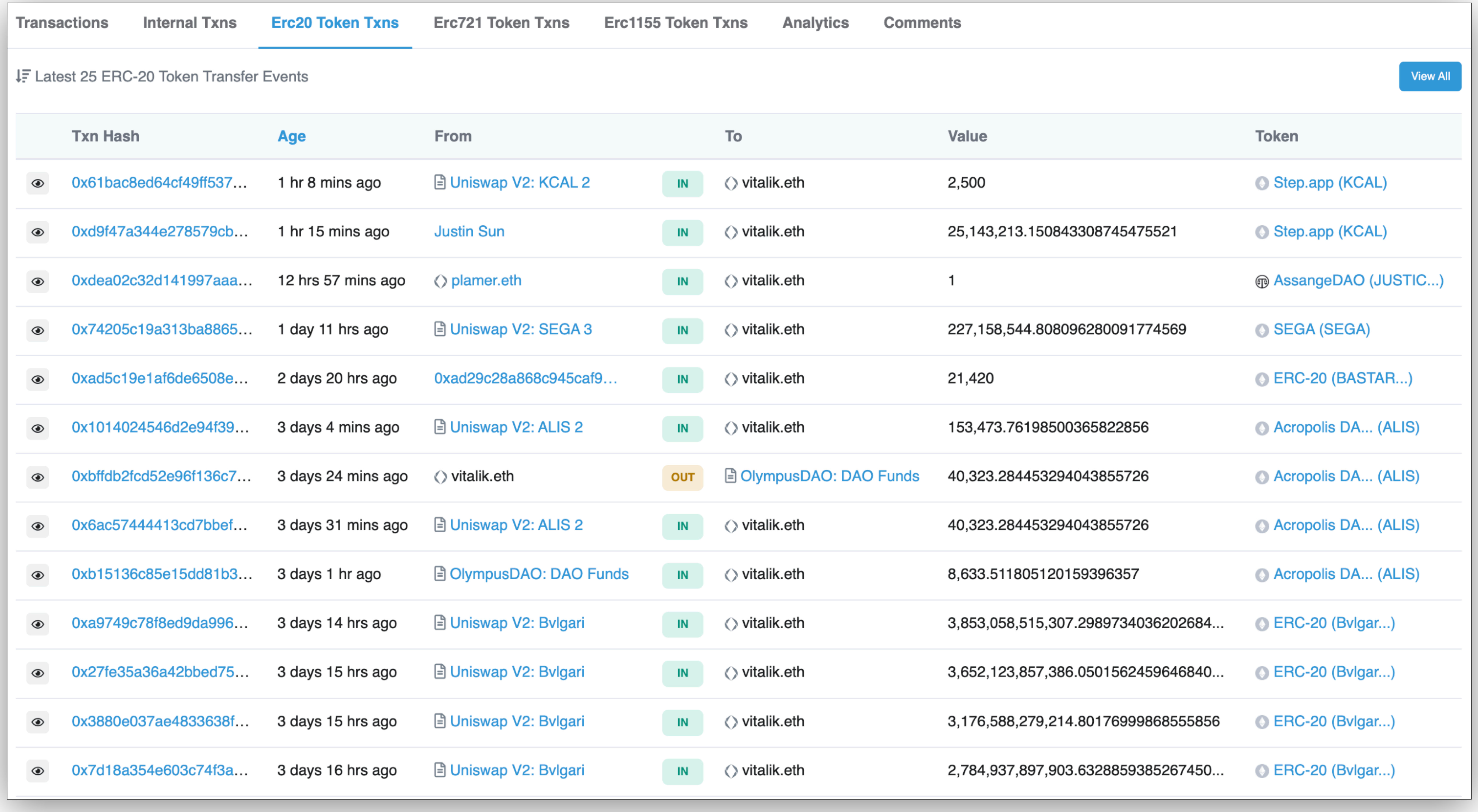
**Morten Dahl**
**Senior Blockchain Director**

