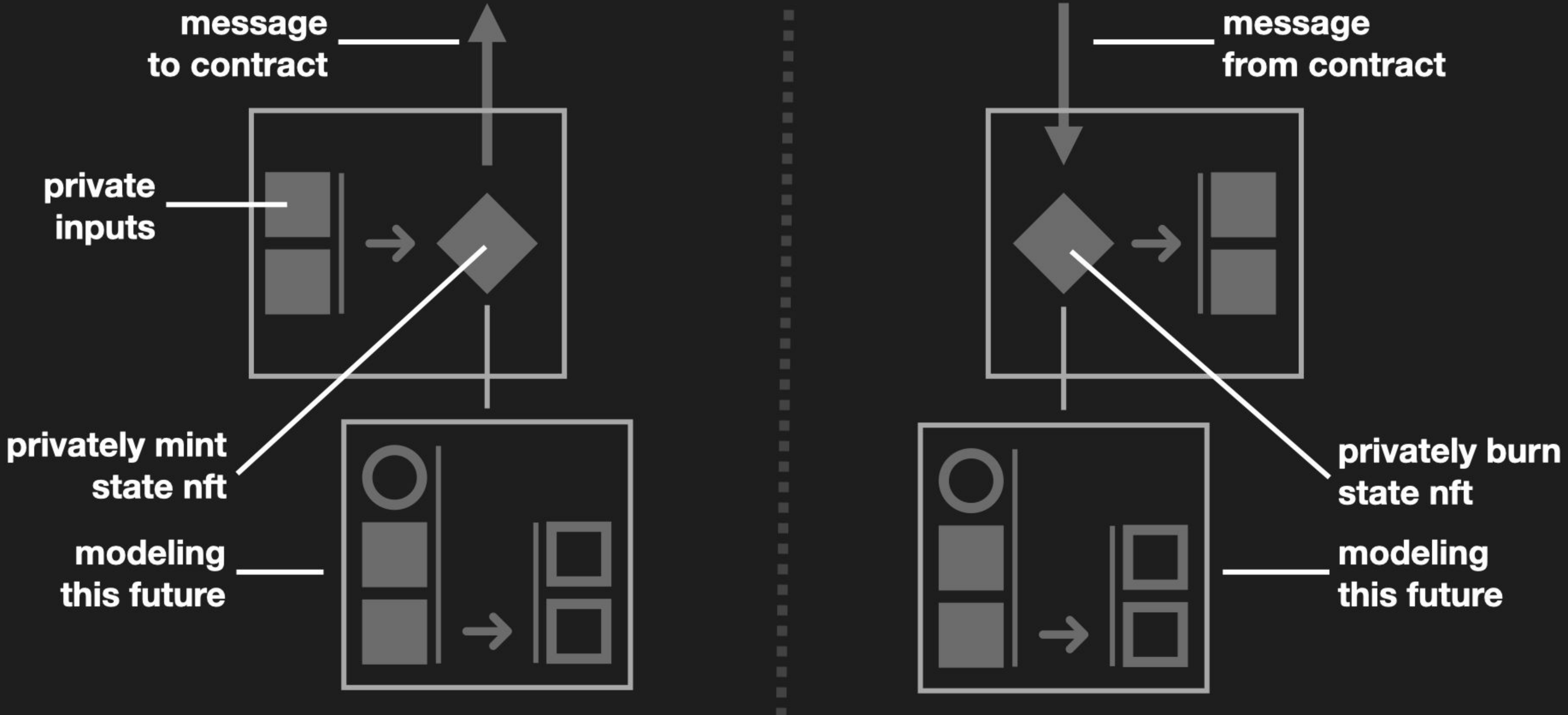
privately mint state nft

modeling this future

message from contract

privately burn state nft

async zk execution via message passing



message to contract

private inputs

privately mint state nft

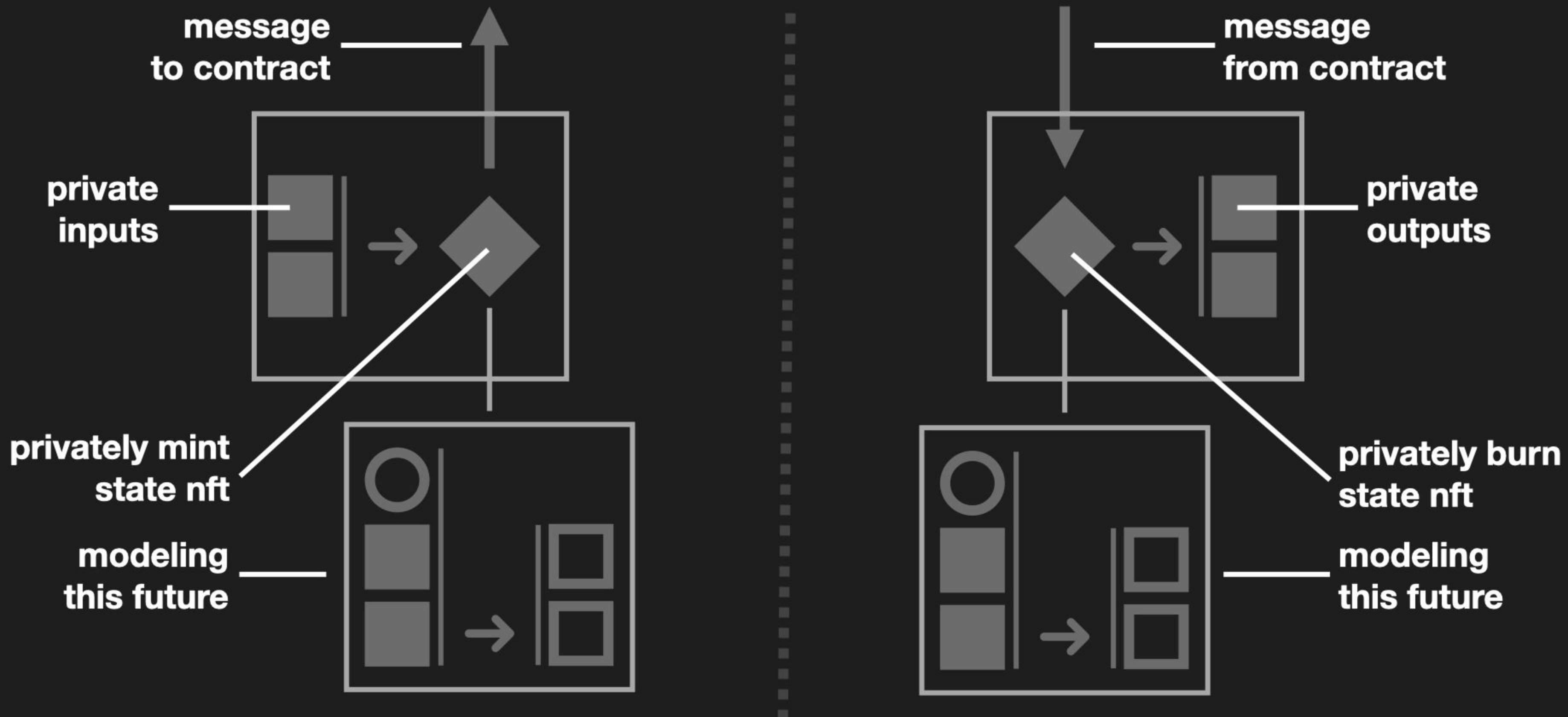
modeling this future

message from contract

privately burn state nft

modeling this future

async zk execution via message passing



second challenge: privacy model

useful blockchains

revolve around

public shared state

how do we allow

private interaction

with public shared state?

two basic strategies:

two basic strategies:

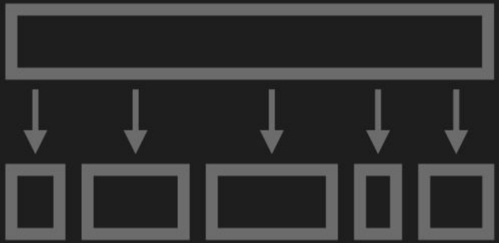
splitting flows

two basic strategies:

