Blockchain

2. Decentralised applications (dapps)

Stanisław Barański
stanislaw.baranski@pg.edu.pl
https://stan.bar

15.12.2022

Agenda

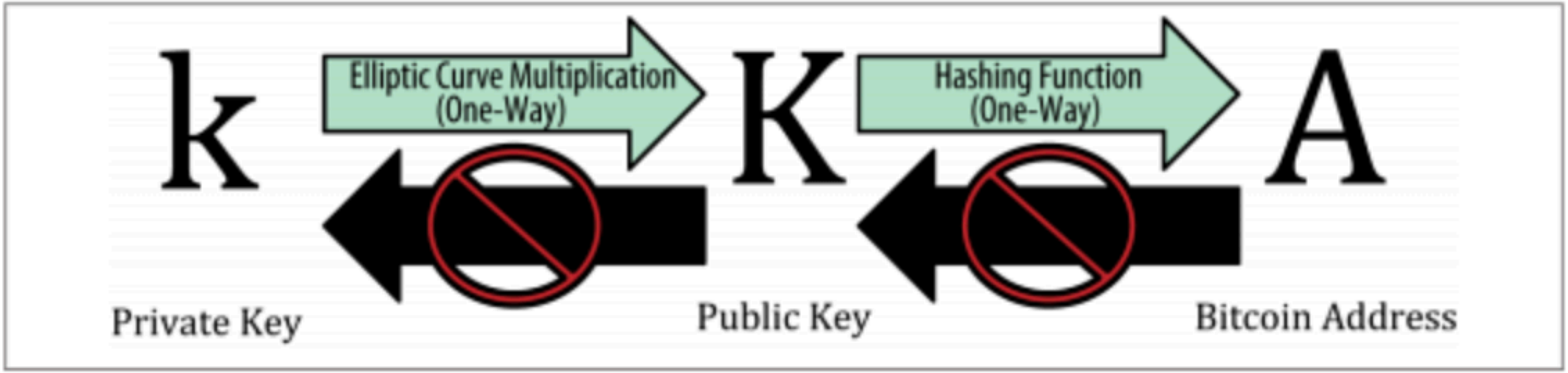
Wallets, addresses, mnemonics

How to update the state — three approaches to building dapps

Case Study: Internet Voting on blockchain

Comparison

📌Keys, Addresses, Wallets



curve: secp256k1 $y^2=x^3+7$ (mod p), where p = 2256-232-29-28-27-26-24-1

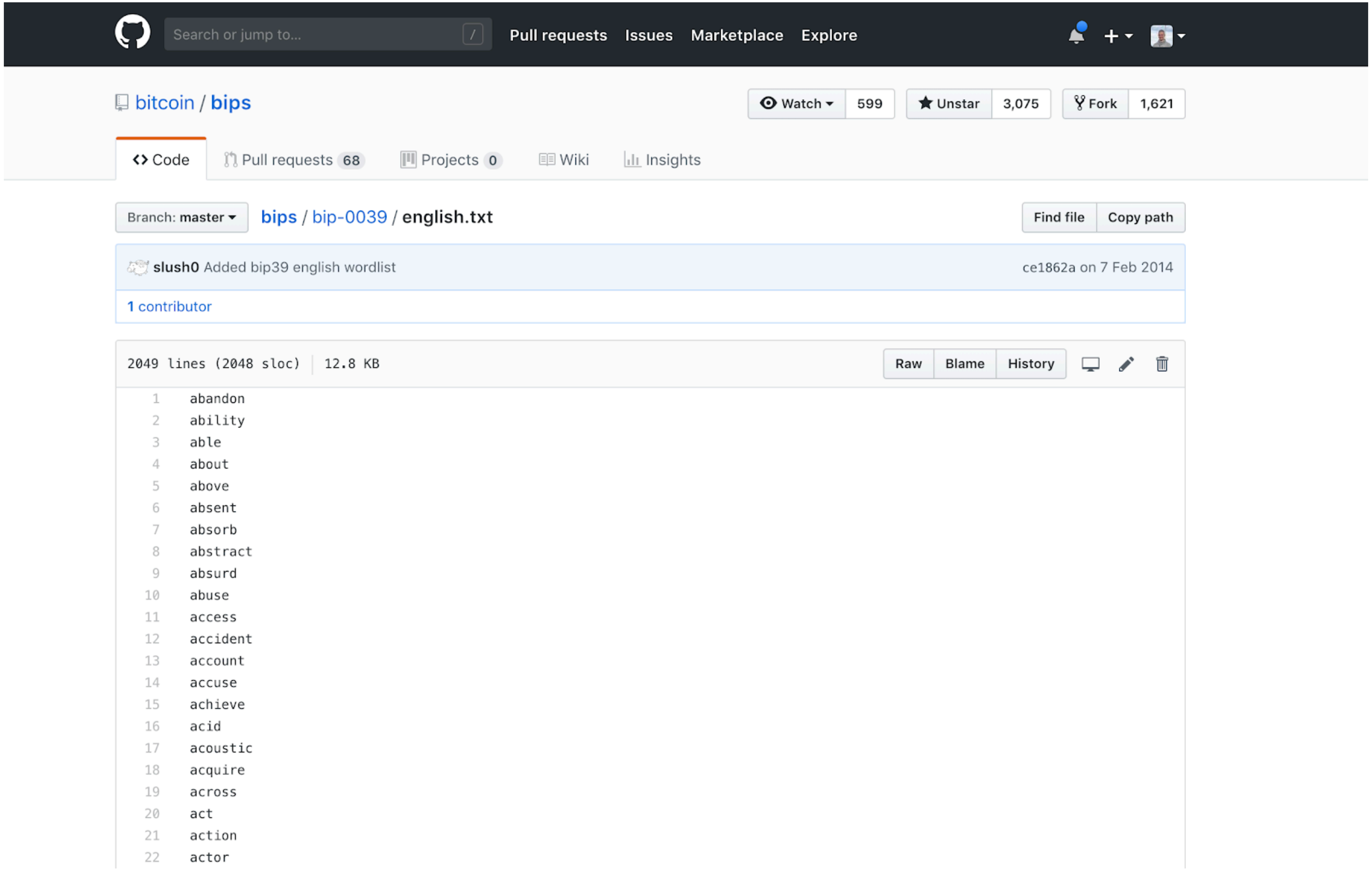
Mnemonic words

Mnemonic phrase is generated as follows:

1. Generate random sequence of 128-256 bits