# fhEVM

## Confidential Smart Contracts Using Homomorphic Encryption

ZAMA

# Everything on a blockchain is public



Latest 25 ERC-20 Token Transfer Events

| Txn Hash | Age | From | | To | Value | Token |
|---|---|---|---|---|---|---|
| 0x61bac8ed64cf49ff537… | 1 hr 8 mins ago | Uniswap V2: KCAL 2 | IN | vitalik.eth | 2,500 | Step.app (KCAL) |
| 0xd9f47a344e278579cb… | 1 hr 15 mins ago | Justin Sun | IN | vitalik.eth | 25,143,213.150843308745475521 | Step.app (KCAL) |
| 0xdea02c32d141997aaa… | 12 hrs 57 mins ago | plamer.eth | IN | vitalik.eth | 1 | AssangeDAO (JUSTIC...) |
| 0x74205c19a313ba8865… | 1 day 11 hrs ago | Uniswap V2: SEGA 3 | IN | vitalik.eth | 227,158,544.808096280091774569 | SEGA (SEGA) |
| 0xad5c19e1af6de6508e… | 2 days 20 hrs ago | 0xad29c28a868c945caf9… | IN | vitalik.eth | 21,420 | ERC-20 (BASTAR...) |
| 0x1014024546d2e94f39… | 3 days 4 mins ago | Uniswap V2: ALIS 2 | IN | vitalik.eth | 153,473.761985003658228 56 | Acropolis DA... (ALIS) |
| 0xbffdb2fcd52e96f136c7… | 3 days 24 mins ago | vitalik.eth | OUT | OlympusDAO: DAO Funds | 40,323.284453294043855726 | Acropolis DA... (ALIS) |
| 0x6ac57444413cd7bbef… | 3 days 31 mins ago | Uniswap V2: ALIS 2 | IN | vitalik.eth | 40,323.284453294043855726 | Acropolis DA... (ALIS) |
| 0xb15136c85e15dd81b3… | 3 days 1 hr ago | OlympusDAO: DAO Funds | IN | vitalik.eth | 8,633.511805120159396357 | Acropolis DA... (ALIS) |
| 0xa9749c78f8ed9da996… | 3 days 14 hrs ago | Uniswap V2: Bvlgari | IN | vitalik.eth | 3,853,058,515,307.2989734036202684... | ERC-20 (Bvlgar...) |
| 0x27fe35a36a42bbed75… | 3 days 15 hrs ago | Uniswap V2: Bvlgari | IN | vitalik.eth | 3,652,123,857,386.0501562459646840... | ERC-20 (Bvlgar...) |
| 0x3880e037ae4833638f… | 3 days 15 hrs ago | Uniswap V2: Bvlgari | IN | vitalik.eth | 3,176,588,279,214.80176999868555856 | ERC-20 (Bvlgar...) |
| 0x7d18a354e603c74f3a… | 3 days 16 hrs ago | Uniswap V2: Bvlgari | IN | vitalik.eth | 2,784,937,897,903.6328859385267450... | ERC-20 (Bvlgar...) |

# This leads to many privacy issues

## THEFT

Criminals know what you own, so they can easily target you and steal your crypto.

## SURVEILLANCE

Governments can surveil you, even if you use multiple addresses.

## MEV

Bots can front-run you, creating a hidden tax on every transaction.

# Fully Homomorphic Encryption (FHE) enables encrypted data processing

$$E[x] + E[y] = E[x + y]$$

$$E[x] < E[y] = E[x < y]$$

More generally:

$$f(E[x], \ldots, E[y]) = E[f(x, \ldots, y)]$$

# Zama's fhEVM enables confidential smart contracts using homomorphic encryption

**E2E encryption of transactions and state**

**Composability and data availability on-chain**

**No impact on existing dapps and state**

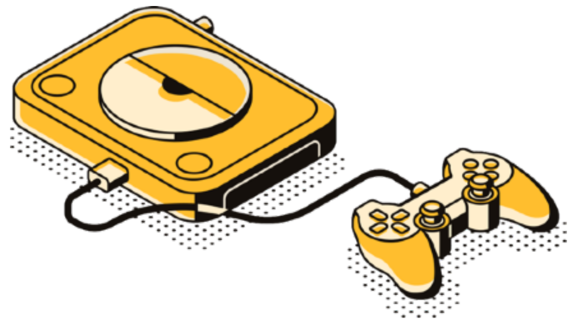
# The fhEVM unlocks a myriad of use cases

### Tokenization

Manage and swap tokenized assets without other seeing it

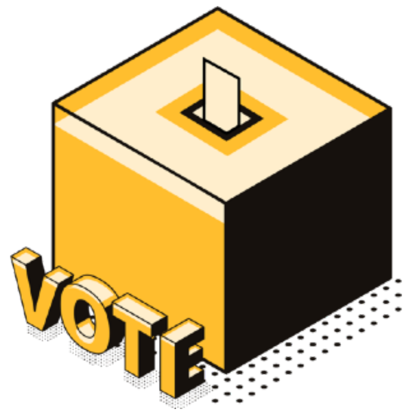
### Blind Auctions

Bid on items without revealing the amount or the winner



### On-chain Games

Hide cards and moves until reveal (e.g. poker, blackjack, ..)