splitting flows

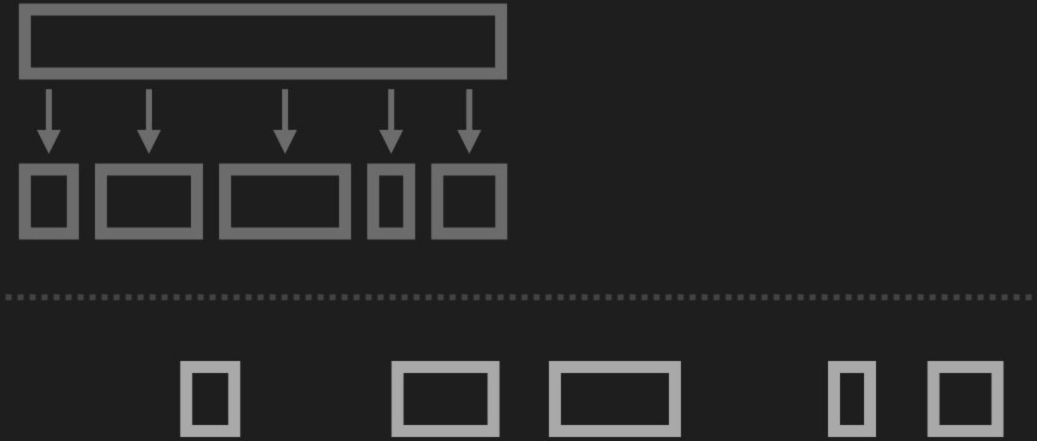
batching flows

splitting flows



splitting flows

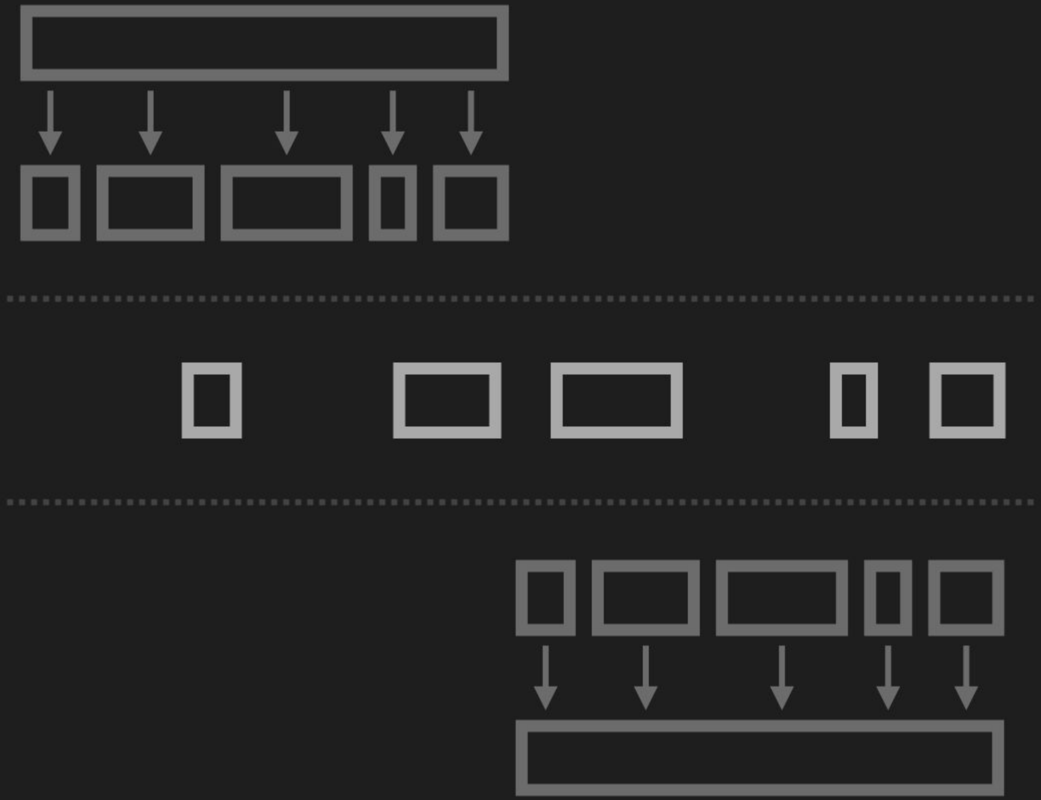
split value into randomized sub-amounts



splitting flows

split value into randomized sub-amounts

reveal in distinct transactions



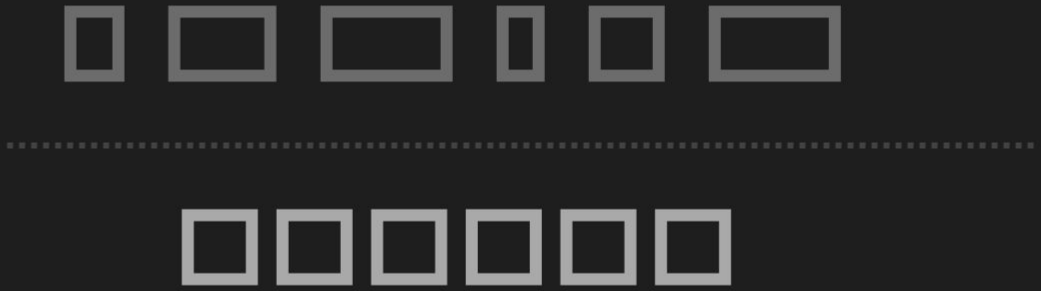
splitting flows

split value into randomized sub-amounts

reveal in distinct transactions

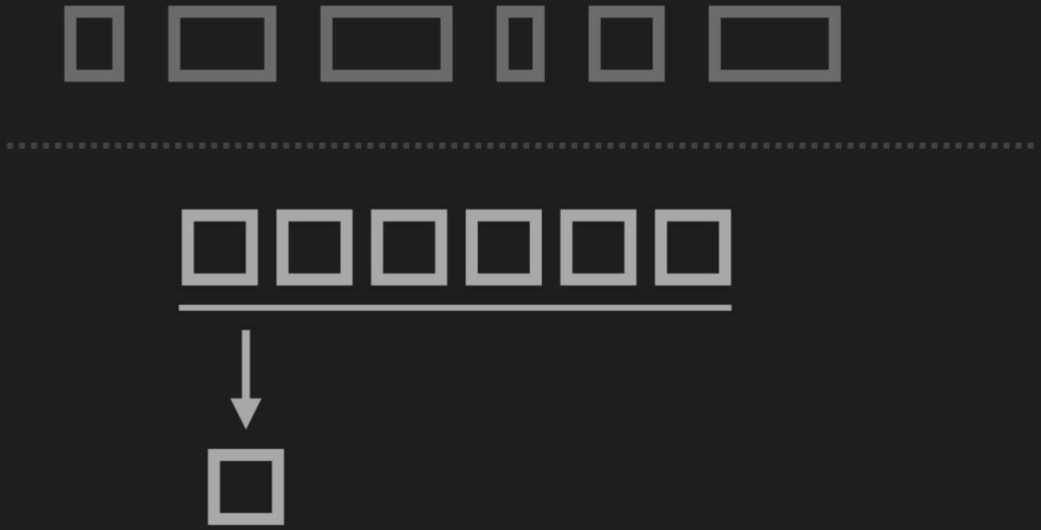
only works with shielded base layer

batching flows



batching flows

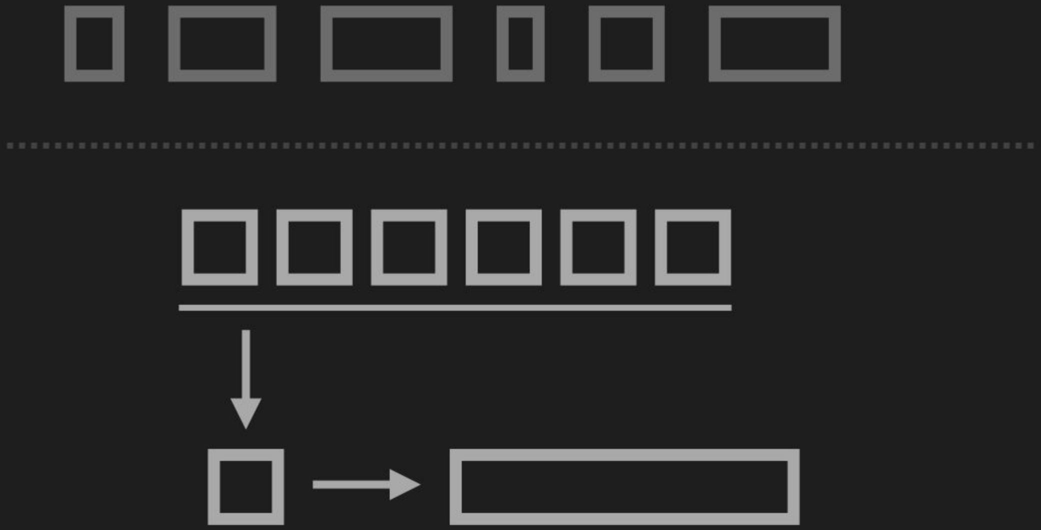
users encrypt
integer amounts
with **flow encryption**