2. Create checksum of the random bits by taking first 32bits of its SHA256 hash
3. Checksum is appended to the random sequence
4. Divide the sequence into sections of 11 bits, using those to index a dictionary of 2048 predefined words
5. Produce 12 or 24 words representing the mnemonic code.

https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

Mnemonic words



Hardware/paper/physical wallets

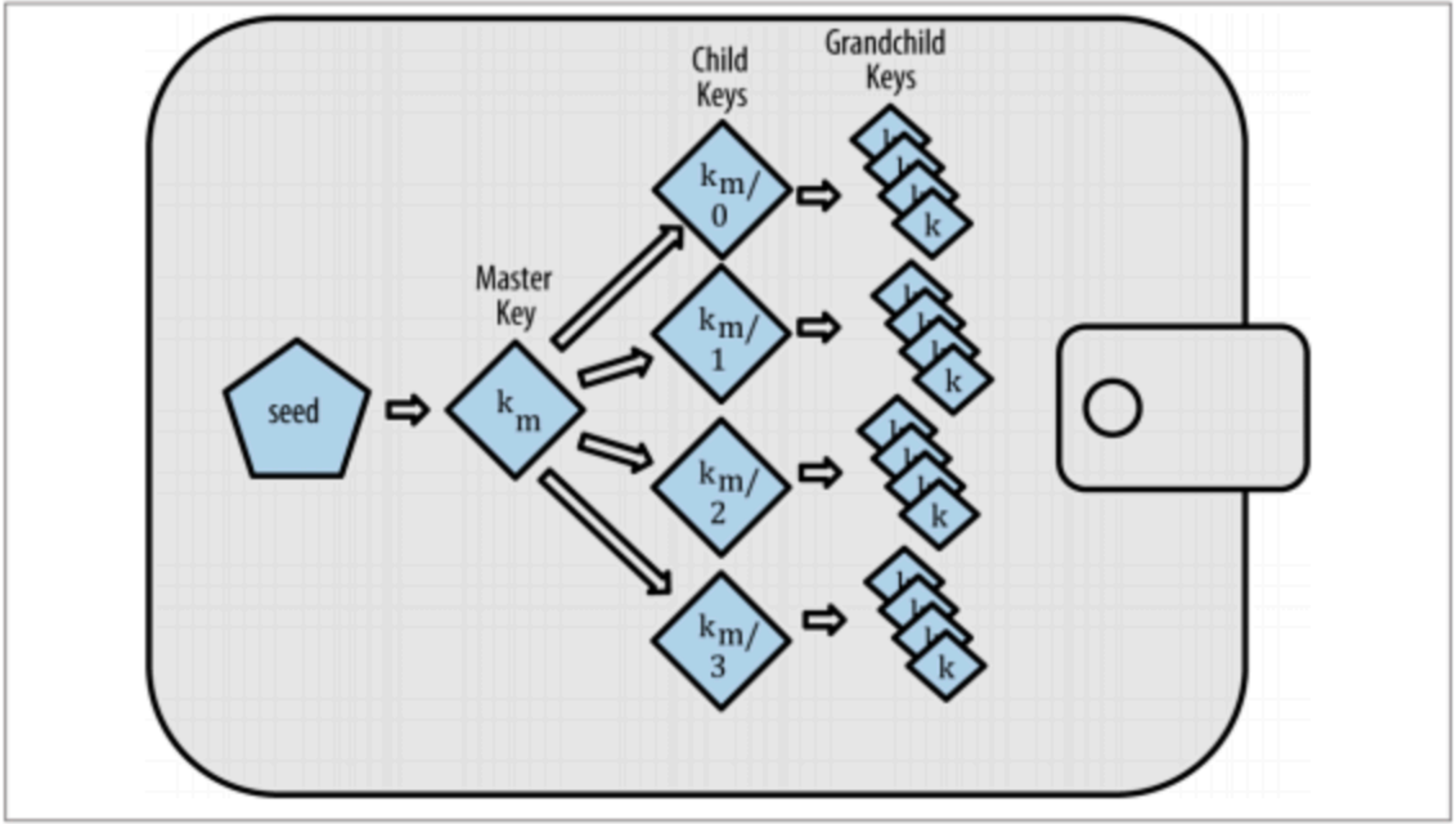
Source: https://www.thecryptomerchant.com/blogs/resources/hardware-wallet-redundancy-strategies