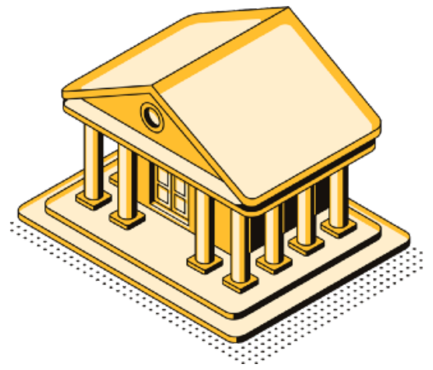


### Confidential Voting

Prevents bribery and blackmailing by keeping votes private

### Encrypted DIDs

Store identities on-chain and generate attestations without ZK



### Private Transfers

Keep balances and amounts private, without using mixers

# Zama's fhEVM is the most comprehensive confidential smart contract solution

| | Zama fhEVM | Other FHE | ZK | Mixers | SGX |
|---|---|---|---|---|---|
| **Operations supported** | Everything | Additions & multiplications | AND & XOR | None | Everything |
| **Privacy Model** | Hides the data | Hides the data | Hides the data | Hides the identity | Hides the data |
| **Data Availability** | On-chain | On-chain | Off-chain | On-chain | On-chain |
| **Encrypted state composability** | Yes | Limited | No | No | Yes |
| **On-chain PRNG** | Yes | No | No | No | Yes |
| **Developer Experience** | Easy | Medium | Hard | Hard | Easy |
| **Compliance** | At the application level | At the user level | At the user level | At the user level | At the application level |
| **Security** | Proven secure | No security proof | Proven secure | Proven secure | Broken |

# Powerful features are available out of the box

### High Precision Integers

Up to 256 bits of precision for integers

### Full range of Operators

All typical operators are available: +,-,*,/,<,>,==,…

### Encrypted If-Else Conditionals

Check conditions on encrypted states

### On-chain Secure Randomness

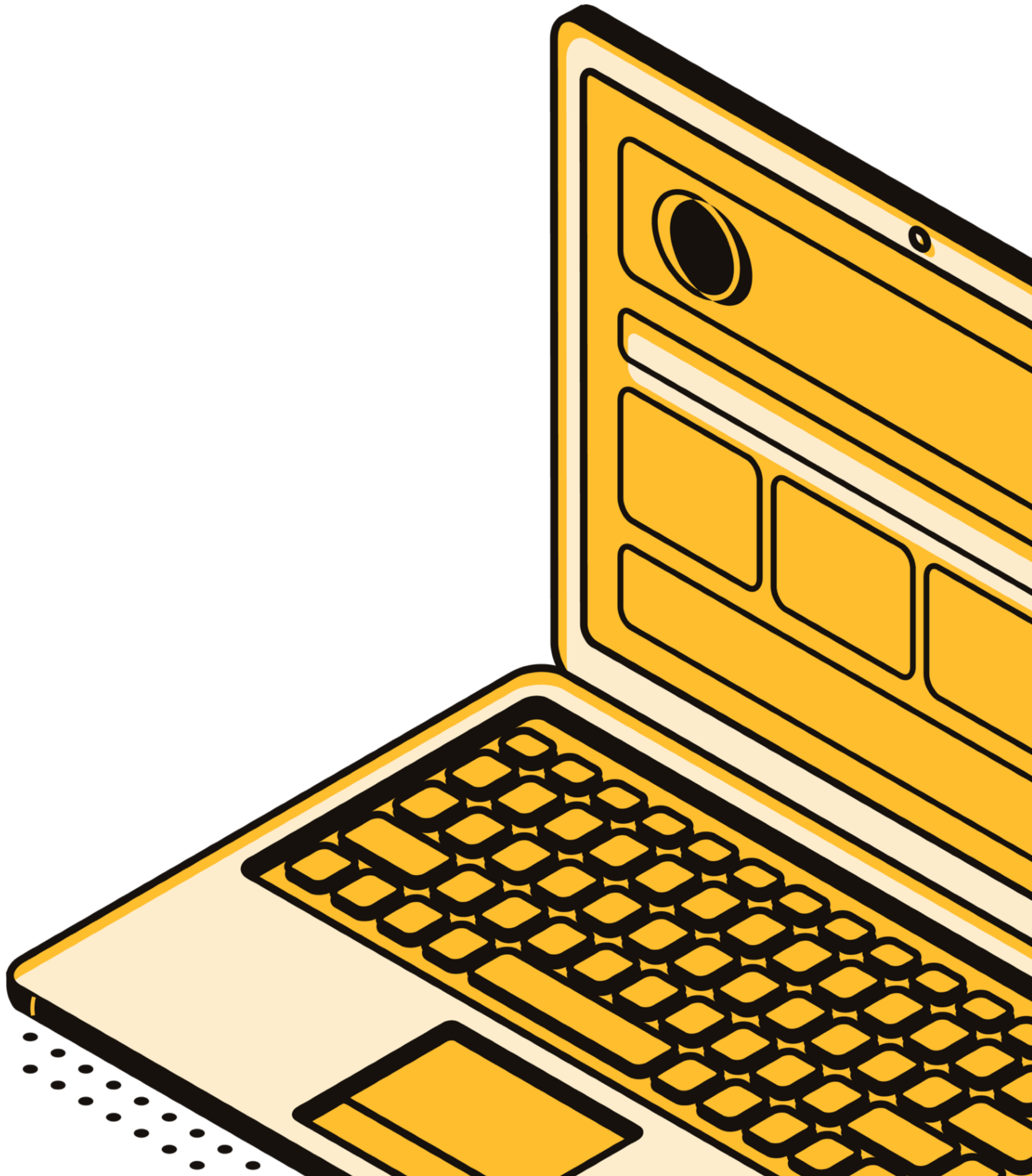
Generate randomness without using oracles