batching flows

users encrypt integer amounts with **flow encryption**

validators sum encryptions and **decrypt batch total**

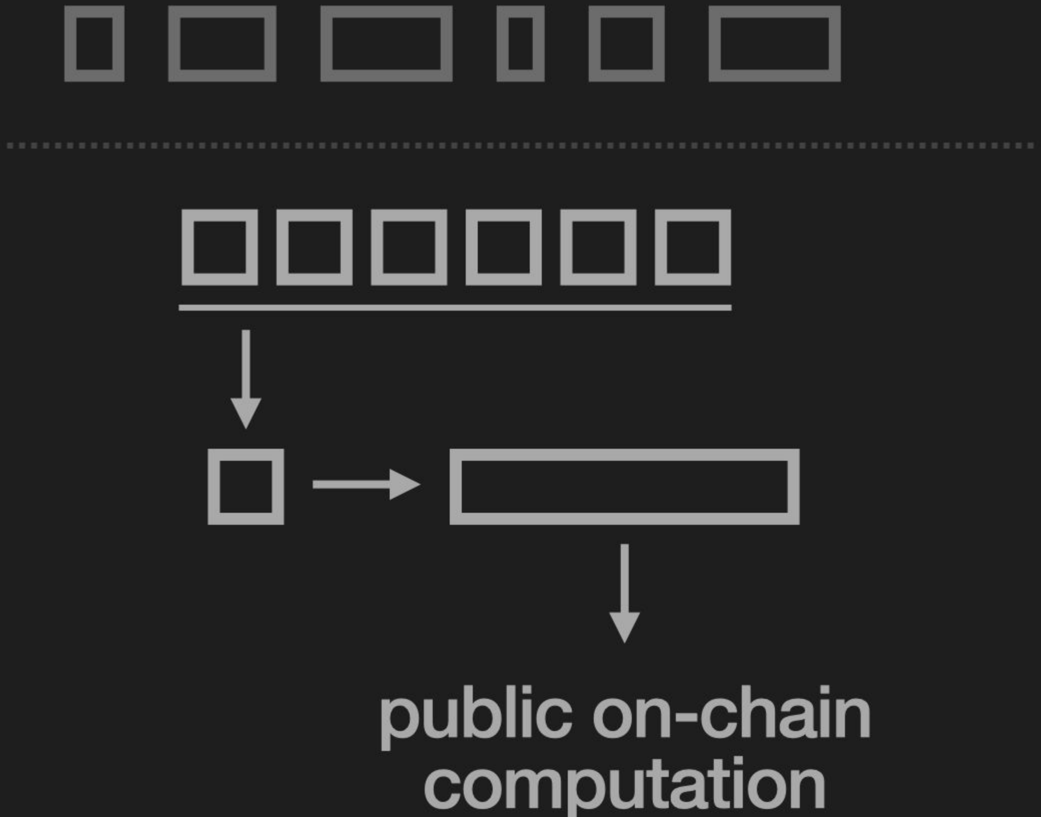


batching flows

users encrypt integer amounts with **flow encryption**

validators sum encryptions and **decrypt batch total**

individual txs have **long-term privacy**



batching flows

users encrypt integer amounts with **flow encryption**

validators sum encryptions and **decrypt batch total**

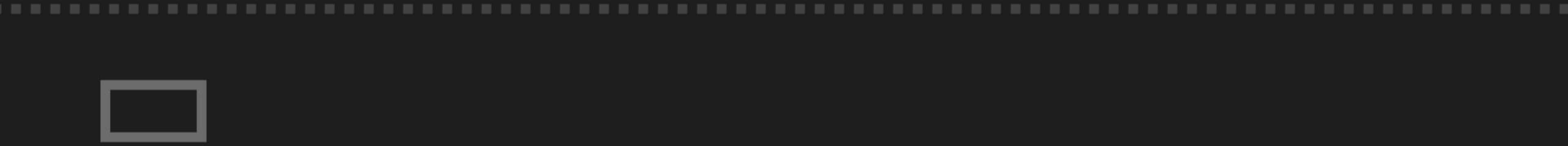
individual txs have **long-term privacy** example:

sealed-input batch swaps

on penumbra

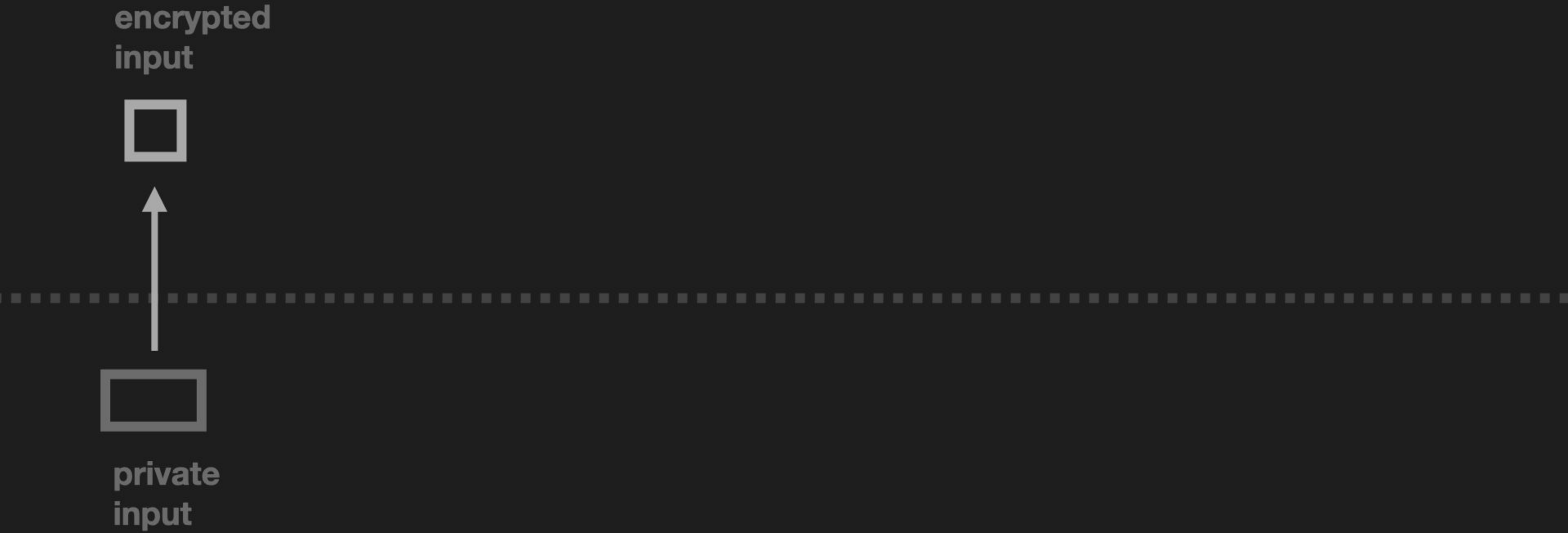
sealed-input batch swaps on penumbra (private state)

sealed-input batch swaps on penumbra (private state)