Deterministic wallets

Seed is generated using PBKDF2(Password-Based Key Derivation Function 2)

seed = **PBKDF2**(PRF: **HMAC-SHA512**
password: **Mnemonic phrase**, //UTF-8
Salt: **"mnemonic" + User defined password**,
iCount: **2048**,
dklen: **512**) //bits or 64 bytes

Hierarchical deterministic wallets



Source: Mastering Bitcoin by Andreas Antonopoulos https://iancoleman.io/bip39/#english

Blockchain Applications (dapp)

Moving beyond payments

Decentralised Applications (dapp)

- A blockchain application (dapp) is any kind of application which uses blockchain as a storage layer.
- Inheriting all the properties of the blockchain paradigm:
  - Decentralisation — no single entity is the owner of our data.
  - Immutability — every transaction is recorded forever on a blockchain.
  - Transparency — every transaction is publicly visible on the blockchain.
  - Verifiability — everyone can verify the correctness of the transaction.
  - Security — only valid transactions are allowed to modify the state, every node in the network validates every transaction.
  - Censorship-resistant — everyone willing to interact with the app can do it.
  - (Optional) Privacy and/or anonymity — an action made by the actor is unknown and/or an actor of the action is unknown

📌 Decentralised Applications (dapps)

There are three approaches to building a custom application in the blockchain paradigm

1. **Hack existing blockchain payment transactions** (use extra/memo field, sequence ids, addresses)

