树图区块链研究院 CONFLUX

LECTURE-3

BLOCKCHAIN: A USER'S PERSPECTIVE

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- what are smart contracts?
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- how do I interact with dapps?

PREREQUISITES

- basic familiarity with the concept of hashing
  https://medium.com/tech-tales/what-is-hashing-6edba0ebfa67
- some knowledge of programming (Node.js)
- have npm and node installed if you want to follow along

TOOLS WE WILL USE

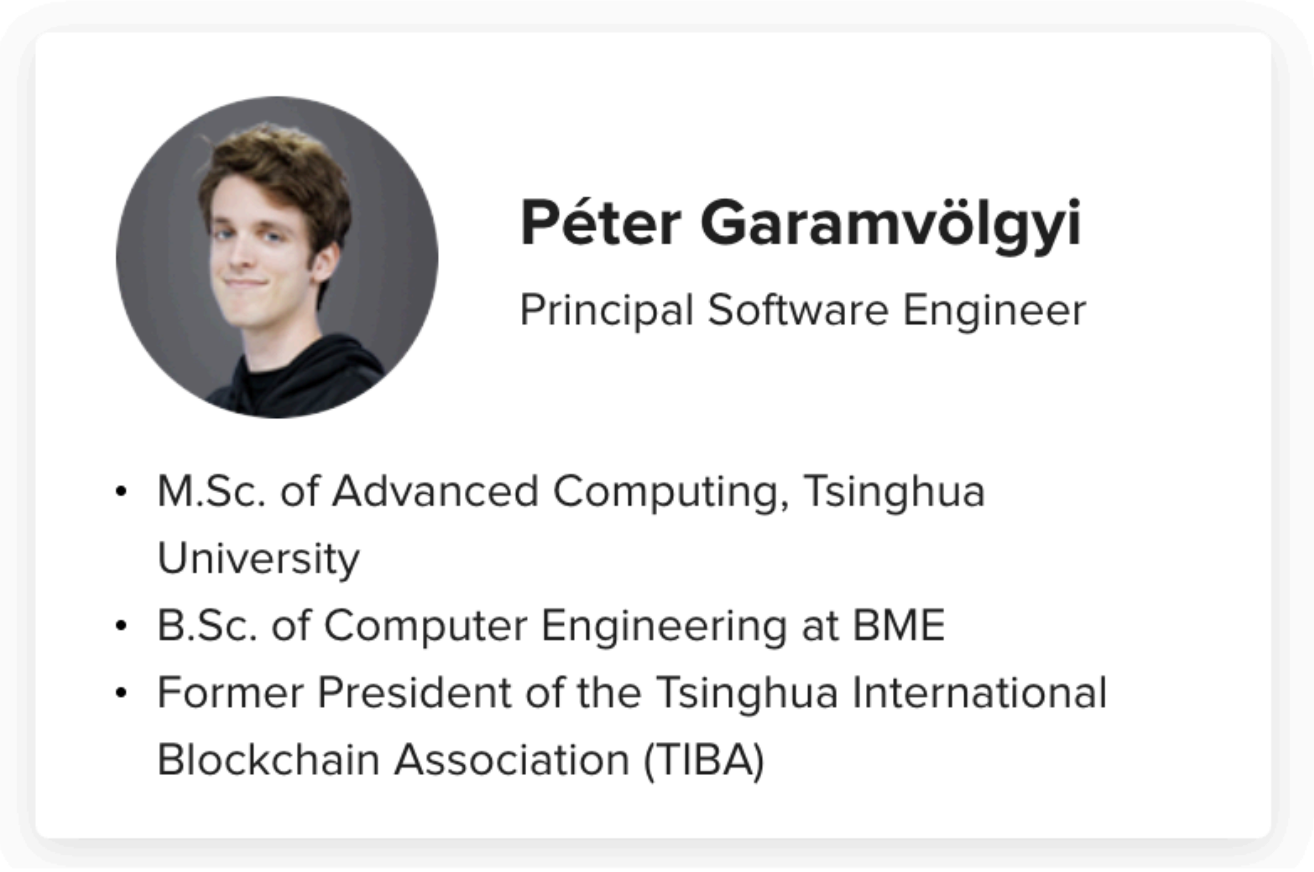
- Conflux Portal (wallet)
  portal.conflux-chain.org
- Conflux Scan (blockchain explorer)
  confluxscan.io
- Conflux Studio (IDE)
  github.com/ObsidianLabs/ConfluxStudio
- js-conflux-sdk (Node.js SDK)
  npmjs.com/package/js-conflux-sdk

QUESTIONS?

- Discord
  discord.com/invite/aCZkf2C
- Conflux Forum
  https://forum.conflux.fun

A BIT ABOUT ME

- I joined Conflux in May 2019
- working on conflux-rust:
  - improving sync
  - adding RPC and pub-sub APIs
  - implementing light nodes
  - fixing various bugs



Péter Garamvölgyi

Principal Software Engineer

- M.Sc. of Advanced Computing, Tsinghua University
- B.Sc. of Computer Engineering at BME
- Former President of the Tsinghua International Blockchain Association (TIBA)

peter@conflux-chain.org

TOPICS WE WILL COVER TODAY

- **what are accounts and transactions?**
- what is a distributed ledger?
- what are smart contracts?
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- how do I interact with dapps?

DIGITAL SIGNATURES

- digital signatures are cryptographic tools and algorithms used to prove message authenticity and integrity
  - *authenticity*: the sender is who they claim to be
  - *integrity*: the message has not been modified

DIGITAL SIGNATURES -- EXAMPLE

- let’s look at a simple transaction:

  Send 30 RMB from Peter’s account to Starbucks.
  (sent by Peter)

DIGITAL SIGNATURES -- EXAMPLE

- let’s look at a simple transaction:

  Send 30 RMB from Peter’s account to Starbucks.
  (sent by Peter)

- *authenticity*: Chris cannot forge this transaction
  (i.e. cannot buy a coffee using my money)

! Send 30 RMB from Peter’s account to Starbucks.
(sent by ~~Peter~~ **Chris**)

DIGITAL SIGNATURES -- EXAMPLE

- let's look at a simple transaction:

  Send 30 RMB from Peter's account to Starbucks.
  (sent by Peter)

- *integrity*: Chris cannot tamper with my transaction

! Send 30 RMB from Peter's account to ~~Starbucks~~ **Luckin**.
(sent by Peter)

DIGITAL SIGNATURES -- EXAMPLE

- I'll attach an unforgeable signature

  Send 30 RMB from Peter's account to Starbucks.
  Sincerely, Peter

- attackers cannot forge my signature

  Send 30 RMB from Peter's account to ~~Starbucks~~ **Luckin**.
  Sincerely, Peter !

DIGITAL SIGNATURES

- in practice, we use a private-public ECDSA keypair
- you can create signatures using the message and the private key
- you can verify signatures using the message, the signature, and the public key
- private- and public keys define an identity, central to blockchain systems

DIGITAL SIGNATURES

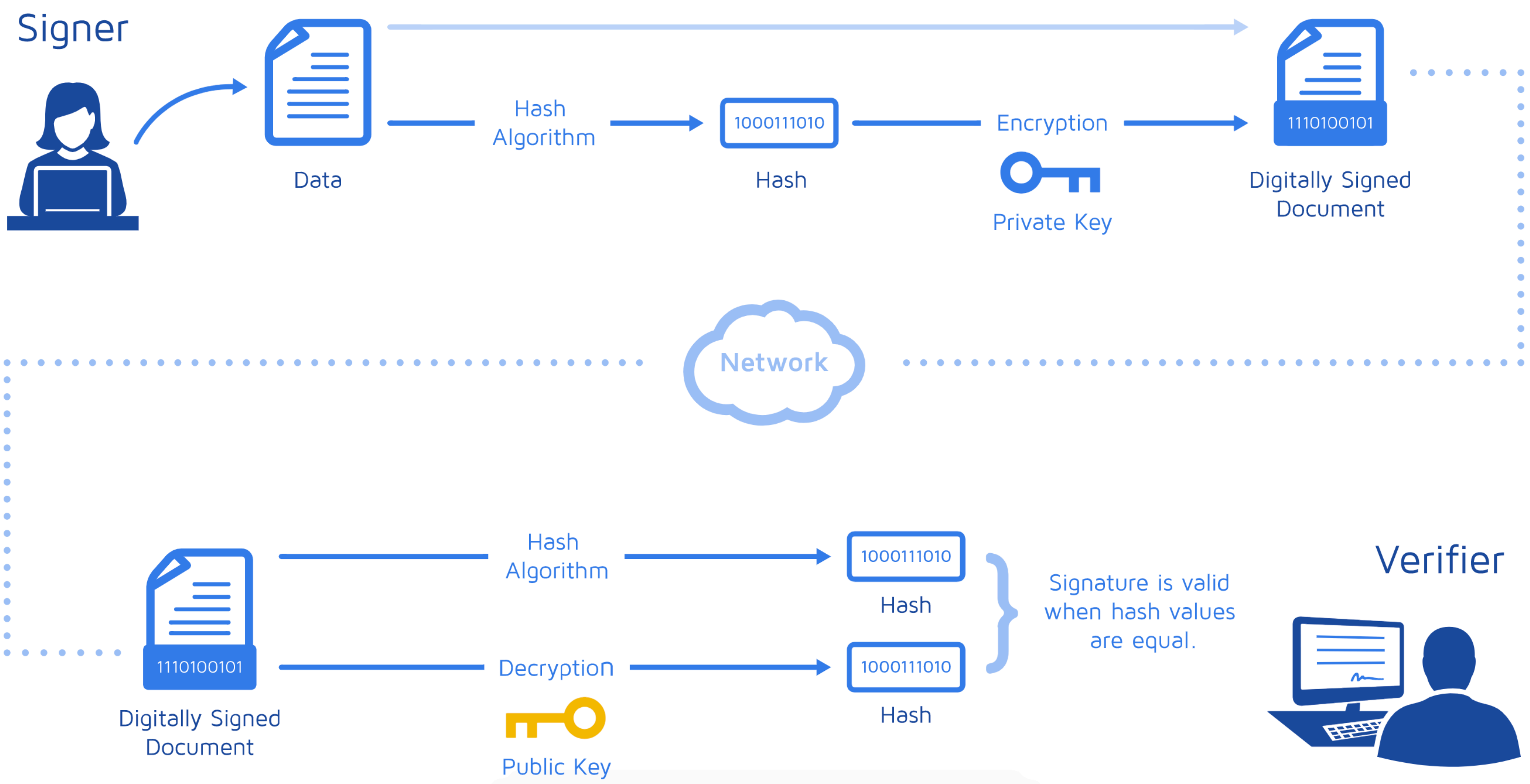


image source: DocuSign
https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq

WALLETS

- your private-public keypair is your identity
- people use wallets to store and manage these
  -- more on this in a bit

NODEJS EXAMPLE

- install Node.js and the js-conflux-sdk package

command line

```
$ mkdir conflux-playground
$ cd conflux-playground
$ npm install js-conflux-sdk@1.0.0-alpha.4
$ node

> const util = require('js-conflux-sdk/src/util');
```

NODEJS EXAMPLE

- generate a random private key

node.js

```
> const util = require('js-conflux-sdk/src/util');

> const privkey = util.sign.randomPrivateKey();
> console.log('private key:', util.format.hex(privkey));
0xab91d522b8c88754e619ed11cbb2da1ce525499d971bfb868eee5ff803acfbac
```

$$\begin{tikzcd} \text{private key} \arrow[r] & \text{public key} \arrow[r] & \text{address} \end{tikzcd}$$

32B 64B 20B

NODEJS EXAMPLE

- derive public key from private key

```
node.js
> const util = require('js-conflux-sdk/src/util');

> const pubkey = util.sign.privateKeyToPublicKey(privkey);
> console.log('public key:', util.format.hex(pubkey));
0x8f5a8321ad6c75a9139c4cd891dcc5ec548717dcfd414040fb86682cd4298...
```

$$\begin{tikzcd}\text{private key} \arrow[r] & \text{public key} \arrow[r] & \text{address}\end{tikzcd}$$

32B 64B 20B

NODEJS EXAMPLE

- derive address from public key

```
node.js
> const util = require('js-conflux-sdk/src/util');

> const address = util.sign.publicKeyToAddress(pubkey);
> console.log('address:', util.format.hex(address));
0x18dc2130e3da374df9e04d94ec0113a2e1b82695
```

$$\begin{tikzcd}\text{private key} \arrow[r] & \text{public key} \arrow[r] & \text{address}\end{tikzcd}$$

32B 64B 20B

NODEJS EXAMPLE

- note: this does not work the other way around
- just by knowing my address, you cannot find my privacy

$$\begin{tikzcd}
\text{private key} \arrow[r, shift left] & \text{public key} \arrow[l, shift left, "\times" description] \arrow[r, shift left] & \text{address} \arrow[l, shift left, "\times" description] \\
\text{32B} & \text{64B} & \text{20B}
\end{tikzcd}$$

NODEJS EXAMPLE

- sign a message using our private key

node.js

```
> const Message = require('js-conflux-sdk/src/message');

> const msg = new Message("Hello! Sincerely, Peter");
> msg.sign(privkey);
> console.log('signed message:', msg)

Message {
  message: 'Hello! Sincerely, Peter',
  signature: '0x3890bd3a6a3a347bd1a685d75b0f16e1b6467ea01f4f88e007
               5e7c2620dc59a25fb8e5cf840d401e3092678c12f05d6cdff3a7
               33cfb3f503bebbbb105e33223001'
}
```

NODEJS EXAMPLE

- check message authenticity

node.js

```
> const Message = require('js-conflux-sdk/src/message');
> const assert = require('assert');

> let recovered = Message.recover(msg.signature, msg.hash)
> assert.equal(recovered, util.format.hex(pubkey))
```

NODEJS EXAMPLE

- tamper with message

node.js

```
> const Message = require('js-conflux-sdk/src/message');
> const assert = require('assert');

> console.log('message:', msg.message);
message: Hello! Sincerely, Peter

> msg.message = "Hello! Sincerely, ----- Hacker";
> recovered = Message.recover(msg.signature, msg.hash)
> assert.notEqual(recovered, util.format.hex(pubkey))
```

NODEJS EXAMPLE

- tamper with message

```
node.js
> const Message = require('js-conflux-sdk/src/message');
> const assert = require('assert');

> console.log('message:', msg.message);
message: Hello! Sincerely, Peter

> msg.message = "Hello! Sincerely, ----- Hacker";
> recovered = Message.recover(msg.signature, msg.hash)
> assert.notEqual(recovered, util.format.hex(pubkey))
```

NODEJS EXAMPLE -- TRY FOR YOURSELF!

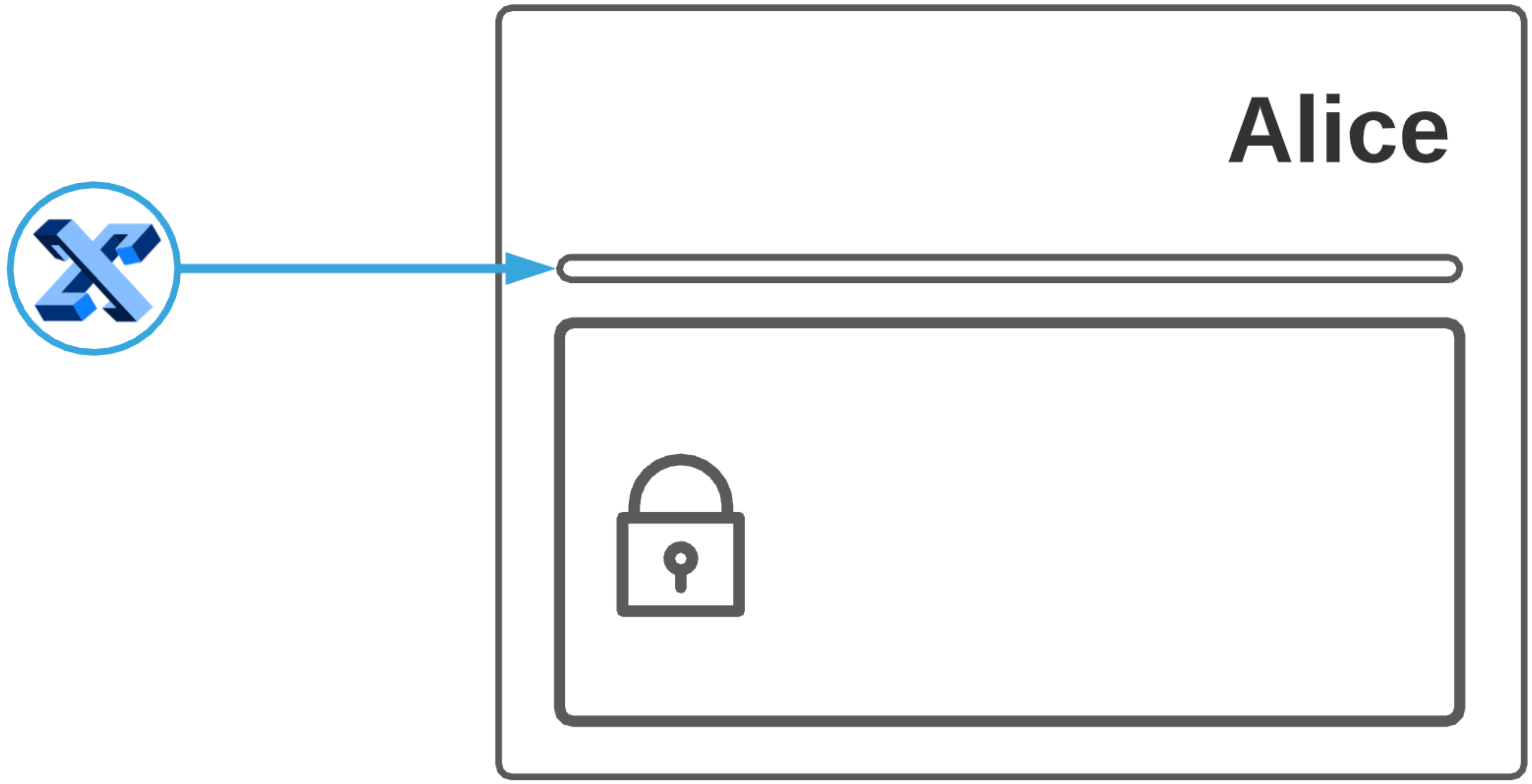
- read and run the full example
  gist.github.com/Thegaram/0652c8e359f73b1772311c8ed34a32c0
- js-conflux-sdk documentation
  https://www.npmjs.com/package/js-conflux-sdk/v/1.0.0-alpha.4

ACCOUNT

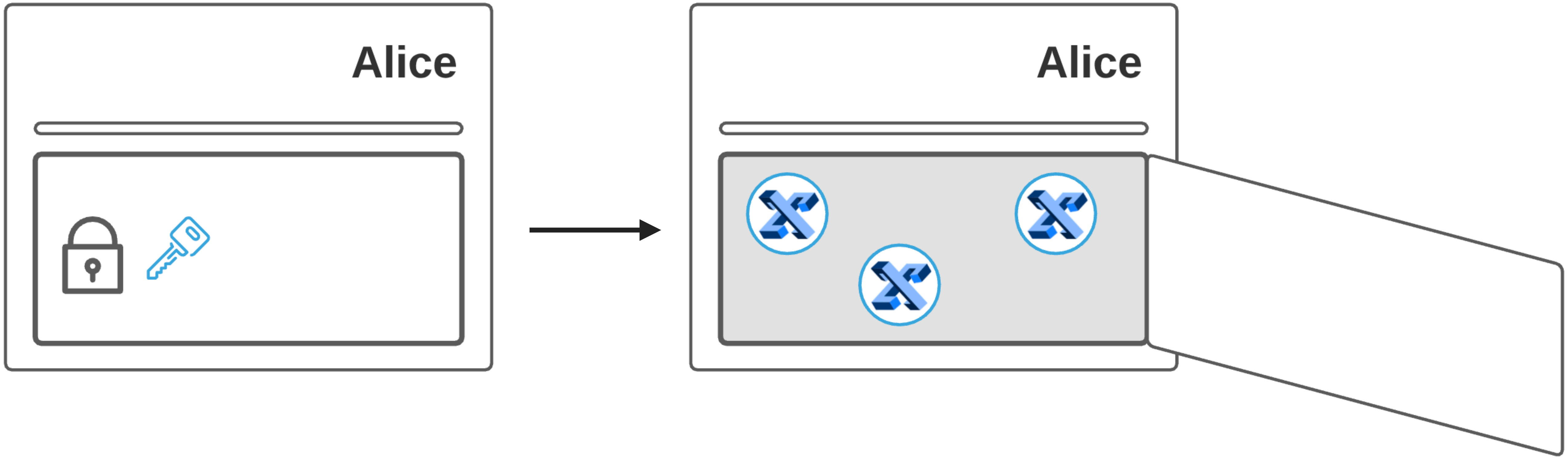
- for each identity, there is a corresponding account maintained on the blockchain
- the account captures a state at a given time: the current balance, the number of transactions sent, code, …
- your address will not change
  (though you might have multiple addresses)
- the account state associated to your address will change as you send and receive transactions