### Configurable Decryption

Threshold, centralized or KMS decryption

### Unbounded Compute Depth

Unlimited consecutive FHE operations



# fhEVM: How It Works

# Zama's fhEVM combines state of the art cryptography in a provably secure way

**FHE** + **MPC** + **ZK**

**FHE**

Homomorphic encryption is used to compute on private state, directly on-chain

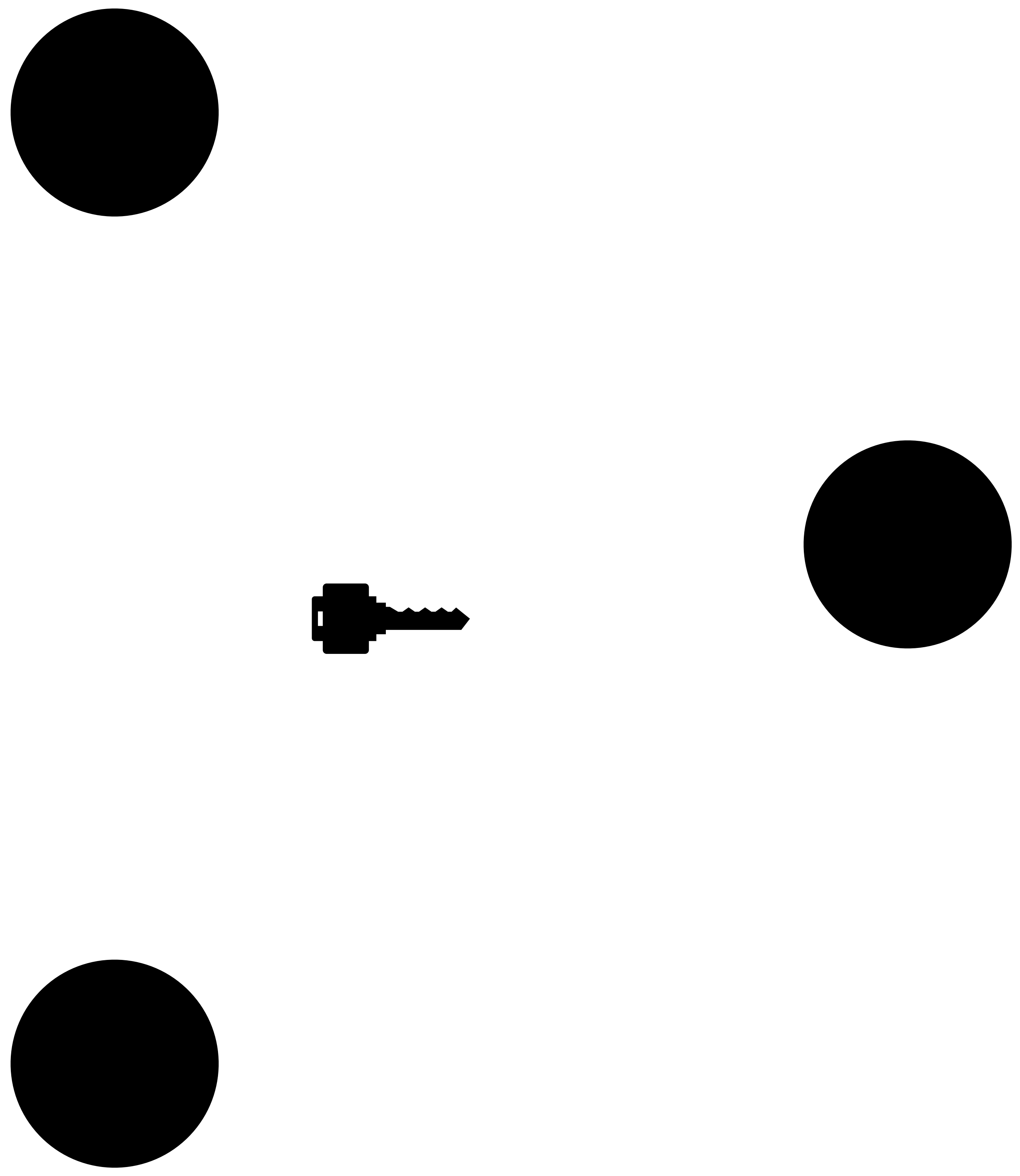
**MPC**

Multi-party computation is used for threshold decryption of FHE ciphertexts

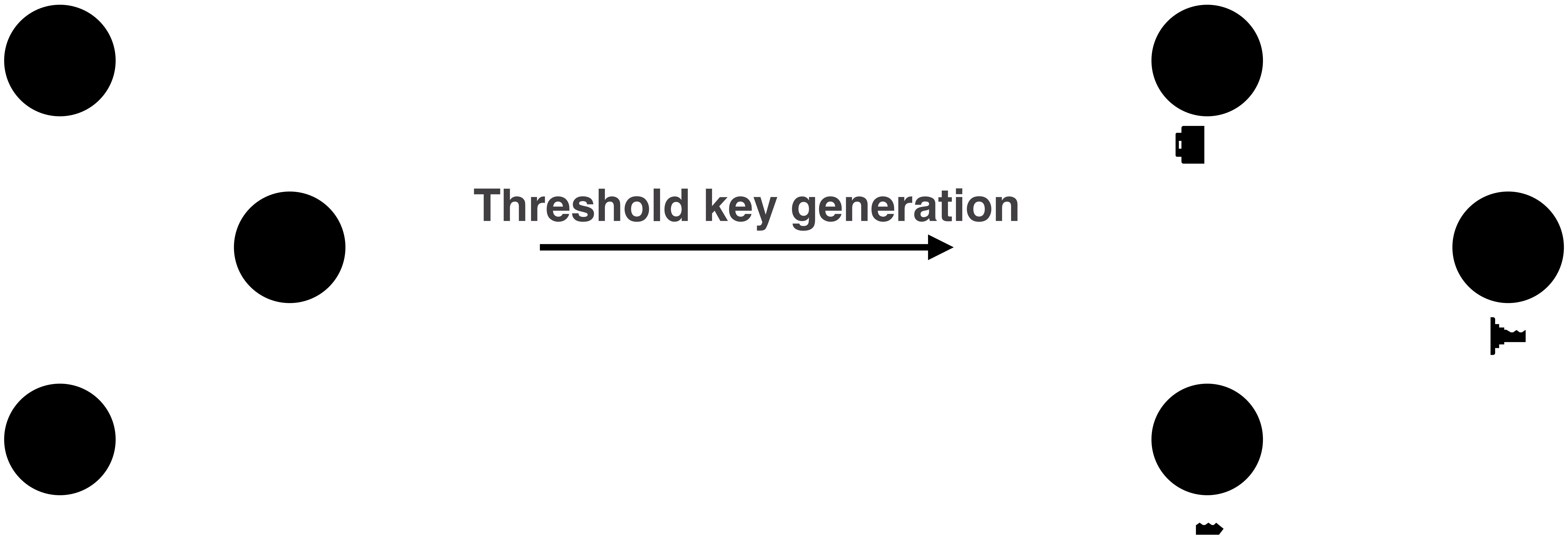
**ZK**

Zero-Knowledge Proofs of Knowledge are used to ensure encryption and decryption integrity