2. Non-Turing Complete **(NTC) Smart Contracts** (Stellar)

3. Turing-Complete **(TC) Smart Contracts** (EVM, WASM, etc.)

4. Create a **dedicated blockchain** (Filecoin, Chainlink, ZCash, Lisk, Substrate)

Transition State Machine

Payment Transaction

S − states

T − payment transaction

$\text{Apply} : S \times T \to S$ − state transition function

$S_{n+1} \leftarrow \text{Apply}(S_n, T_n)$

$\text{Apply}(s, t) = \{$

$\quad \text{ensure}(s[t_{from}] \geq t_{value})$

$\quad s[t_{from}] \leftarrow s[t_{from}] - t_{value}$

$\quad s[t_{to}] \leftarrow s[t_{to}] + t_{value}$

$\}$

NTC smart contracts

T - {PAYMENT, CREATE_ACC, CREATE_TOKEN, CREATE_AN_OFFER, MANAGE_DATA, etc…}

$S_{n+1} = \text{Apply}(S_n, T_n)$

$\text{Apply}(S_n, T_n) = \text{SWITCH}(S_n, T_n)$

TC smart contracts

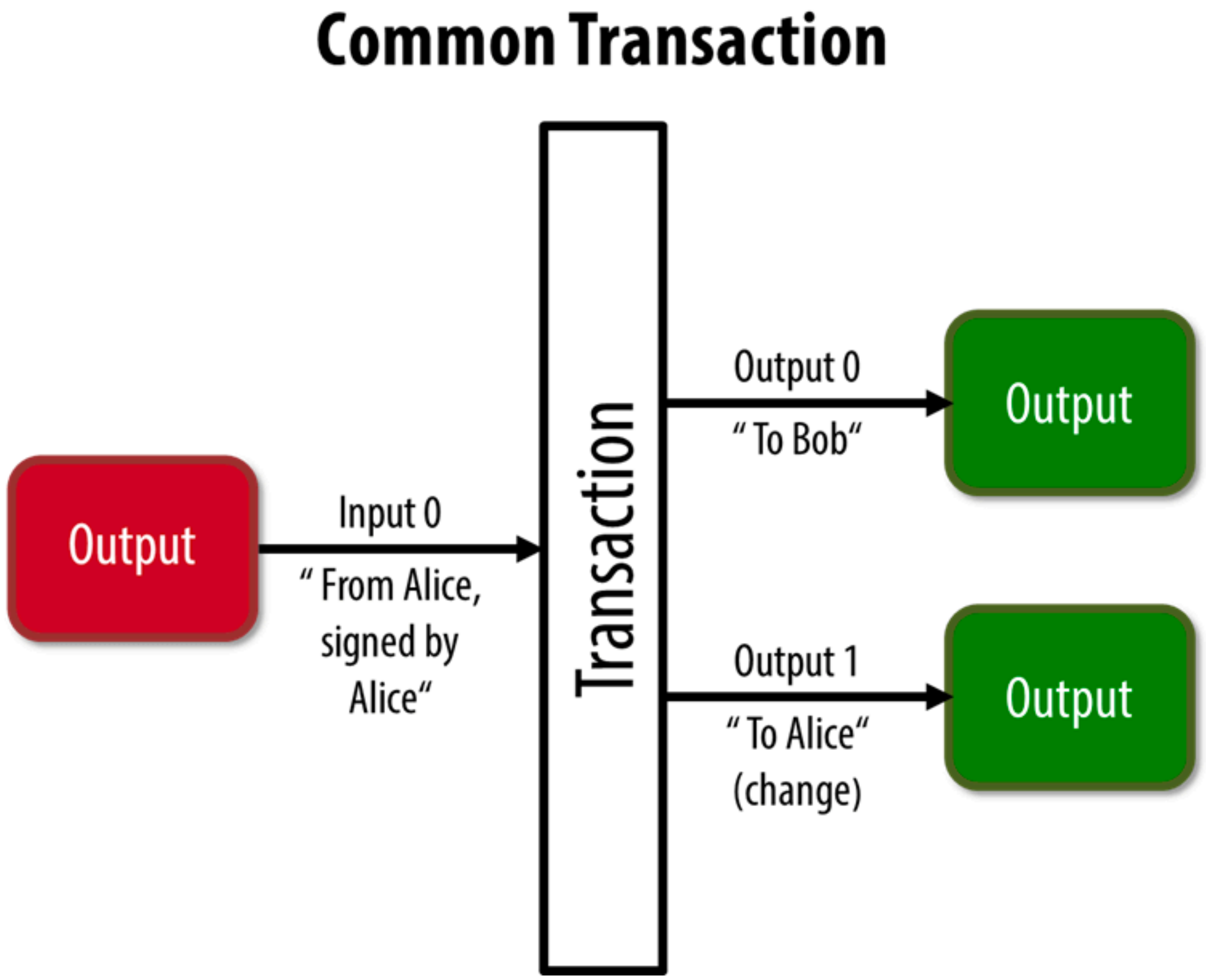
T - smart contract codes

$S_{n+1} = \text{Apply}(S_n, T_n)$

$\text{Apply}(S_n, T_n) = \text{VM}(S_n, T_n)$

Decentralised Applications

Hack existing blockchain's transactions



Source: Mastering Bitcoin by Andreas Antonopoulos

- We have the following variables: inputs, outputs, extra field (memo)
- Business logic needs to be interpreted on the client side, blockchain is just data storage.
- Examples:
  - https://proofofexistence.com/
  - Colored coins (tokens)
  - Internet voting

Proof of Existence

Hack existing blockchain's transactions

```
type Transaction = {
  inputs: Array<{address: Address, value: uint256}>;
  outputs: Array<{address: Address, value: uint256}>;
  memo: bytes256; // also called, message, extra, tag, etc.
}
```

```
type ProofOfExistence = {
  from: Address; // owner of the document
  to: Address; // registry address
  what: bytes256; // hash of the document
}
```

- https://proofofexistence.com/

Decentralised Applications

Turing-Complete (TC) Smart Contracts

- Turing-complete execution, and high expressiveness, but comes at some costs.
- FT: implement interface ERC20
- NFT: implement interface ERC721
- Number of virtual machines: EVM, WASM, Docker (HL Fabric - JVM, Go, Node.js),
- https://solidity-by-example.org/
- Business logic is encoded mostly in the smart contract — “our product is stored in the code on blockchain”
- Software-developer-friendly
- Easiest for innovative projects: ICO, Oracles, Bridges, DAOs, FT, NFTs, zkSNARKs …

- Execution time limit.
- Error-prone - risky.

Decentralised Applications
Non-Turing Complete (NTC) Smart Contracts

- Some blockchains offer a limited number of transactions
- More expressive than hacking, and less expressive than TC smart contracts.
- Limited, but often sufficient (for some domain of problems) set of operations.
- 1. Take the most promising, exciting, useful smart contracts,
- 2. Standardise them, optimise them, and
- 3. Provide them as standard operations

- Mixed business logic interpretation, both chain- and client-side.

- Stellar Operations https://developers.stellar.org/docs/fundamentals-and-concepts/list-of-operations
- Cardano Marlowe https://docs.cardano.org/marlowe/learn-about-marlowe
- Bitcoin Script https://en.bitcoin.it/wiki/Script

Decentralised Applications

Create a dedicated blockchain

- Turing-complete execution and the highest expressiveness, but it comes at some costs.
- Overcome the execution time limits.
- Great for super innovative projects that can not be executed on EVM/WASM.
- Or just a different approach than any existing Blockchain: Filecoin (PoSt), IOTA (Blockchain for IoT), Mina (super succinct BC)

- High effort to create a dedicated blockchain
- It lowers the overall security of blockchains—there is a limited amount of computing power (or any other scarce resource), and creating a new blockchain split the total hash power.