ACCOUNT

ACCOUNT



ACCOUNT



ACCOUNT

name on the mailbox	public key or address
key to the padlock	private key
contents of the mailbox	account state (changes over time)

ACCOUNT

- the JSON-representation of a new account looks like this

```json
{
  "accumulatedInterestReturn": "0x0",
  "admin": "0x0000000000000000000000000000000000000000",
  "balance": "0x0",
  "codeHash": "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82...",
  "collateralForStorage": "0x0",
  "nonce": "0x0",
  "stakingBalance": "0x0"
}
```

see: https://developer.conflux-chain.org/docs/conflux-doc/docs/json_rpc/#cfx_getaccount

ACCOUNT -- BALANCE

- each account has a balance associated with it

json

```json
{
  "accumulatedInterestReturn": "0x0",
  "admin": "0x0000000000000000000000000000000000000000",
  "balance": "0x0",
  "codeHash": "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82...",
  "collateralForStorage": "0x0",
  "nonce": "0x0",
  "stakingBalance": "0x0",
}
```

ACCOUNT -- BALANCE

- each account has a balance associated with it
- the denominations are:

name	value
Drip	1 Drip
GDrip	10^{9} Drip
Conflux (CFX)	10^{18} Drip

ACCOUNT -- NONCE

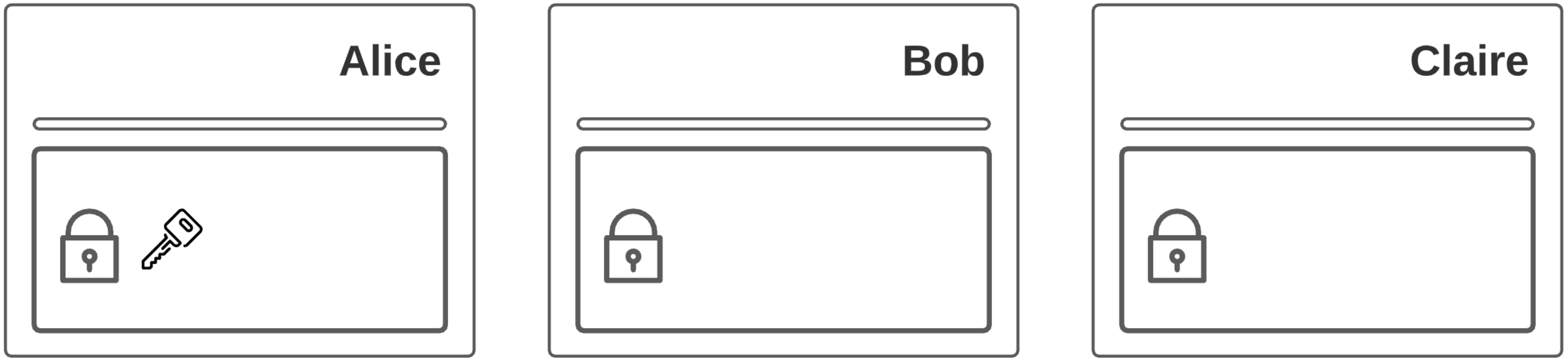
- nonce: number of accepted transactions from this account

```json
{
  "accumulatedInterestReturn": "0x0",
  "admin": "0x0000000000000000000000000000000000000000",
  "balance": "0x0",
  "codeHash": "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82...",
  "collateralForStorage": "0x0",
  "nonce": "0x0",
  "stakingBalance": "0x0"
}
```

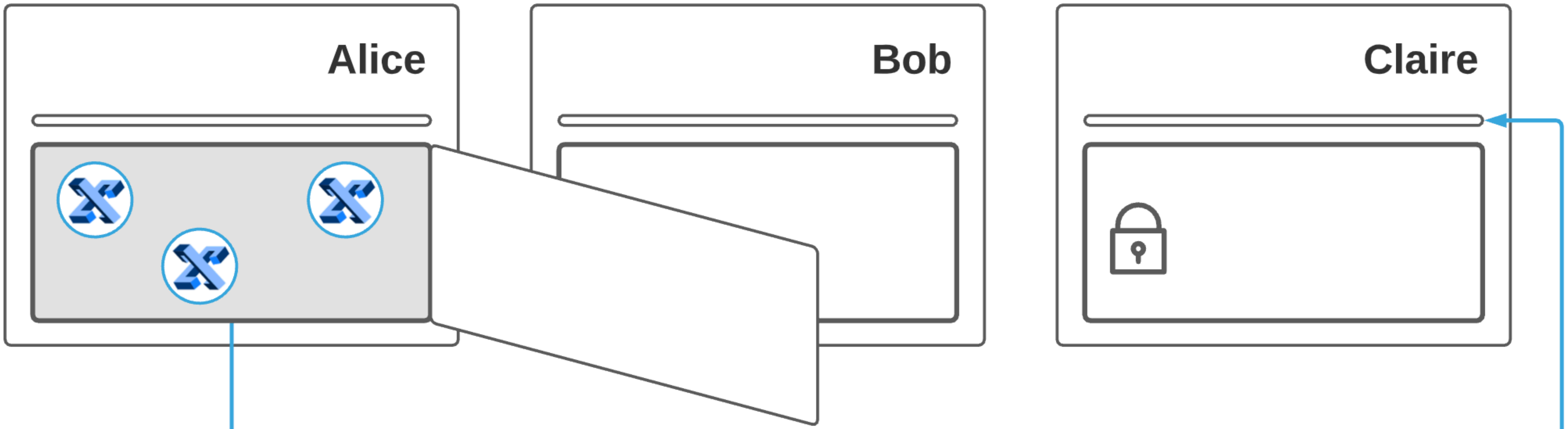
SIMPLE PAYMENT TRANSACTIONS

- simply sending a number of coins (tokens) from one address to another
- the sender signs the transaction using their private key
- if you do not have my private key, you cannot send on my behalf

SIMPLE PAYMENT TRANSACTIONS



SIMPLE PAYMENT TRANSACTIONS



TRANSACTIONS

- the JSON-representation of a transaction looks like this

json

```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x145834072064DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

see: https://developer.conflux-chain.org/docs/conflux-doc/docs/json_rpc/#cfx_gettransactionbyhash

TRANSACTIONS -- SENDER & RECEIVER

- payment transactions specify a sender and a receiver

json
```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407 2064DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- VALUE

- you also need to specify how much you want to send

json
```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407206 4DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- VALUE

- converting from hex string to actual value

node.js

```
> const { Drip } = require('js-conflux-sdk');

> const cfx = Drip.fromDrip('0x3635c9adc5dea00000').toCFX();
> console.log('in CFX:', cfx);
in CFX: 1000
```

TRANSACTIONS -- GAS

- you might need to pay for transaction -- more on this later

json
```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407206 4DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- NONCE

- nonce is the number of the transaction

```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407206 4DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- NONCE

- nonce is the number of the transaction

```
{ from: Peter, to: Alice, nonce: 1, … }
```

will be executed before

```
{ from: Peter, to: Bob, nonce: 2, … }
```

TRANSACTIONS -- DATA

- data is used for smart contract calls -- more on this later

json
```
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407 2064DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- SIGNATURE

- finally, a transaction needs to be signed by the sender

json
```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407 2064DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- SIGNATURE

node.js

```
> const Account = require('js-conflux-sdk/src/account');
> const account = new Account({ privateKey: '0xab91d522b8c...' });

> account.signTransaction({
    from: '0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636',
    to: '0x145834072064DcD9B931237B5aEe217c241e3644',
    ...
});

Transaction {
  from: '0x18dc2130e3da374df9e04d94ec0113a2e1b82695',
  to: '0xb2988210c05a43ebd76575f5421ef84b120ebf80',
  ...
  v: 1,
  r: '0xc55c89be3dc50d34521fa14b3dacb27a6149da7e08ed467edc40b...',
  s: '0x310fc83482196742a4820b8eb490a88ec62c2b59125c2bf3a8cea...'
}
```

TRANSACTIONS -- SIGNATURE

- finally, a transaction needs to be signed by the sender

```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x14583407206 4DcD9B931237B5aEe217c241e3644",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0x",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

TRANSACTIONS -- TRANSACTION HASH

- transactions are identified by the *transaction hash*

```
0x53fe995edeec7d241791ff32635244e94ecfd722c9fe90f34ddf59082d814514
```

- You can do queries like these:
  - Check if transaction `0x53fe995` succeeded
  - Find the sender address of transaction `0x53fe995`
  - Get the execution results of transaction `0x53fe995`

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- **what is a distributed ledger?**
- what are smart contracts?
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- how do I interact with dapps?

WHAT IS IN A BLOCK?

- a block is just a list of transactions (block body)
- … plus some metadata (block header)

BLOCKS

- the JSON-representation of a block looks like this

json

```
{
  "hash": "0xc8f5310402330767adb624f436fa579ca3a6e28bb33f09ec...",
  "miner": "0x1905c5723adf66f14f9a33d6d99263cb00c2992c",
  "nonce": "0x30176787e0290260",
  "parentHash": "0xb11d32edb8e8f09941000c898269f665d275342694...",
  "epochNumber": "0x4bad97",
  "refereeHashes": [ ... ],
  "transactions": [
    "0x0b5b3d8f542a3c830ec57b83c07d128224171247 8edab3ba8e1a05...",
    "0x96b46d10999cda47c4fda1dab7e8e29ce68f5b9ae2d707fde008c7...",
    ...
  ],
}
```

BLOCKS -- BLOCK HASH

- just like txs, blocks are uniquely identified by their hashes

json
```json
{
  "hash": "0xc8f5310402330767adb624f436fa579ca3a6e28bb33f09ec...",
  "miner": "0x1905c5723adf66f14f9a33d6d99263cb00c2992c",
  "nonce": "0x30176787e0290260",
  "parentHash": "0xb11d32edb8e8f09941000c898269f665d275342694...",
  "epochNumber": "0x4bad97",
  "refereeHashes": [ ... ],
  "transactions": [
    "0x0b5b3d8f542a3c830ec57b83c07d1282241712478edab3ba8e1a05...",
    "0x96b46d10999cda47c4fda1dab7e8e29ce68f5b9ae2d707fde008c7...",
    ...
  ],
}
```