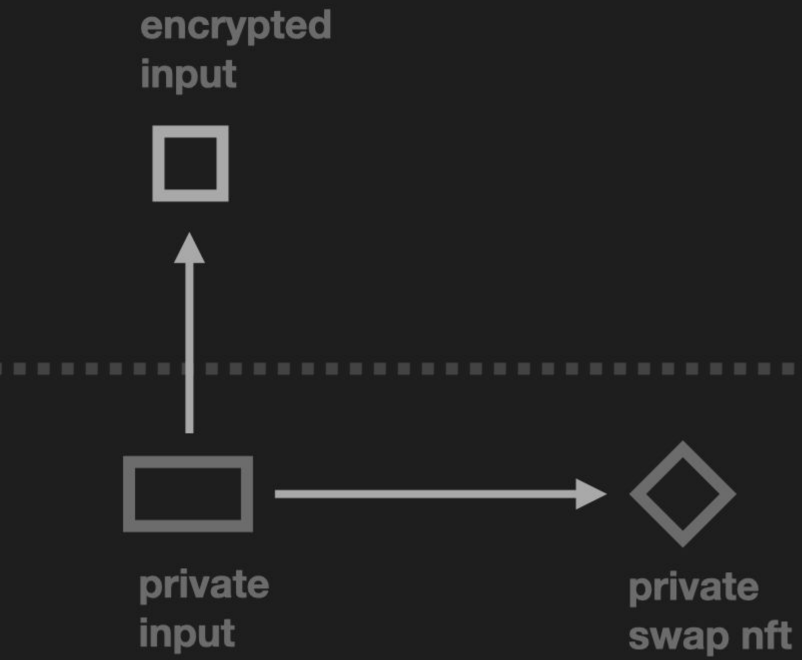


private
input

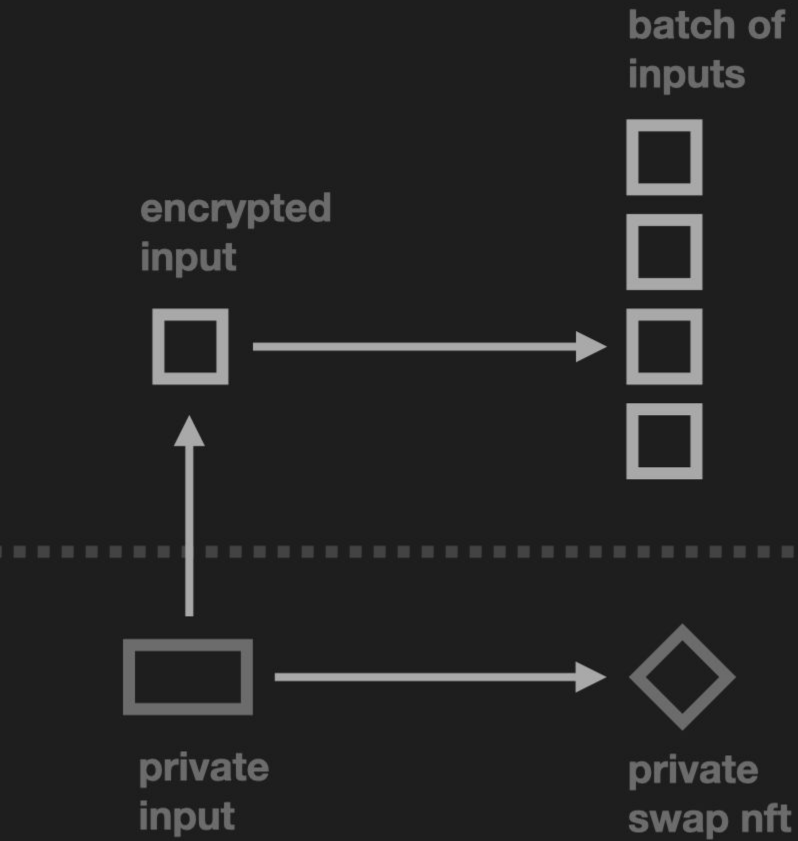
sealed-input batch swaps on penumbra (private state)



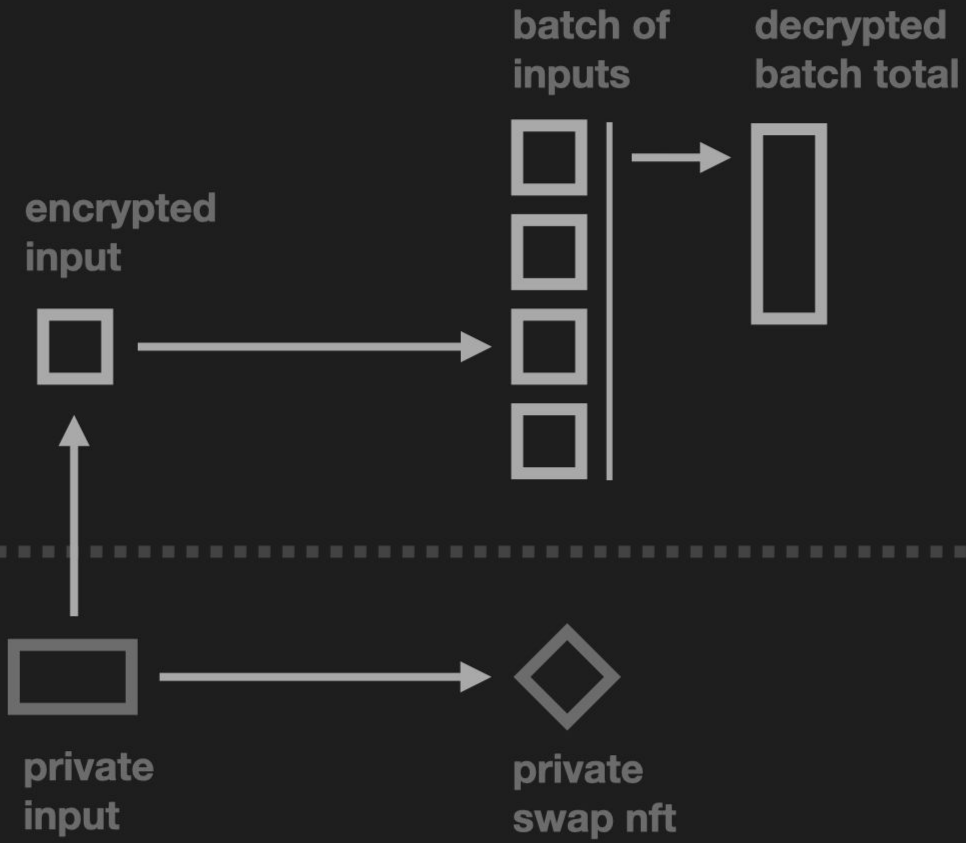
sealed-input batch swaps on penumbra (private state)