Case study: Internet voting using blockchain

Proof of Existence

Hack existing blockchain's transactions

```
type Transaction = {
  inputs: Array<{address: Address, value: uint256}>;
  outputs: Array<{address: Address, value: uint256}>;
  memo: bytes256; // also called, message, extra, tag, etc.
}
```

```
type ProofOfExistence = {
  from: Address; // owner of the document
  to: Address; // registry address
  what: bytes256; // hash of the document
}
```

- https://proofofexistence.com/

Voting protocol as a Proof of Existence: naive

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}
```

```
type Transaction = {
  inputs: Array<{address: Address, value: uint256}>;
  outputs: Array<{address: Address, value: uint256}>;
  memo: bytes256; // also called, message, extra, tag, etc.
}
```

```
const ballotBoxAddress = "0x1234";
const myAddress = "0x5678";
const voteOption = "Alice";

const myVote: Vote = {
  ballotBox: ballotBoxAddress,
  from: myAddress,
  candidate: myVoteOption,
}
```

Voting protocol as a Proof of Existence: naive

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}

const ballotBoxAddress = "0x1234";
const myAddress = "0x5678";
const voteOption = "Alice";

const myVote: Vote = {
  ballotBox: ballotBoxAddress,
  from: myAddress,
  candidate: myVoteOption,
}
```

Anonymity ❌

Privacy ❌

Voting protocol as a Proof of Existence: commit-reveal

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}
```

```
const ballotBoxAddress = "0x1234";
const myAddress = "0x5678";
const myVoteOption = "Alice";
```

```
const salt = randombytes(20);
const myVoteImproved = {
  ballotBox: ballotBoxAddress,
  from: myAddress,
  commitment: hash(voteOption + salt),
}
```

Voting protocol as a Proof of Existence: commit-reveal

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}

const ballotBoxAddress = "0x1234";
const myAddress = "0x5678";
const myVoteOption = "Alice";

const salt = randombytes(20);
const myVoteImproved = {
  ballotBox: ballotBoxAddress,
  from: myAddress,
  commitment: hash(voteOption + salt),
}
```

```
// After the end of the voting
const revealVote = {
  ballotBoxAddress: ballotBoxAddress,
  from: myAddress,
  commitment: commitment,
  candidate: voteOption.
  salt: salt,
}
```

Voting protocol as a Proof of Existence: commit-reveal

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}

const ballotBoxAddress = "0x1234";
const myAddress = "0x5678";
const myVoteOption = "Alice";

const salt = randombytes(20);
const myVoteImproved = {
  ballotBox: ballotBoxAddress,
  from: myAddress,
  commitment: hash(voteOption + salt),
}
```

```
// After the end of the voting
const revealVote = {
  ballotBoxAddress: ballotBoxAddress,
  from: myAddress,
  commitment: commitment,
  candidate: voteOption.
  salt: salt,
}
```

Problems:

- Where to publish revealVote transactions? On a blockchain?
- By revealing, we lose privacy anyway.
- Who manages the list of eligible voters?
- How to prevent multiple votes?
- Who counts the results?

Voting protocol as a Proof of Existence: asymmetric encryption

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}

const ballotBoxAddress = "0x1234";
const encryptionKey = "0x4321";

const myPrivateKey = "0x5678";
const myPublicKey = "0x9abc";
const voteOption = "Alice";

const key = DHKE(myPrivateKey, encryptionKey);
const myVoteImproved = {
  ballotBox: ballotBoxAddress,
  from: myPublicKey,
  commitment: encrypt(key, voteOption),
}
```

Voting protocol as a Proof of Existence: asymmetric encryption

Hack existing blockchain's transactions

```
type Vote = {
  from: Address;
  ballotBox: Address;
  candidate: bytes256;
}

const ballotBoxAddress = "0x1234";
const encryptionKey = "0x4321";

const myPrivateKey = "0x5678";
const myPublicKey = "0x9abc";
const voteOption = "Alice";

const key = DHKE(myPrivateKey, encryptionKey);
const myVoteImproved = {
  ballotBox: ballotBoxAddress,
  from: myPublicKey,
  commitment: encrypt(key, voteOption),
}
```

```
// After the end of the voting,
// organizer publishes the decryptionKey
// Then everyone can compute the results
const decryptionKey = "0x9876";
const results = votes.reduce((results, vote) => {
  const key = DHKE(decryptionKey, vote.from)
  const candidate = decrypt(key, vote.commitment)

  results[candidate] =
    results[candidate] ? results[candidate] + 1 : 1
  return results;
}, {})
```

Voting protocol as a Proof of Existence

Hack existing blockchain's transactions

Problems:

- Where to publish revealVote transactions? N/A ✅
- Who counts the results? Voters ✅
- How to prevent multiple votes?
- Who manages the list of eligible voters?
- By revealing, we lose privacy anyway.

Non-Turing Complete (NTC) Smart Contracts

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Issue a limited number of VOTE NFTokens (a number everyone can verify).

Transfer each VOTE token to each eligible voter (everyone can verify that on bc).

Only transactions spending VOTE tokens are counted.

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Problems:

- Who counts the results? Voters ✅
- How to prevent multiple votes? ✅
- Who manages the list of eligible voters? ✅
- By revealing, we lose privacy anyway.

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Problems:

- Who counts the results? Voters ✅
- How to prevent multiple votes? ✅
- Who manages the list of eligible voters? ✅
- By revealing, we lose privacy anyway.
- Organisers know the address of each eligible voter, they can link their identity with their address and hence, their vote option.