BLOCKS -- MINER & NONCE

- some information about the creator of the block (see PoW)

json

```json
{
  "hash": "0xc8f5310402330767adb624f436fa579ca3a6e28bb33f09ec...",
  "miner": "0x1905c5723adf66f14f9a33d6d99263cb00c2992c",
  "nonce": "0x30176787e0290260",
  "parentHash": "0xb11d32edb8e8f09941000c898269f665d275342694...",
  "epochNumber": "0x4bad97",
  "refereeHashes": [ ... ],
  "transactions": [
    "0x0b5b3d8f542a3c830ec57b83c07d1282241712478edab3ba8e1a05...",
    "0x96b46d10999cda47c4fda1dab7e8e29ce68f5b9ae2d707fde008c7...",
    ...
  ],
}
```

BLOCKS -- PARENT, EPOCH & REFEREES

- information about the blocks position on our ledger

json

```
{
  "hash": "0xc8f531040233067adb624f436fa579ca3a6e28bb33f09ec...",
  "miner": "0x1905c5723adf66f14f9a33d6d99263cb00c2992c",
  "nonce": "0x30176787e0290260",
  "parentHash": "0xb11d32edb8e8f09941000c898269f665d275342694...",
  "epochNumber": "0x4bad97",
  "refereeHashes": [ ... ],
  "transactions": [
    "0x0b5b3d8f542a3c830ec57b83c07d128224171247 8edab3ba8e1a05...",
    "0x96b46d10999cda47c4fda1dab7e8e29ce68f5b9ae2d707fde008c7...",
    ...
  ],
}
```

BLOCKS -- TRANSACTIONS

- and most importantly: a list of transactions

```json
{
  "hash": "0xc8f5310402330767adb624f436fa579ca3a6e28bb33f09ec...",
  "miner": "0x1905c5723adf66f14f9a33d6d99263cb00c2992c",
  "nonce": "0x30176787e0290260",
  "parentHash": "0xb11d32edb8e8f09941000c898269f665d275342694...",
  "epochNumber": "0x4bad97",
  "refereeHashes": [ ... ],
  "transactions": [
    "0x0b5b3d8f542a3c830ec57b83c07d1282241712478edab3ba8e1a05...",
    "0x96b46d10999cda47c4fda1dab7e8e29ce68f5b9ae2d707fde008c7...",
    ...
  ],
}
```

SO… WHAT IS A BLOCKCHAIN?

- our end goal is to agree on which transactions are valid and which are not
- consider this scenario:

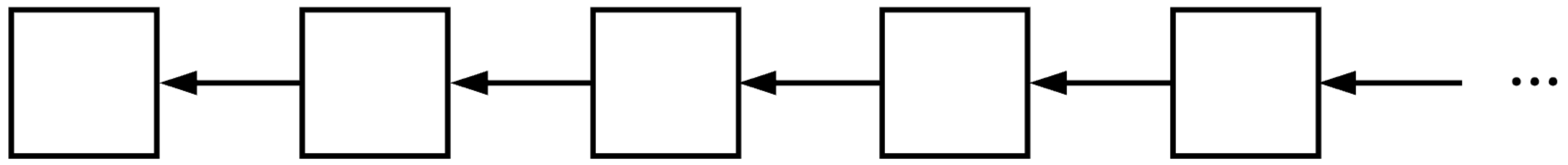
  initial balance: 10 CFX

  tx1: send 10 CFX to Bobby's Cafe

  tx2: send 10 CFX to my other account
- I cannot spend the same money twice! we need to agree which transaction succeeds: tx1 or tx2 (not both)

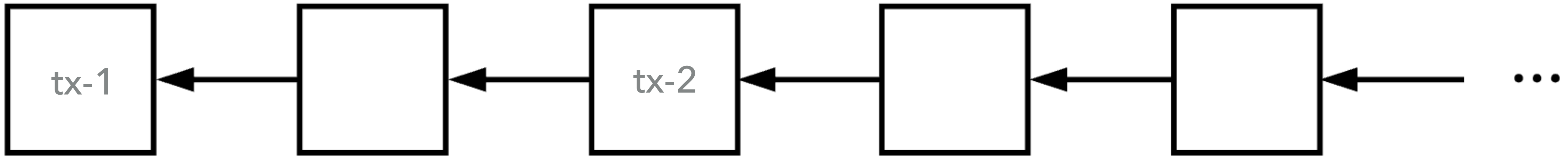
THE LEDGER OF BITCOIN AND ETHEREUM

- in Bitcoin and Ethereum, each block references exactly one previous block by its hash, thus forming a chain of blocks



THE LEDGER OF BITCOIN AND ETHEREUM -- TX ORDER

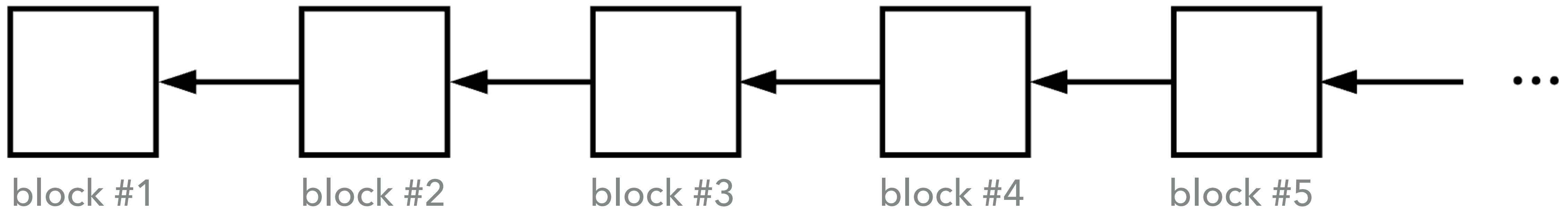
- the chain of blocks defines the canonical order of txs



… by sealing blocks with Proof-of-Work, it becomes extremely hard to change the chain (and thus to change the order)

THE LEDGER OF BITCOIN AND ETHEREUM -- BLOCK NUMBER

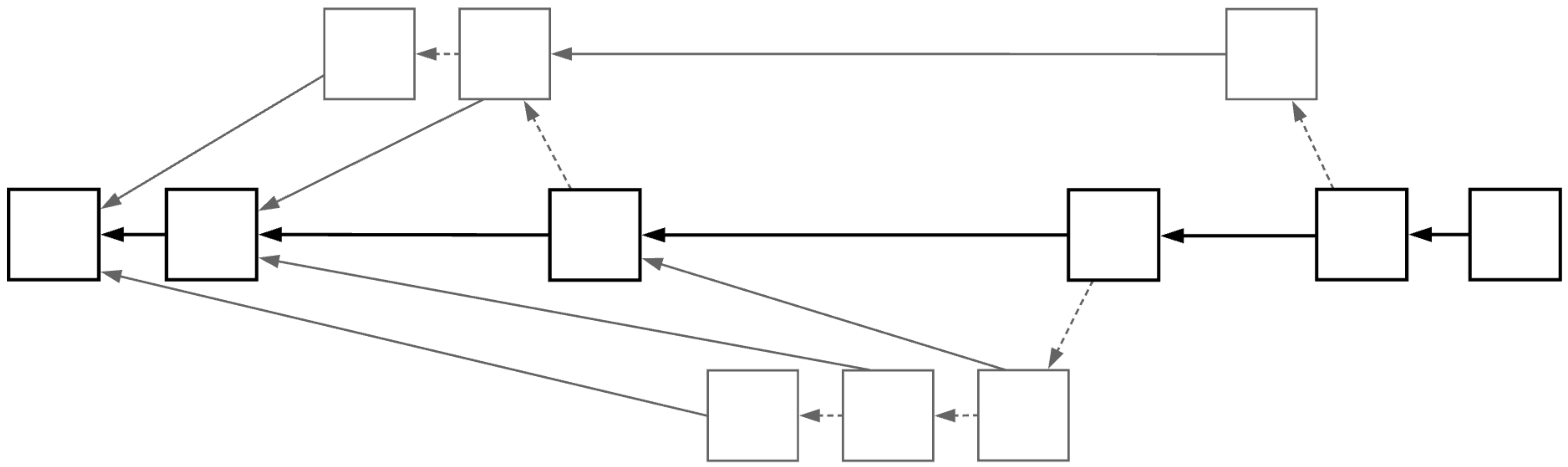
- we can reference blocks by their number or block hash
- the state of an account might change from block to block



balance(Alice) = 10 balance(Alice) = 0 balance(Alice) = 100

THE LEDGER OF CONFLUX -- TREE-GRAPH

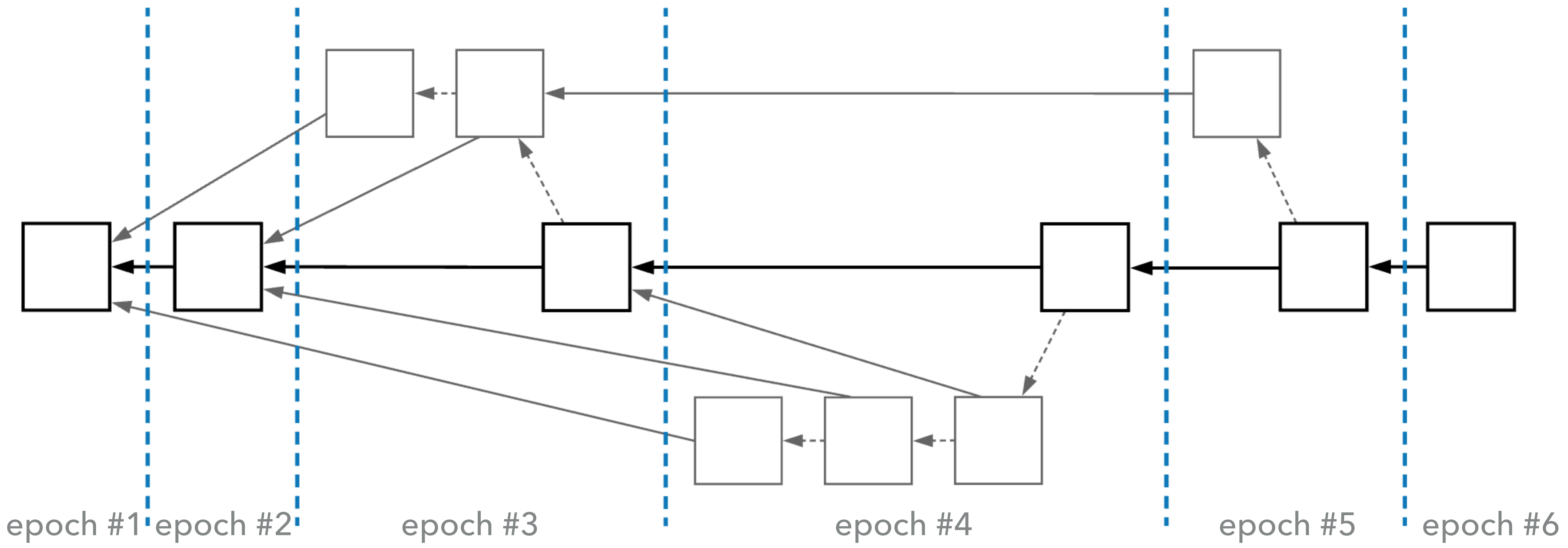
- Conflux uses a Tree-Graph ledger instead of a single chain