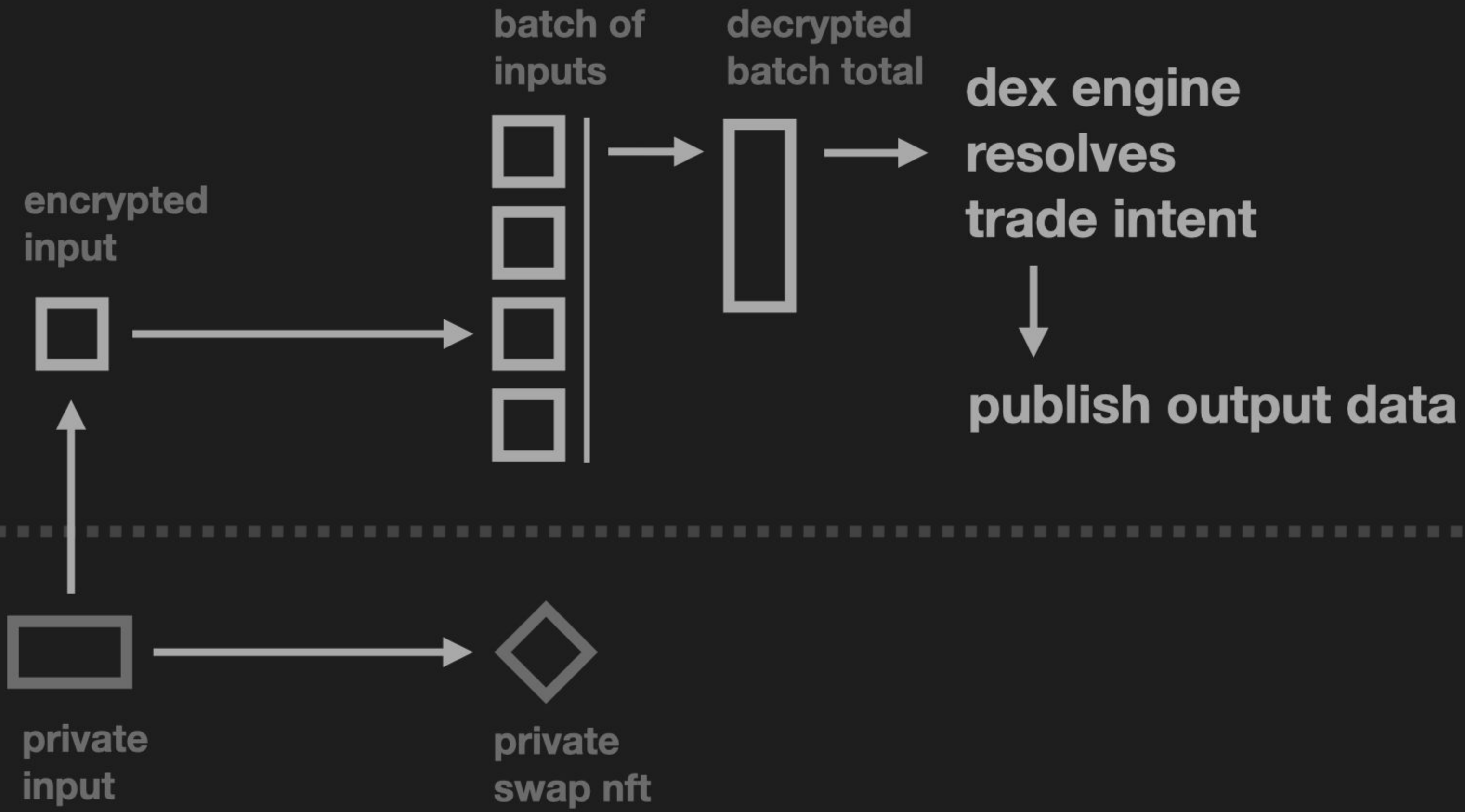
sealed-input batch swaps on penumbra (private state)



sealed-input batch swaps on penumbra (private state)



sealed-input batch swaps on penumbra (private state)