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Split voting into two untrackable stages:

1. Authentication

2. Authorization

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Split voting into two untrackable stages:

1. Authentication

2. Authorization

https://stellot.com

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Problems:

- Who counts the results? Voters ✅
- How to prevent multiple votes? ✅
- Who manages the list of eligible voters? ✅
- By revealing, we lose privacy anyway ✅
- Organisers know the address of each eligible voter, they can link their identity with their address and hence, their vote option ✅

Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Problems:

- Who counts the results? Voters ✅
- How to prevent multiple votes? ✅
- Who manages the list of eligible voters? ✅
- By revealing, we lose privacy anyway ✅
- Organisers know the address of each eligible voter, they can link their identity with their address and hence, their vote option ✅
- How to prevent bribing?
- How to prevent organisers from decrypting the votes before the end of voting?

Turing Complete (NTC) Smart Contracts

Voting protocol

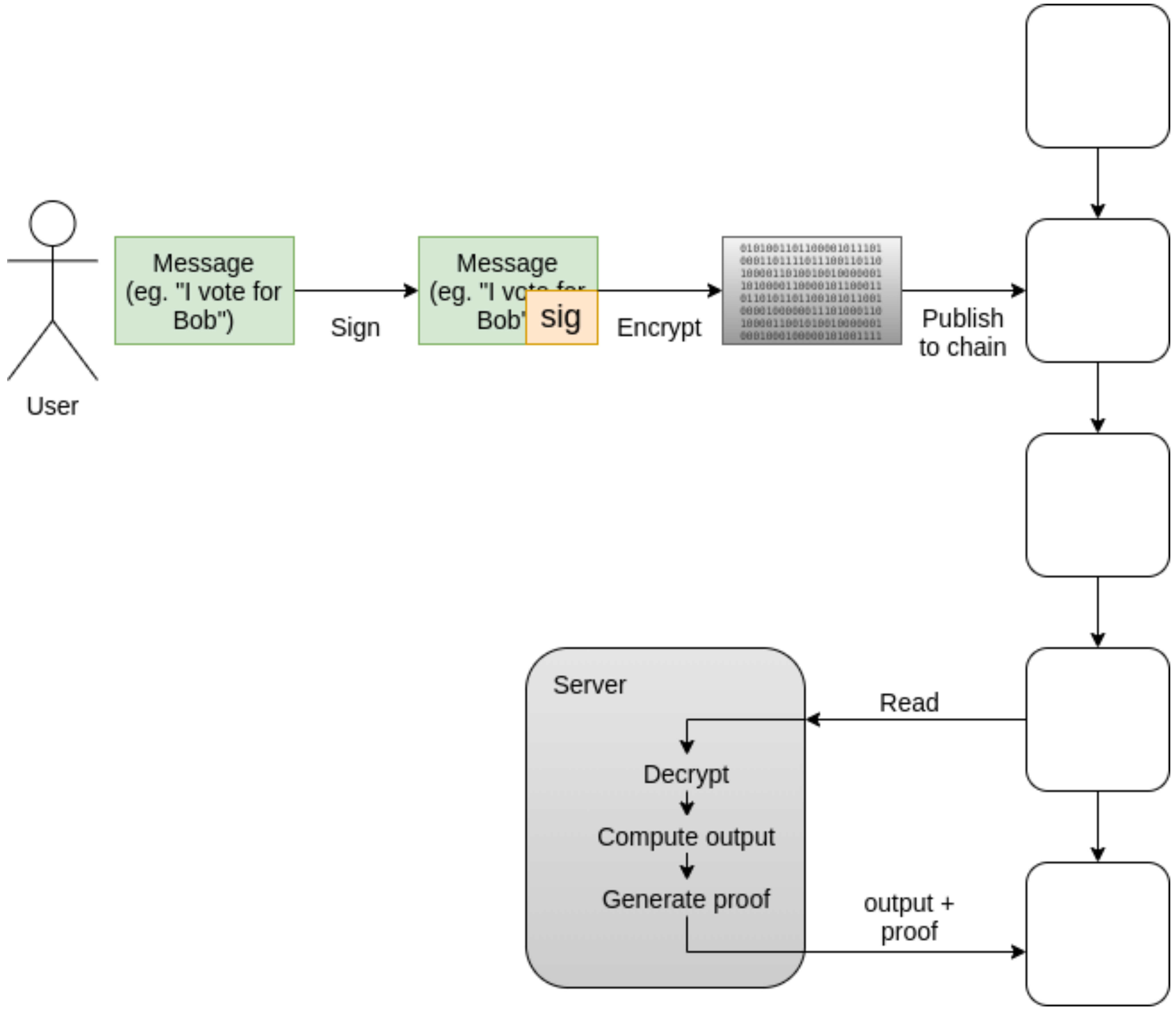
Bribing resistance

Allow casting multiple votes, each time optionally allowing for invalidating the previous one, in such a way that no one can tell which one is valid; therefore, it can not be proven to the briber.

Minimum Anti-Collusion Infrastructure (MACI)

https://github.com/privacy-scaling-explorations/maci/tree/master/specs

https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413



Voting protocol

Non-Turing Complete (NTC) Smart Contracts

Problems:

- Who counts the results? Voters ✅
- How to prevent multiple votes? ✅
- Who manages the list of eligible voters? ✅
- By revealing, we lose privacy anyway ✅
- Organisers know the address of each eligible voter, they can link their identity with their address and hence, their vote option ✅
- How to prevent bribing? ✅
- How to prevent organisers from decrypting the votes before the end of voting?
- Organiser? Is is still dapp?