… this way, if two blocks are created simultaneously, the system can use both, thus increasing throughput and security

THE LEDGER OF CONFLUX -- EPOCHS

- we deterministically divide the ledger into epochs



… the last block in each epoch is called the pivot block

THE LEDGER OF CONFLUX -- FURTHER READING

- Conflux 101 Webinar
  youtube.com/playlist?list=PLoO0tXb18JZ8atJW5bU-GCiykL9yfc3xR
- whitepaper
  usenix.org/conference/atc20/presentation/li-chenxing
- technical presentation
  confluxnetwork.org/static/Conflux_Technical_Presentation_20200309.pdf

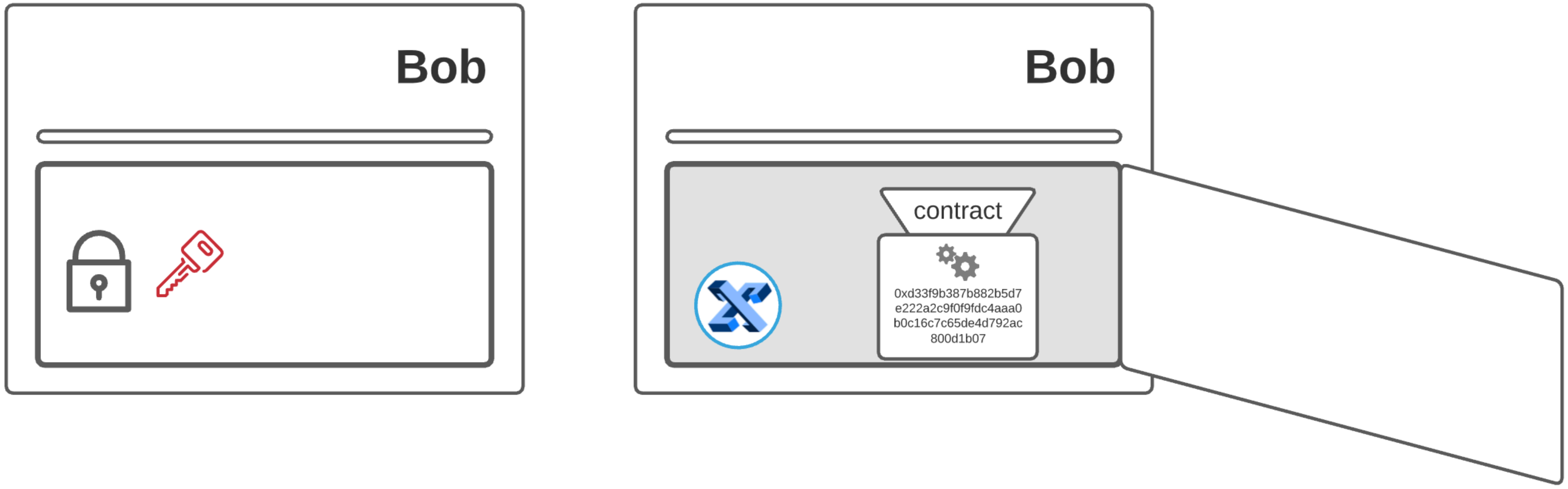
TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- **what are smart contracts?**
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- how do I interact with dapps?

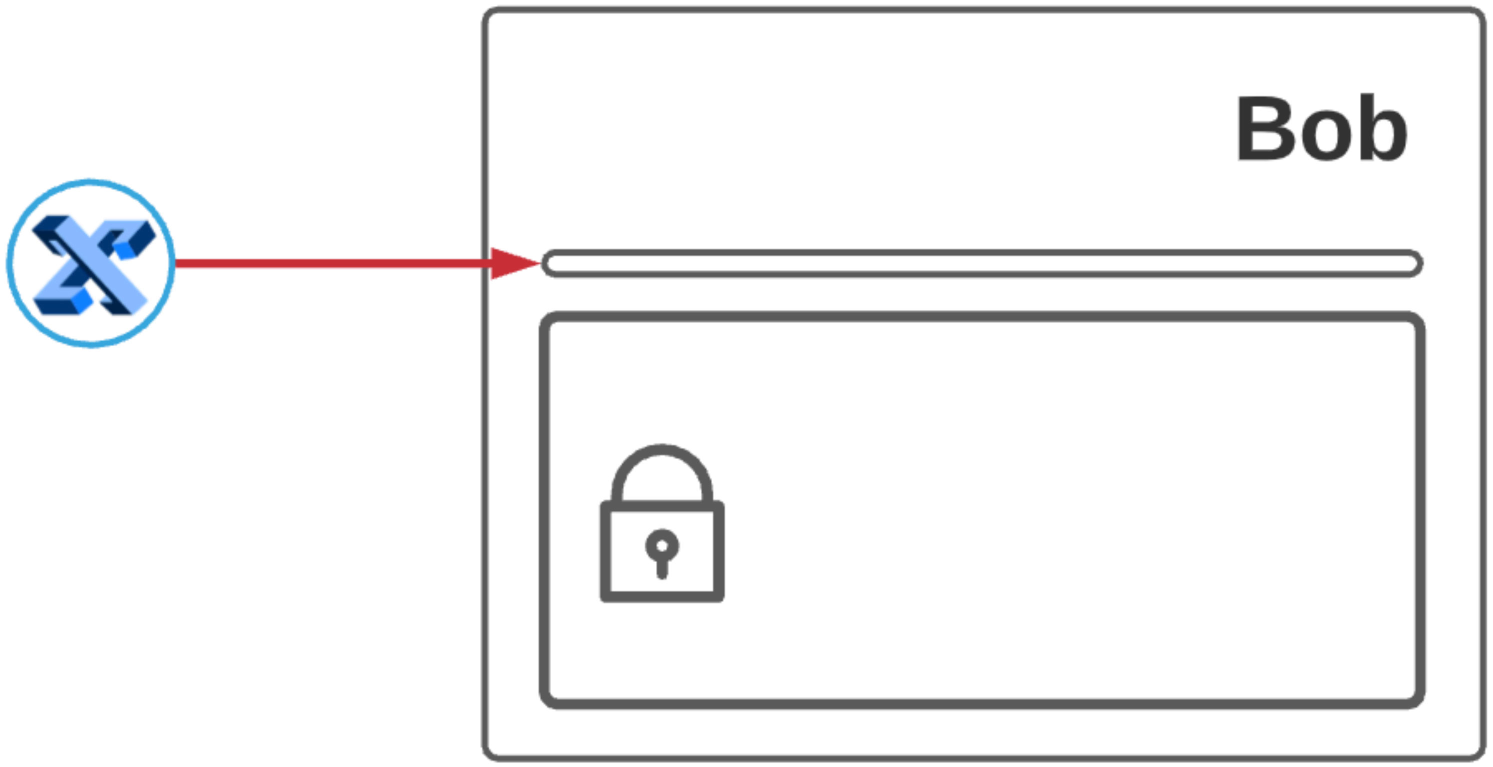
SMART CONTRACTS

- imagine that your mailbox has a machine inside that decides what to do with the coins
- moreover, you can also pass it notes with instructions, not just coins
- this machine can be unique to each box, built by its owner