batch of inputs

decrypted batch total

dex engine resolves trade intent

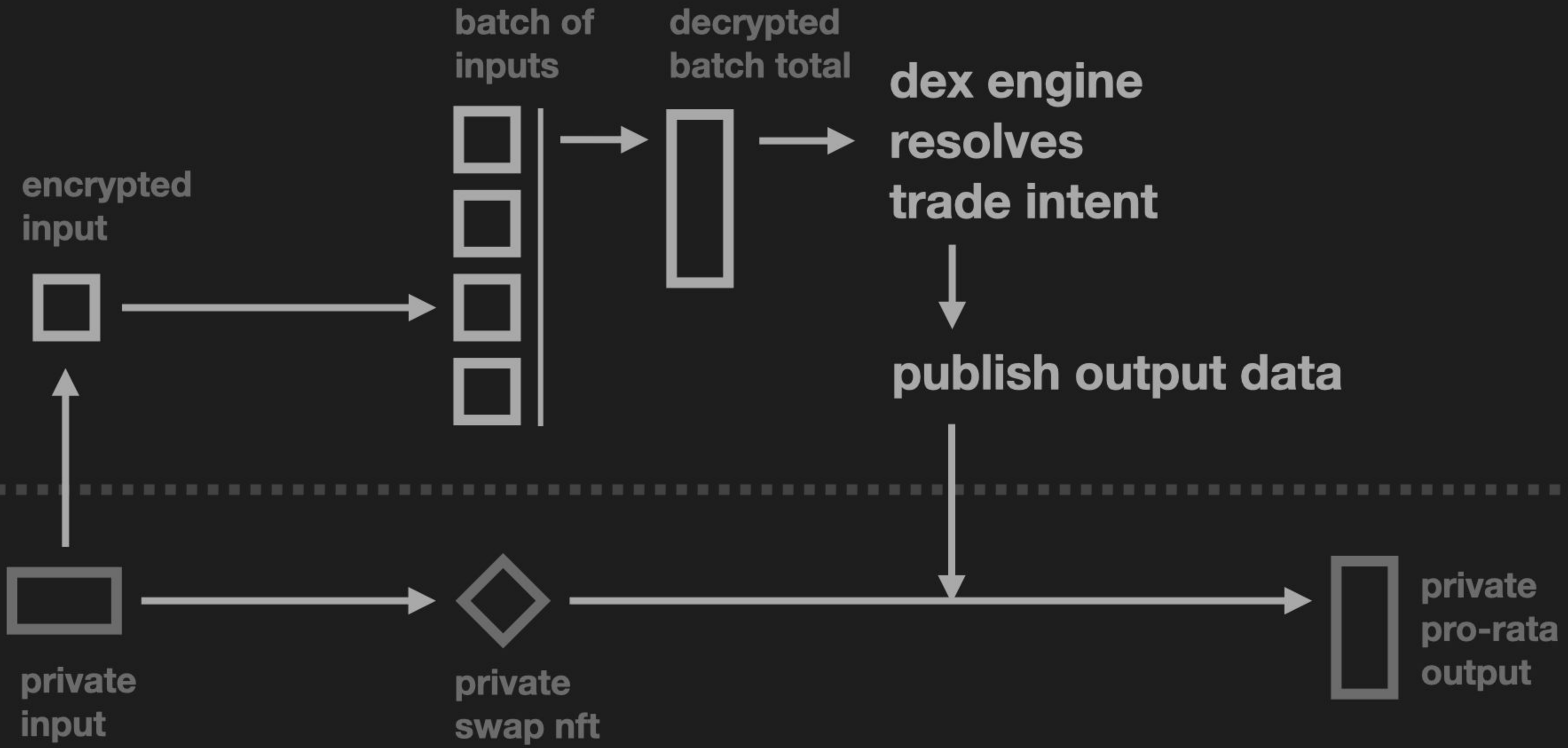
publish output data

encrypted input

private input

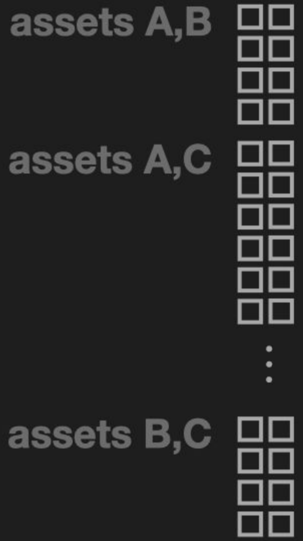
private swap nft

sealed-input batch swaps on penumbra (private state)



sealed-input batch swaps on penumbra (public state)

sealed-input batch swaps on penumbra (public state)



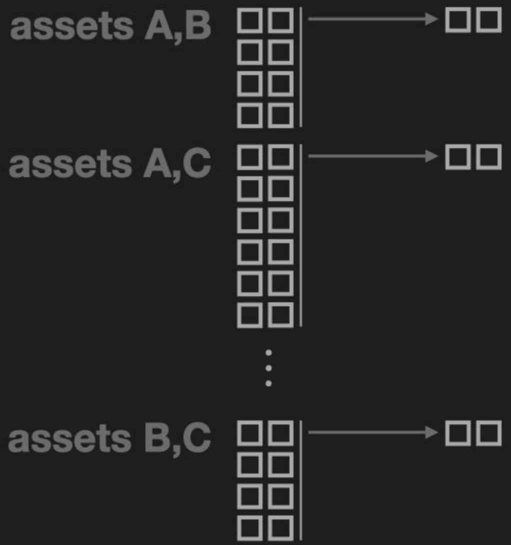
assets A,B