Dedicated Blockchain

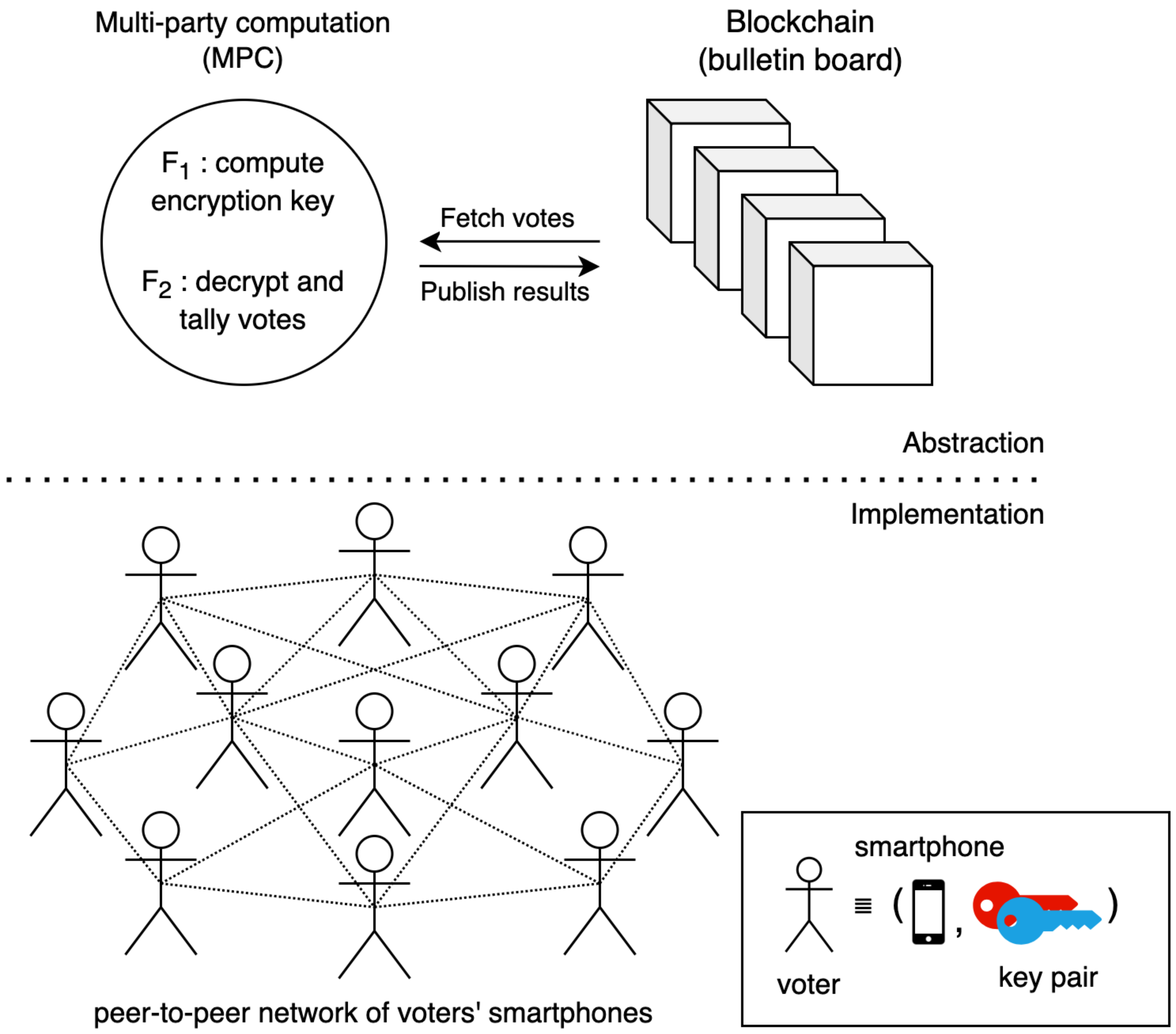
Voting protocol

Turing Complete (NTC) Smart Contracts

Voters generate encryption key using Distributed key generation (DKG) or Shamir Secret Sharing (SSS).