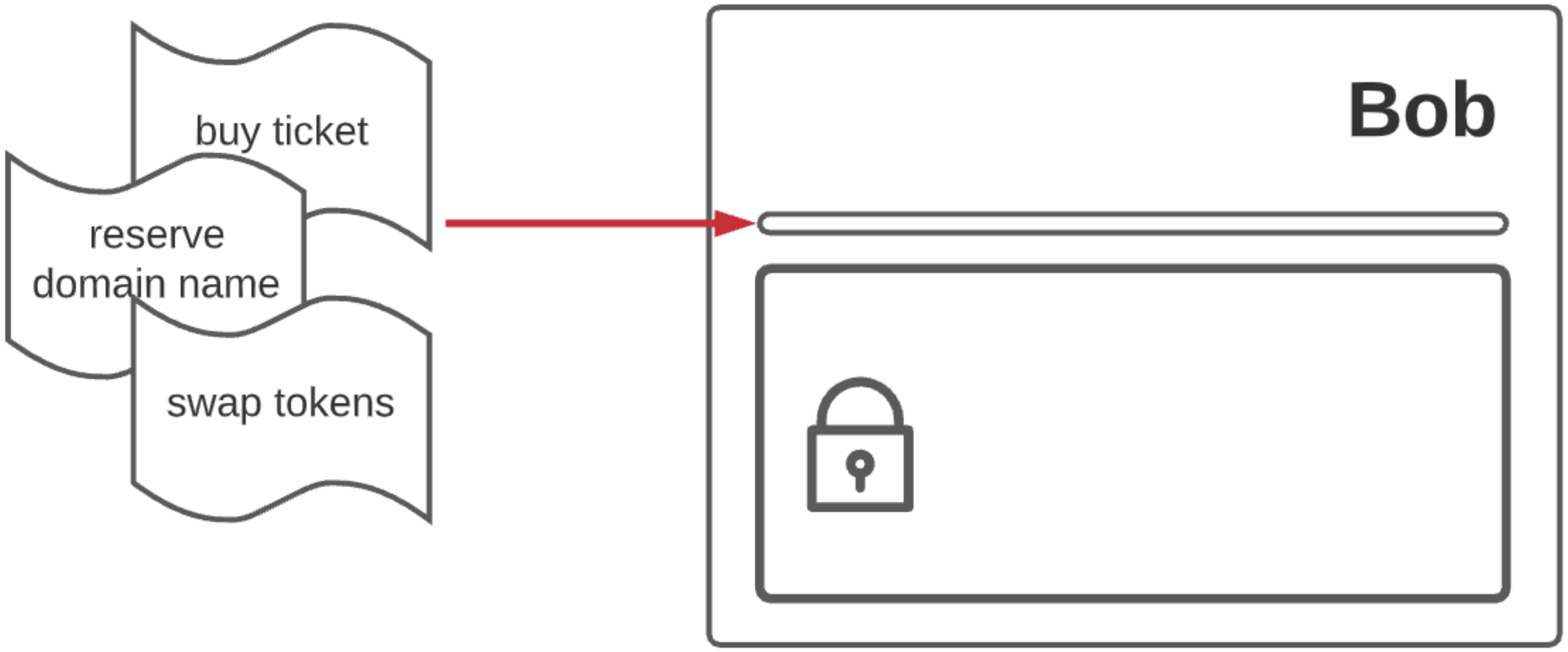
SMART CONTRACTS



SMART CONTRACTS



SMART CONTRACTS

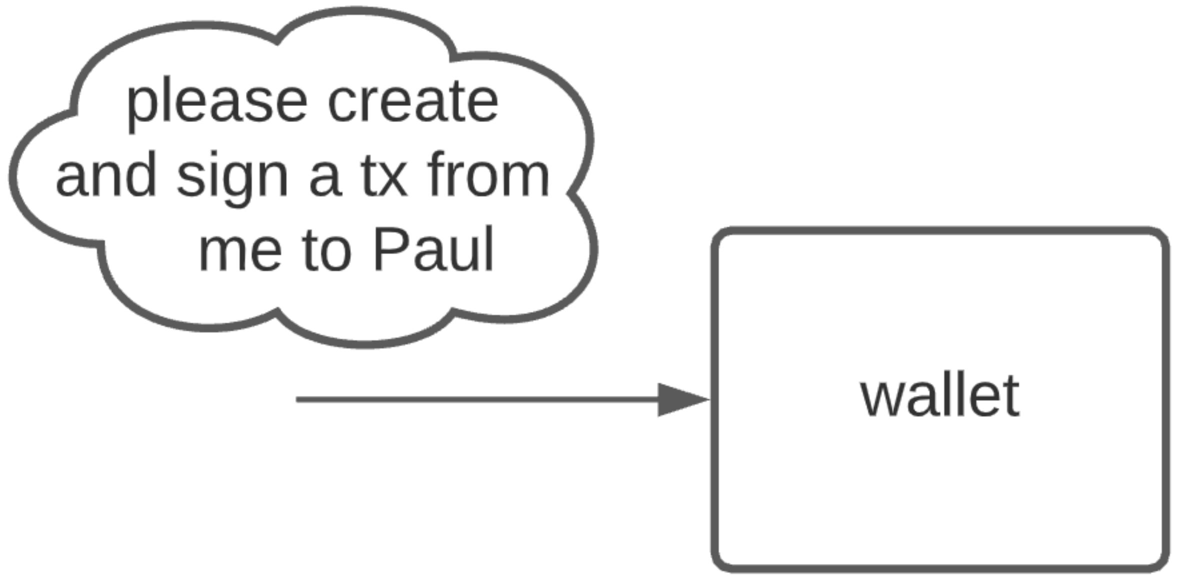


SMART CONTRACT TRANSACTIONS

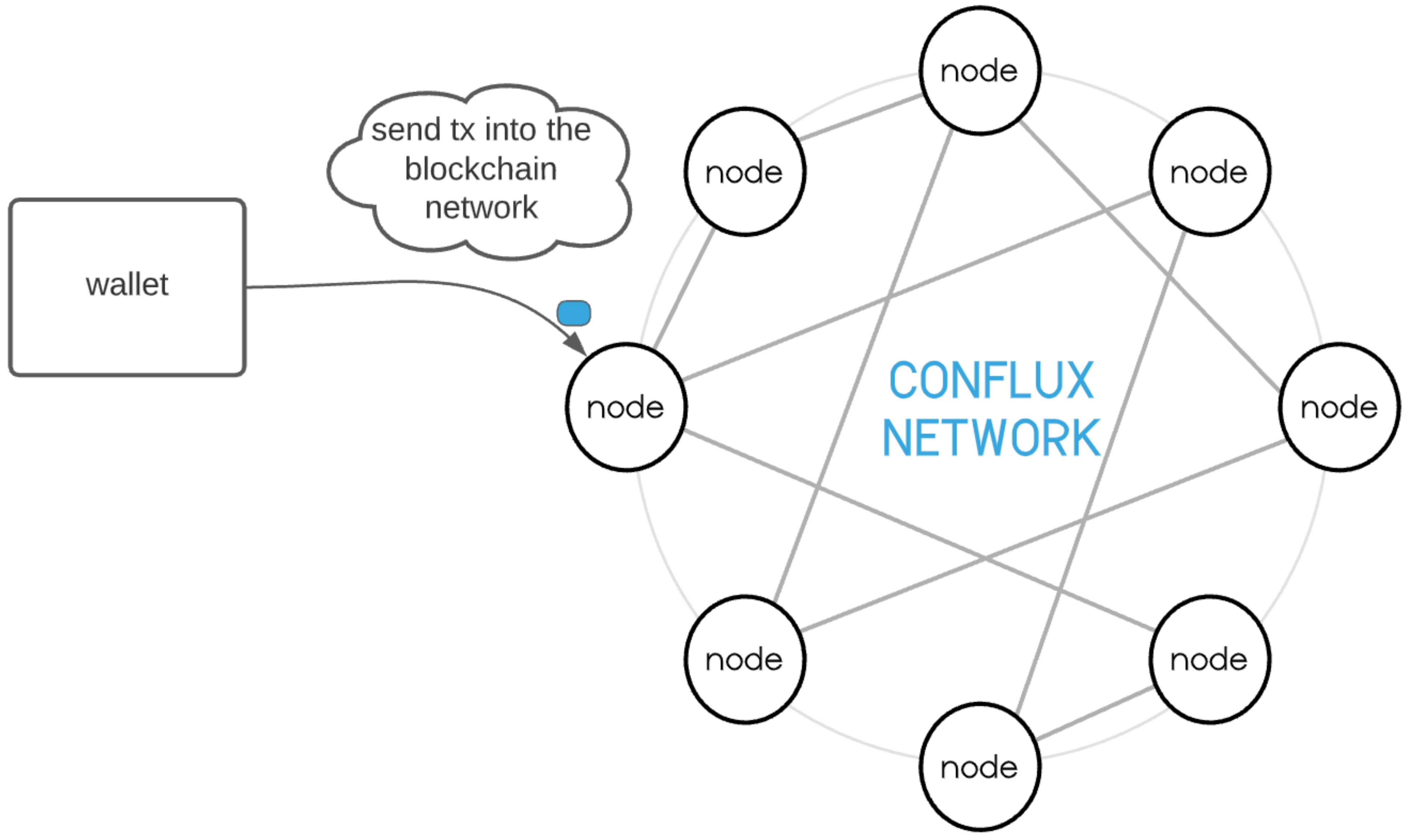
- the JSON-representation of a smart contract transaction

json
```json
{
  "from": "0x1dBDA5dD2e952914bC74a802510D0FA59F9d7636",
  "to": "0x8b017126d2fede908a86b36b43969f17d25f3770",
  "value": "0x3635c9adc5dea00000",
  "gas": "0x5208",
  "gasPrice": "0x174876e800",
  "nonce": "0x1",
  "data": "0xa6f2ae3a",
  "v": "0x1",
  "r": "0x27e5cb110dd198b8fc963d4741ec0840400a6351d9e0c458eed...",
  "s": "0x2c486d8e26da3c867fbcf4ab242af1265a5036c5e23ea42c8ab...",
}
```