# Everything is encrypted under single global FHE public key

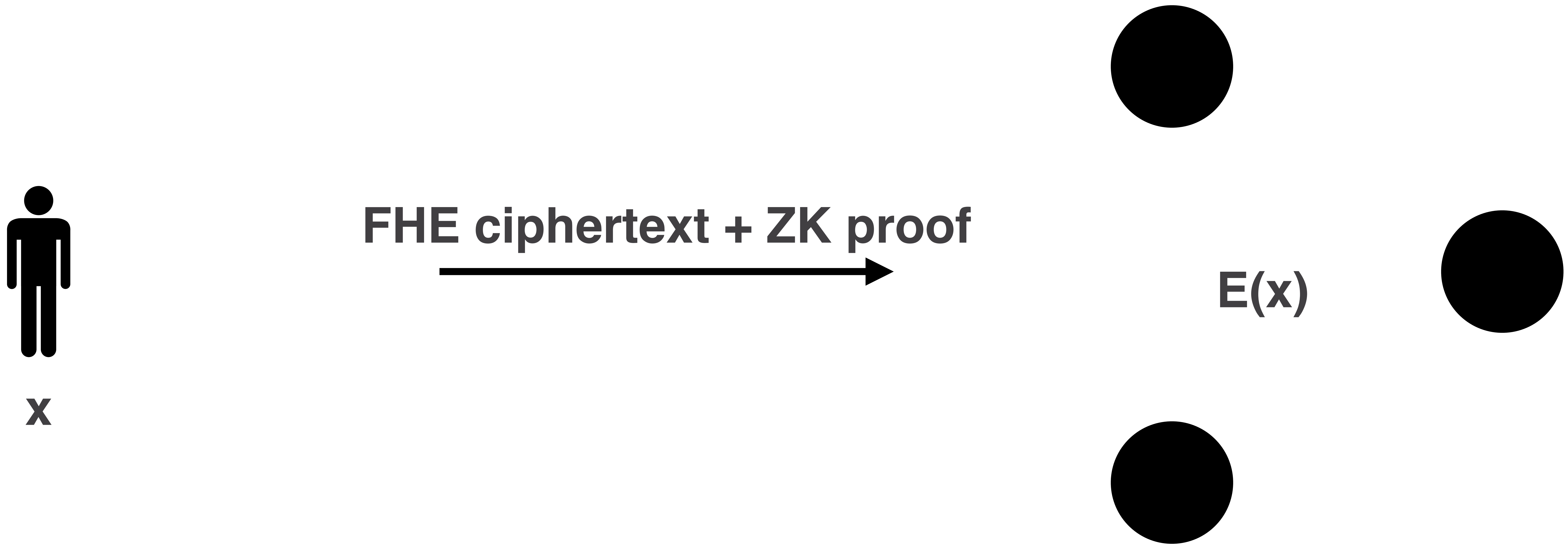


# The global key is generated securely using a threshold protocol