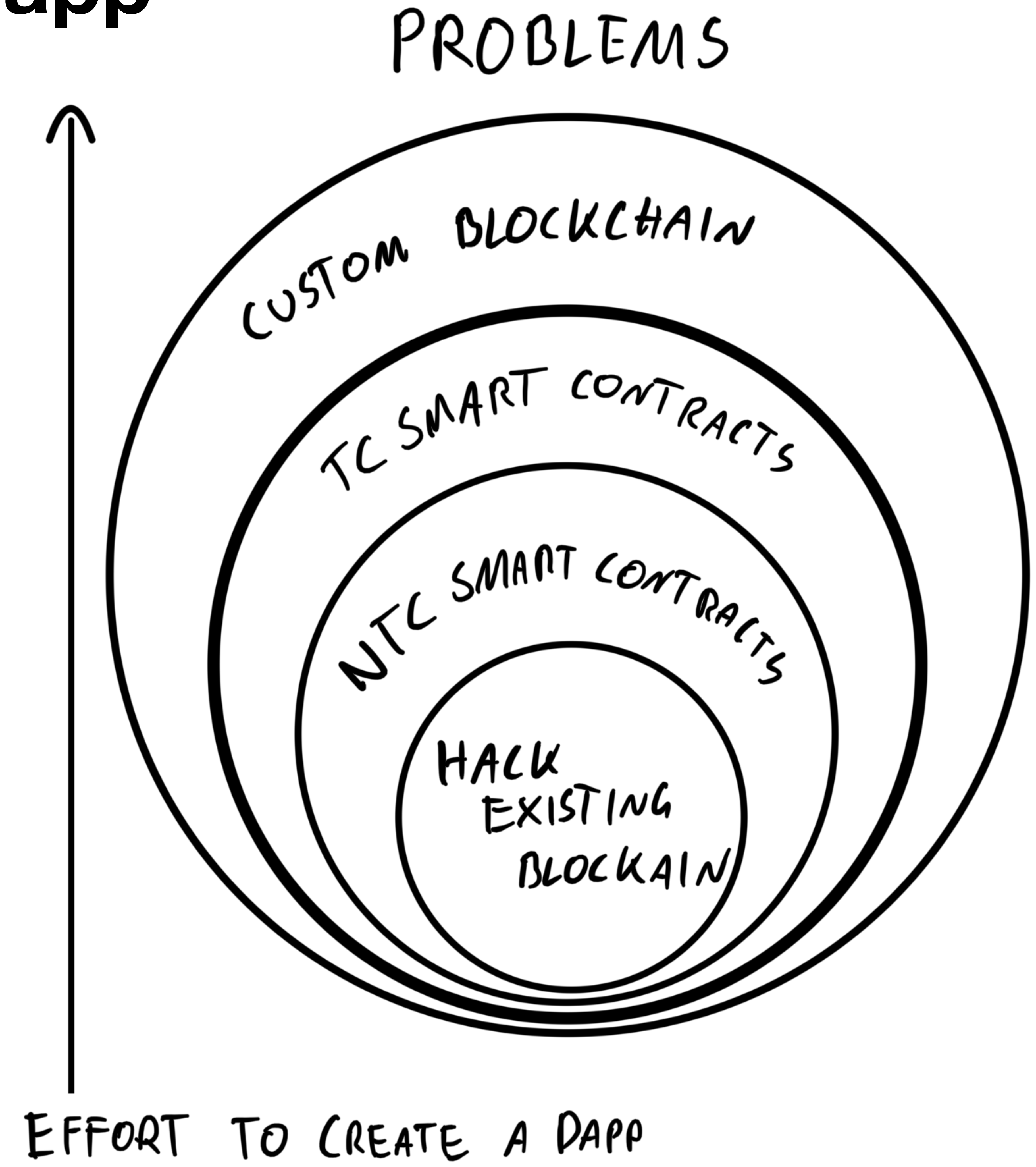
Get rid of the organiser (semi-trusted-third party).

Voters participate in MPC protocol to compute calculations on encrypted data



Decentralised Applications

Effort to create a dapp



Comparison

	Hacking Existing BC	NTC Smart Contract	TC Smart Contract	Dedicated BC
Limits	A few variables of constant type	Limited number of operations	8 sec of execution and costly	Unlimited
Fee	Low and constant	Low and constant	High and vary on execution	Custom
Expressivity	Low	Medium	High	Unlimited
Interpretation	Client-side	Both client- and chain-side	Chain-side	Chain-side
Effort to create	Low	Medium	High	Enormous
Platforms	All blockchains	Bitcoin, Stellar, Cardano	EVM (Ethereum, NEAR/Aurora, BSC), WASM (NEAR, Solana), Docker(HL Fabric), Cardano, Aleo, Wasm	DIY, Fork, Substrate, Exonum
Languages	N/A	Bitcoin’s Script, Stellar OPS, Marlowe	Solidity, Vyper (Python), Plutus (Haskell), TS, Go, Java, **Rust**, C, Go, Java, Leo	**Rust**, go, C++
Example applications	{Proof of existance, Colored Coins}	∪ {escrow, multisigs, payment channels, stable coins, DeFi, DEX, Internet voting}	∪ {zkSNARKs, DEX+, Gambling,TornadoCash}	Filecoin, Golem, StorJ, zkSync, StarkNet, and other side-chains

Conclusions

- Try to formulate your problem to fit the standard blockchain transaction (like proof of existence).
- If it’s hard, troublesome, or impossible then move to NTC smart contract.
- If it’s hard, troublesome, or impossible then move to TC smart contract.
- If it’s too expensive, or too slow or does not meet your trust assumptions create a dedicated blockchain.
- Similar to building a mobile app: web app, multi-platform app, then native apps.

Rate the lecture from 0 to 5

- Go to https://stellot.com/#/voting/rate-the-lecture-from-0-to-5-qk3nuv
- Rate the lecture *anonymously* on #blockchain

Questions?

Stanisław Barański
stanislaw.baranski@pg.edu.pl
https://stan.bar