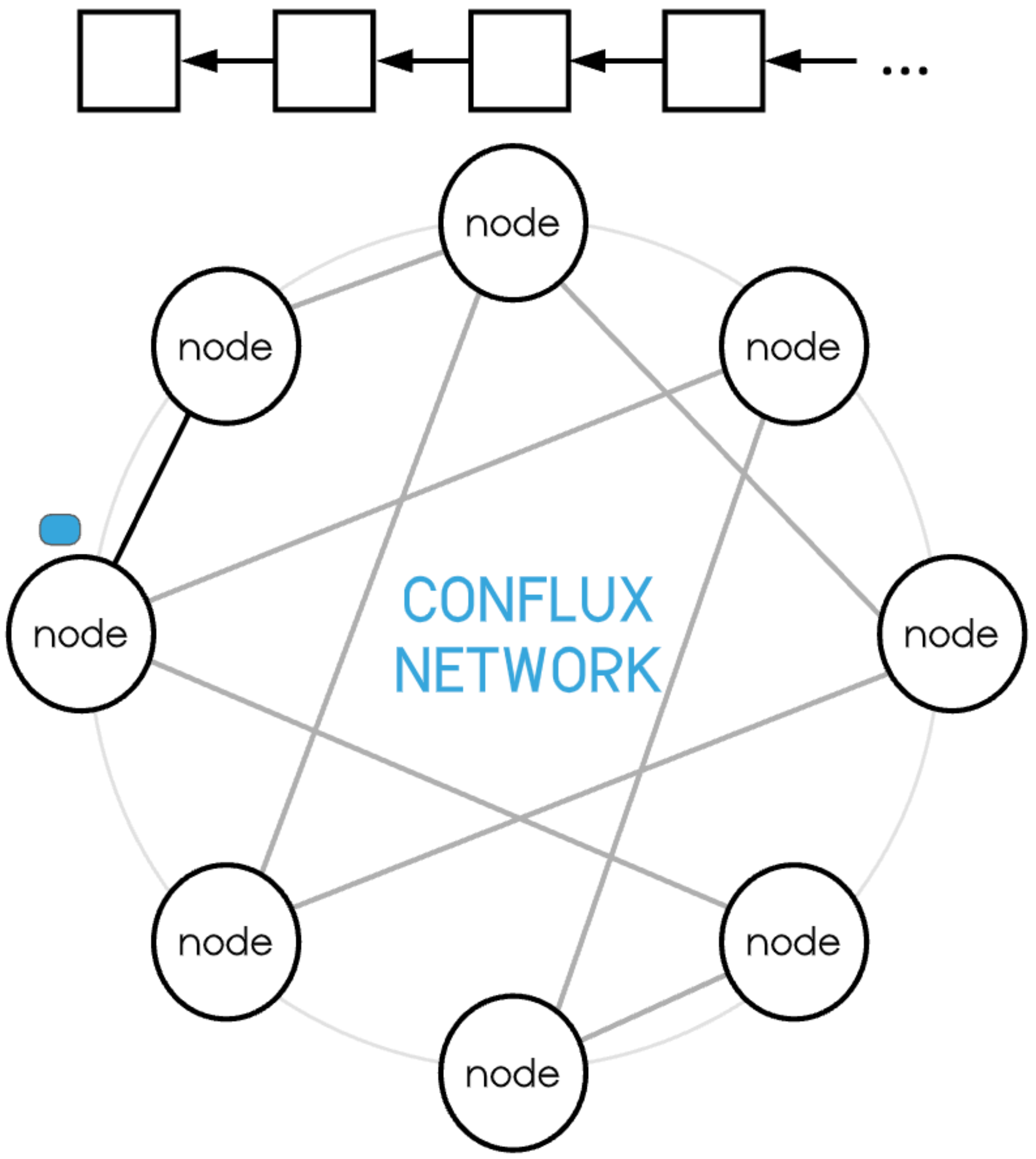
PUTTING IT ALL TOGETHER

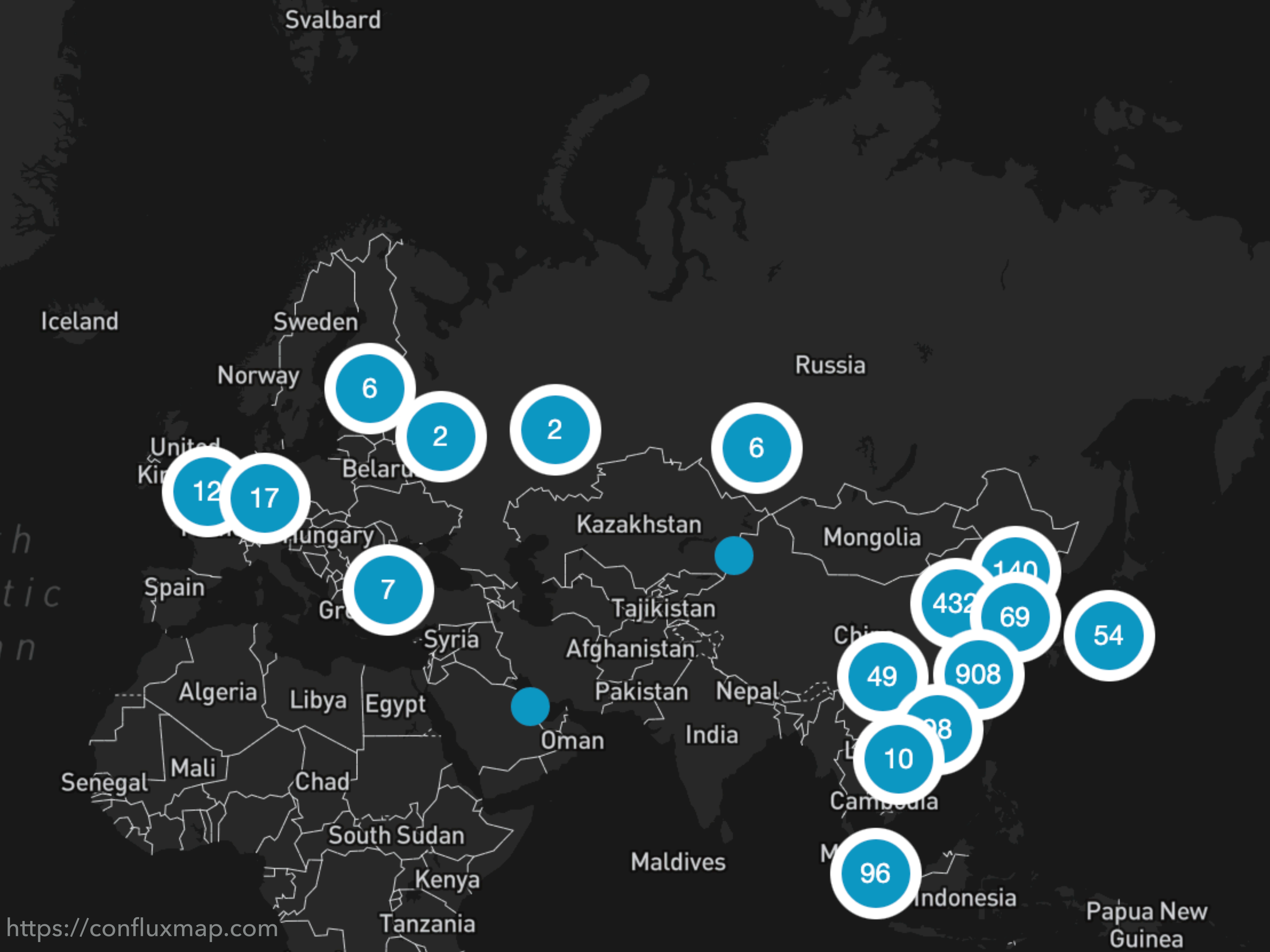


PUTTING IT ALL TOGETHER



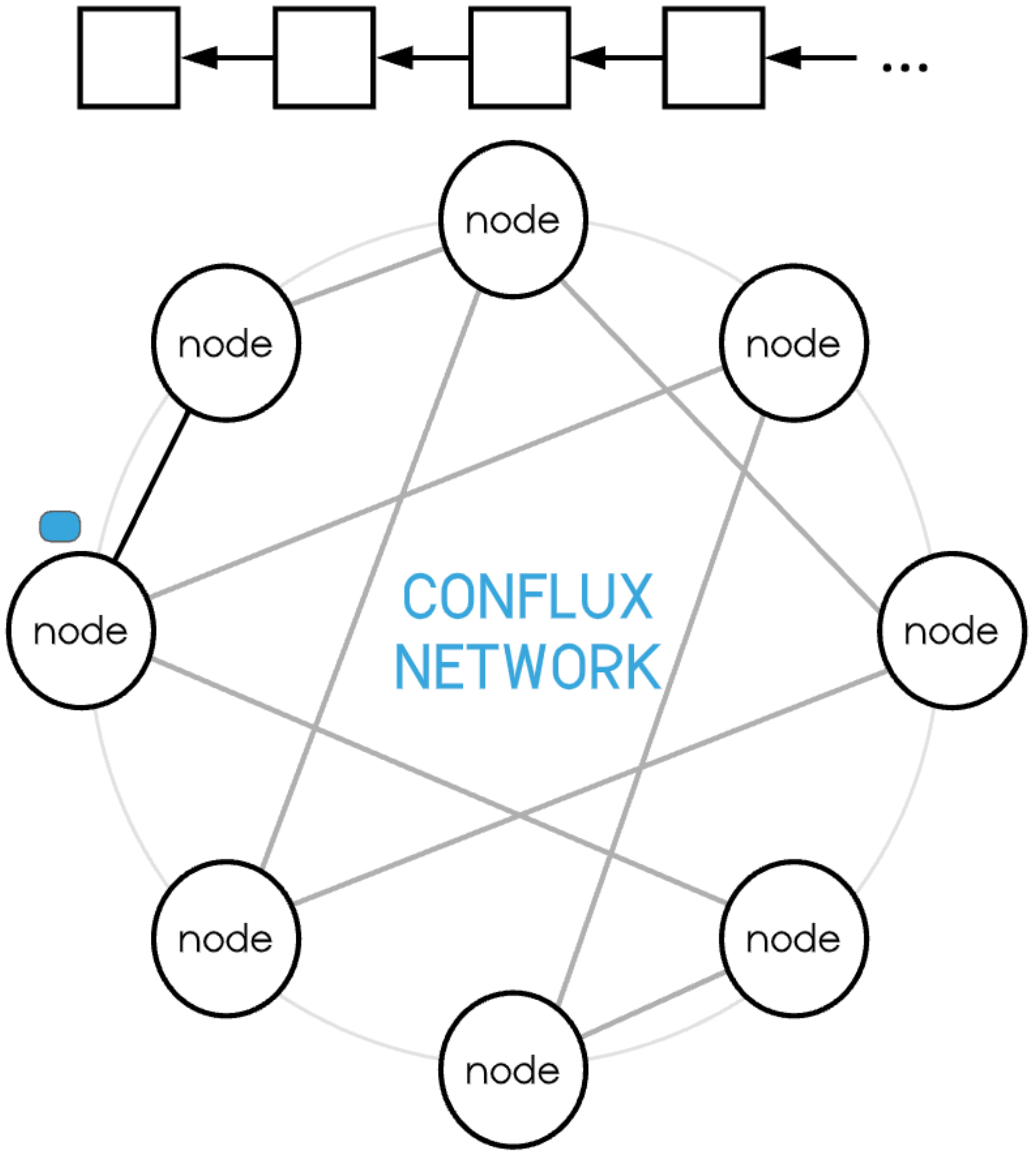
PUTTING IT ALL TOGETHER



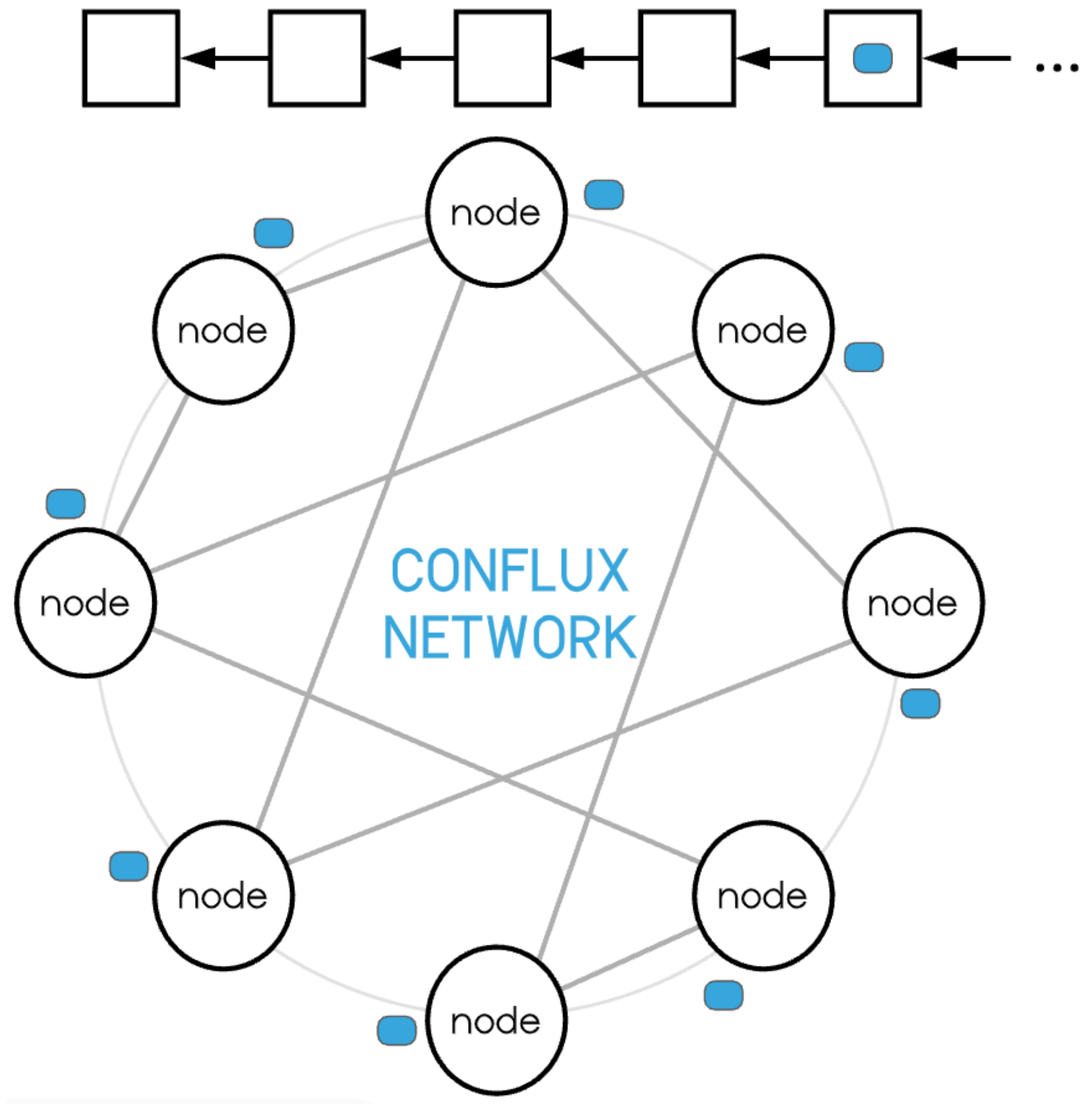


https://confluxmap.com

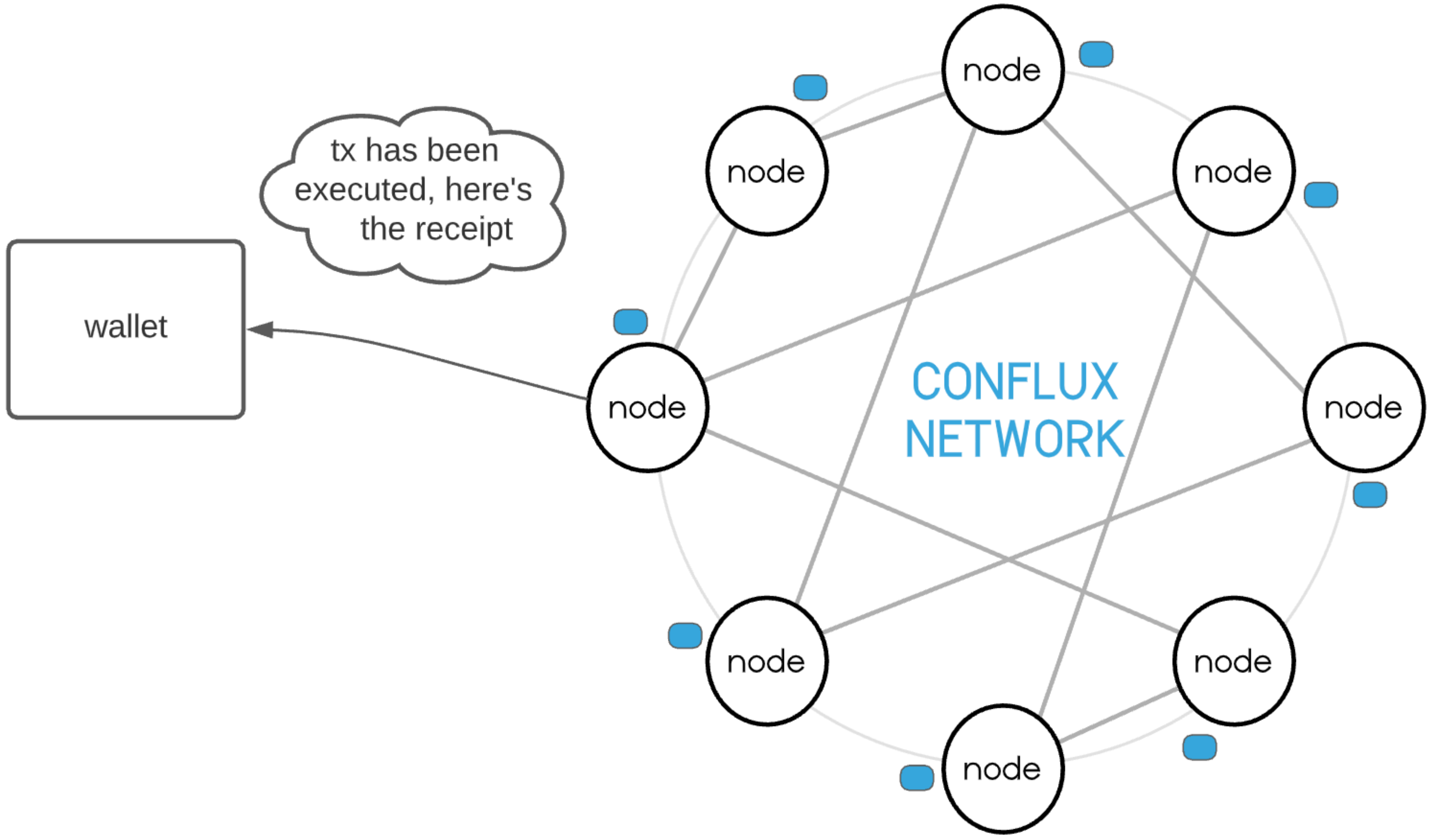
PUTTING IT ALL TOGETHER



PUTTING IT ALL TOGETHER



PUTTING IT ALL TOGETHER



TRANSACTION RECEIPT

- the JSON-representation of a receipt looks like this

json
```json
{
   "from": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
   "to": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
   "transactionHash": "0x53fe995edeec7d241791ff32635244e94ecfd7..."
   "epochNumber": 451990,
   "blockHash": "0xbb1eea3c8a574dc19f7d8311a2096e23a39f12e649a...",
   "index": 0,
   "outcomeStatus": 0,
   "contractCreated": null,
   "gasUsed": "0x5208",
   "logs": [],
   "logsBloom": "0x00000000000000000000000000000000000000000000...",
}
```

TRANSACTION RECEIPT

- transaction info

json
```json
{
   "from": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
   "to": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
   "transactionHash": "0x53fe995edeec7d241791ff32635244e94ecfd7..."
   "epochNumber": 451990,
   "blockHash": "0xbb1eea3c8a574dc19f7d8311a2096e23a39f12e649a...",
   "index": 0,
   "outcomeStatus": 0,
   "contractCreated": null,
   "gasUsed": "0x5208",
   "logs": [],
   "logsBloom": "0x000000000000000000000000000000000000000000000...",
}
```

TRANSACTION RECEIPT

- location on the ledger

```json
{
  "from": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
  "to": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
  "transactionHash": “0x53fe995edeec7d241791ff32635244e94ecfd7...”
  "epochNumber": 451990,
  "blockHash": "0xbb1eea3c8a574dc19f7d8311a2096e23a39f12e649a...",
  "index": 0,
  "outcomeStatus": 0,
  "contractCreated": null,
  "gasUsed": "0x5208",
  "logs": [],
  "logsBloom": "0x0000000000000000000000000000000000000000000...",
}
```

TRANSACTION RECEIPT

- execution results

```json
{
  "from": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
  "to": "0xb2988210c05a43ebd76575f5421ef84b120ebf80",
  "transactionHash": "0x53fe995edeec7d241791ff32635244e94ecfd7..."
  "epochNumber": 451990,
  "blockHash": "0xbb1eea3c8a574dc19f7d8311a2096e23a39f12e649a...",
  "index": 0,
  "outcomeStatus": 0,
  "contractCreated": null,
  "gasUsed": "0x5208",
  "logs": [],
  "logsBloom": "0x0000000000000000000000000000000000000000000...",
}
```

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- what are smart contracts?
- **how do I inspect data on the ledger?**
- how do I set up a wallet and send transactions?
- how do I interact with dapps?

BLOCKCHAIN EXPLORERS