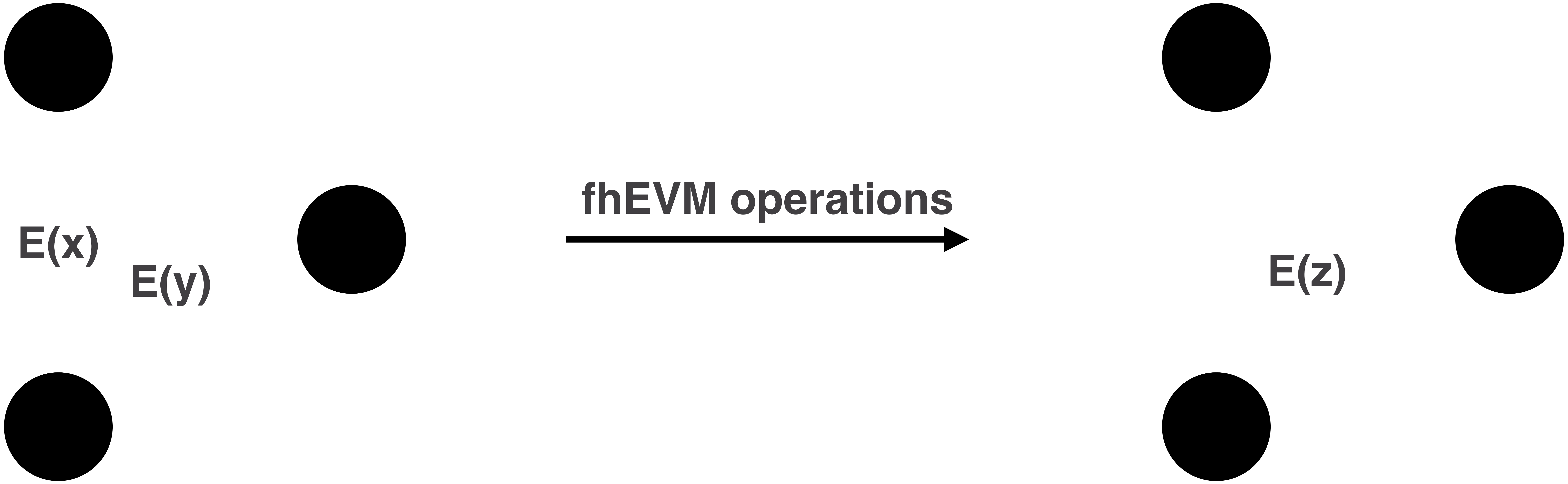


Threshold key generation

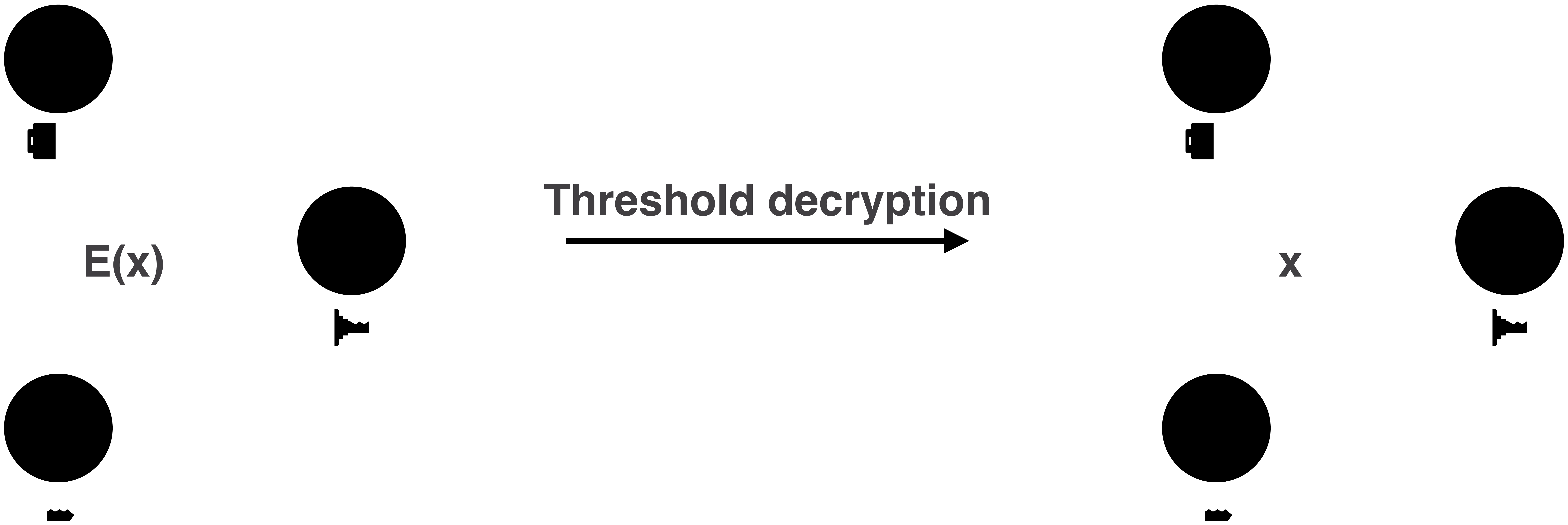
# The inputs are simply encrypted using the global public FHE key