assets A,C

assets B,C

group
inputs
by pair

sealed-input batch swaps on penumbra (public state)



assets A,B

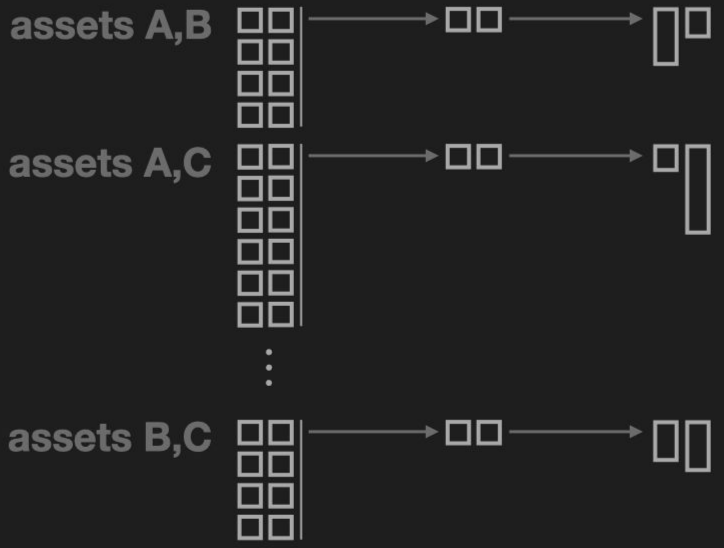
assets A,C

assets B,C

group
inputs
by pair

batch
encrypted
inputs

sealed-input batch swaps on penumbra (public state)

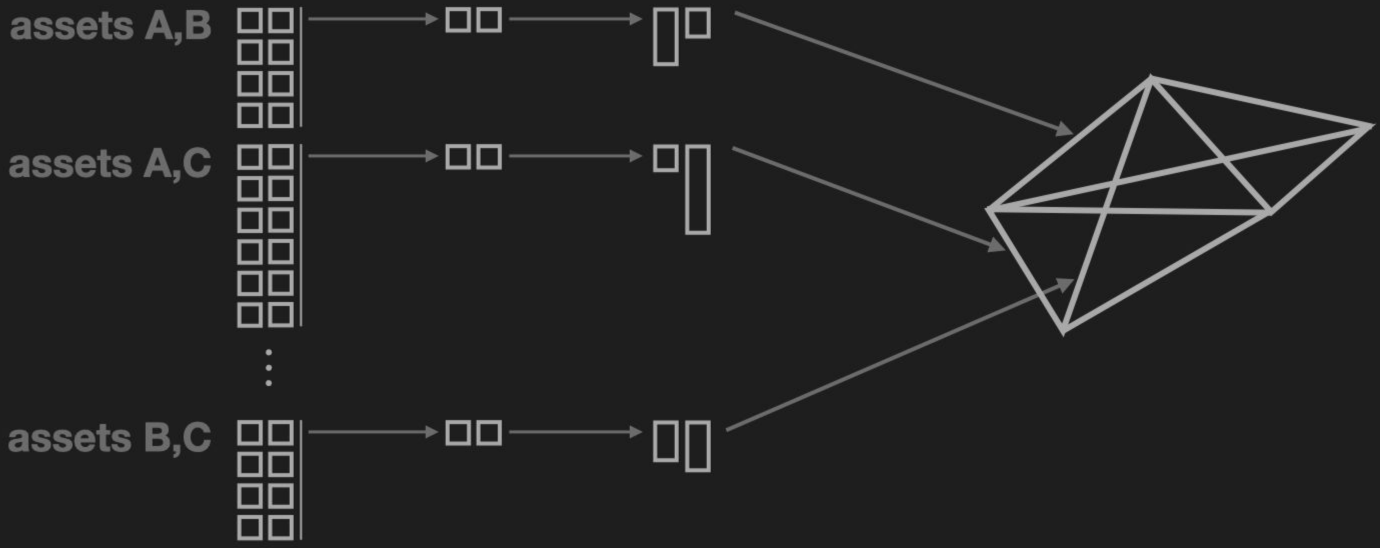


group
inputs
by pair

batch
encrypted
inputs

decrypt
batch
totals

sealed-input batch swaps on penumbra (public state)