- blockchain explorers are data aggregators that you can use to query accounts, transactions, blocks, etc.
- for Conflux, the official blockchain explorer is called Conflux Scan
- let's look at some examples!

USING CONFLUX SCAN (DEMO)

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- what are smart contracts?
- how do I inspect data on the ledger?
- **how do I set up a wallet and send transactions?**
- how do I interact with dapps?

WHAT DO WALLETS DO?

- a keypair represents an identity on the blockchain

  public key / address: your public identity

  private key: your authorization to act on behalf of this identity

- if you know an account's private key, you can sign transactions from that account

- storing keys securely is challenging even for experts

WHAT DO WALLETS DO?

- store private keys, create and sign transactions
- connect to a blockchain node
- send transactions into the system
- monitor transaction and account state

CONFLUX PORTAL

- Conflux Portal is the wallet we will use for this course
  portal.conflux-chain.org
- it is a browser extension available for all major browsers
- it allows you to send simple transactions or even interact with smart contracts and dapps
- Conflux Portal is based on MetaMask (Ethereum)

CREATING A WALLET

- first, we need to install the extension
- during onboarding, we can choose whether we want to import and existing wallet or create a new one
- the wallet will generate a private key and store it encrypted using a password you choose
- instead of a hex string, you'll see the private key as a mnemonic phrase -- 12 English words

CREATING A WALLET (DEMO)

WALLET SECURITY 101

- remember: if you lose your private key (seed phrase), no one will be able to recover your finds
- remember: if someone gains access to your private key, they can (and will) steal your funds
- best to backup seed phrase on a piece of paper or using a password manager
- for higher security, use a hardware wallet

HOW TO ACQUIRE TOKENS?

- get free test CFX on testnet or Oceanus through faucet
- earn CFX through mining (mining.confluxnetwork.org)
- earn FC through bounty (bounty.conflux-chain.org)
- earn FC through grants (grants.confluxnetwork.org)
- buy FC on moondex.io or other exchanges

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- what are smart contracts?
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- **how do I interact with dapps?**

INTERACTING WITH DAPPS (DEMO)

INTERACTING WITH DAPPS (DEMO)

- want to try? you can do it here! http://167.172.160.61

TOPICS WE WILL COVER TODAY

- what are accounts and transactions?
- what is a distributed ledger?
- what are smart contracts?
- how do I inspect data on the ledger?
- how do I set up a wallet and send transactions?
- how do I interact with dapps?