# Computation is done locally by validators using homomorphic operations



# Values can be decrypted by validators using a threshold protocol

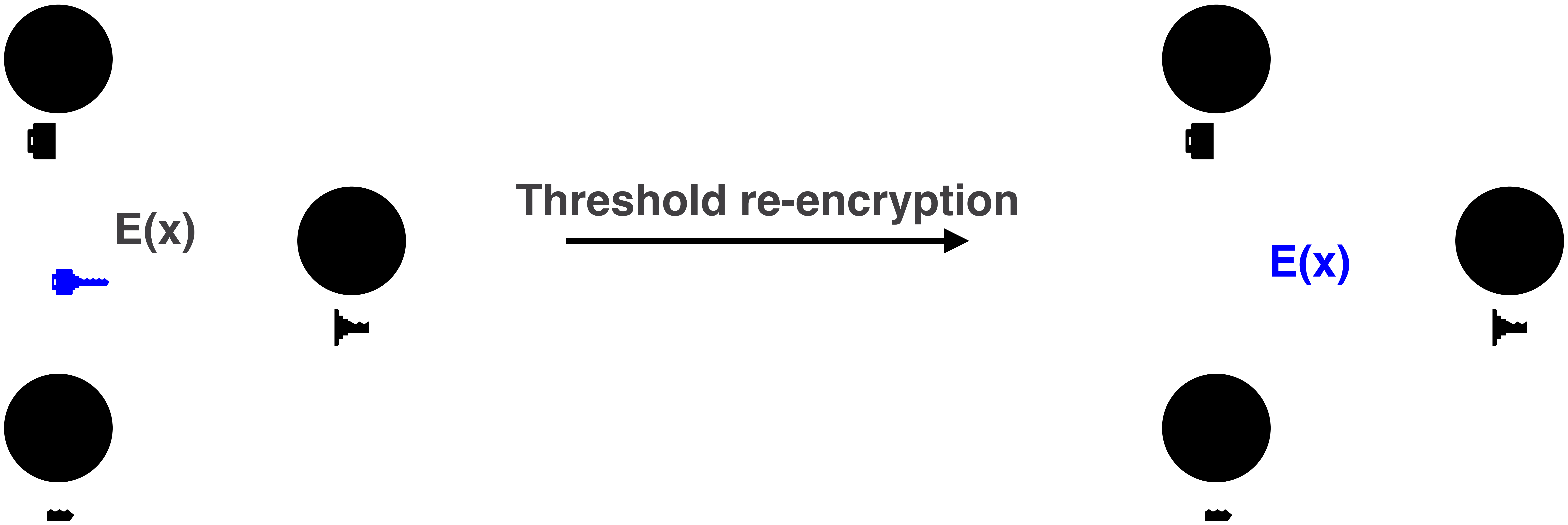


E(x)