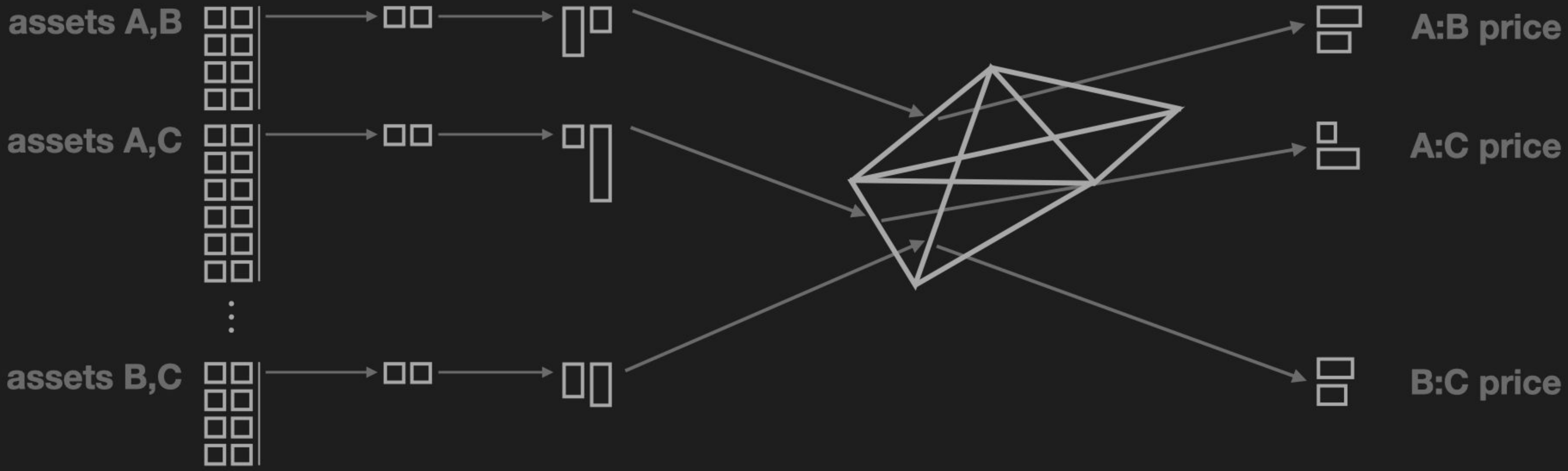


group inputs by pair

batch encrypted inputs

decrypt batch totals

sealed-input batch swaps on penumbra (public state)



assets A,B

assets A,C

assets B,C

A:B price

A:C price

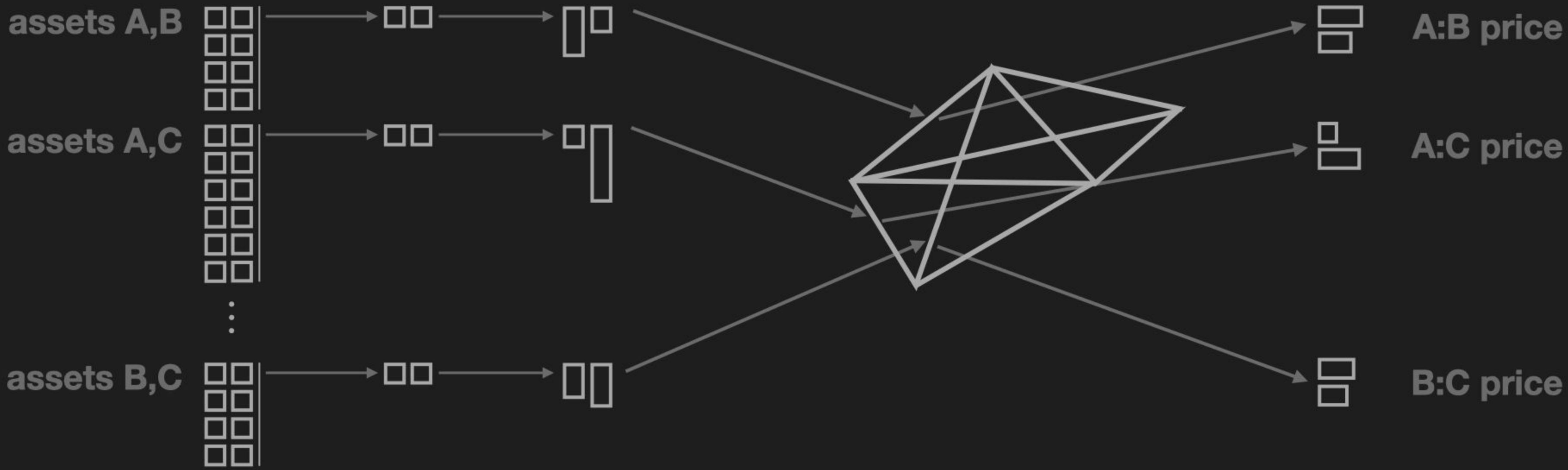
B:C price

group inputs by pair

batch encrypted inputs

decrypt batch totals

sealed-input batch swaps on penumbra (public state)



group inputs by pair

batch encrypted inputs

decrypt batch totals

globally resolve all trading intent with optimal arbitrage

shielded swaps are live
on weekly penumbra testnets

discord + github links	penumbra.zone
design docs	protocol.penumbra.zone
testnet instructions	guide.penumbra.zone
dashboards	testnet.penumbra.zone