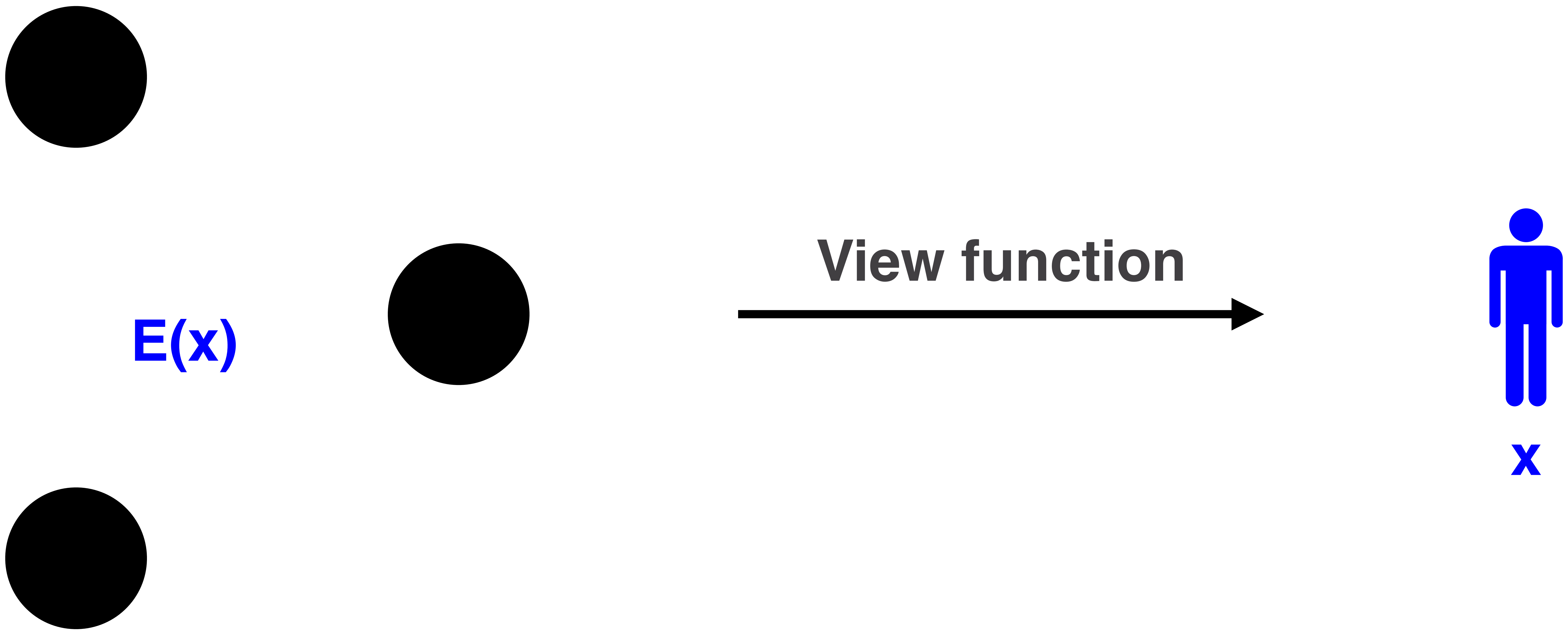
Threshold decryption

x

# Values can also be re-encrypted to user public key using a threshold protocol

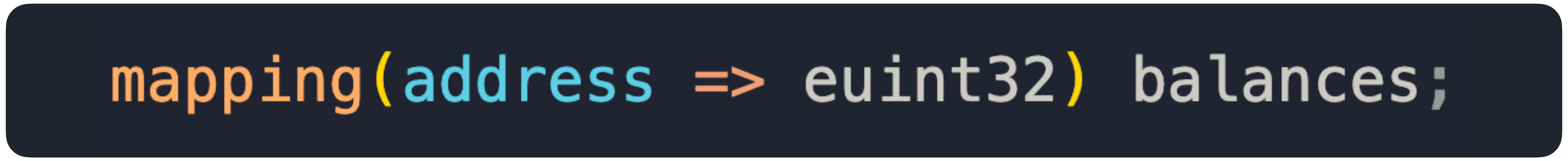


# Re-encrypted values can be read and decrypted by the user owning the key



# Encrypted Smart Contracts

# TFHE::euint32



```
mapping(address => euint32) balances;
```

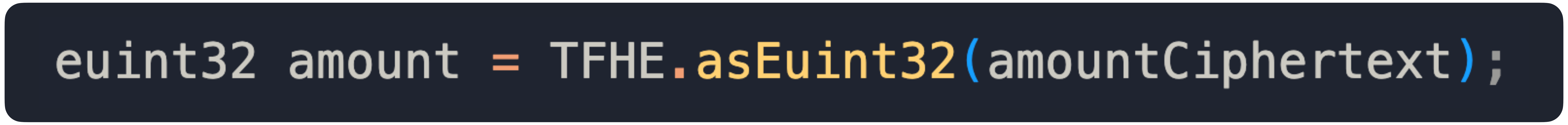
Represents an encrypted value

Can be used for computation, storage, composition, etc

Efficient since they are small (only handles to ciphertexts)

euint8, euint16, ueint32, …
add, sub, mul, eq, le, gt, …

# TFHE.asEuint



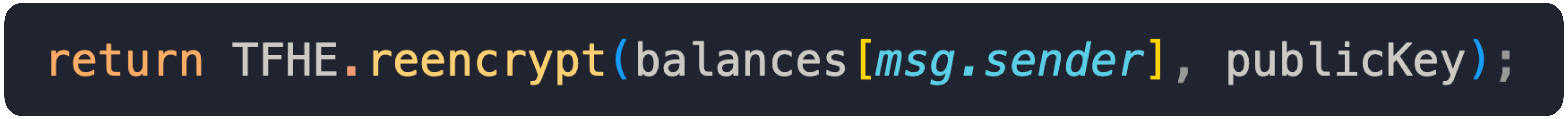
```
euint32 amount = TFHE.asEuint32(amountCiphertext);
```

Well-formed to not leak
anything about global FHE
secret key

Prevent user from decrypting
arbitrary ciphertexts

Ciphertexts include
ZK proof of plaintext knowledge,
that must be checked

# TFHE.reencrypt

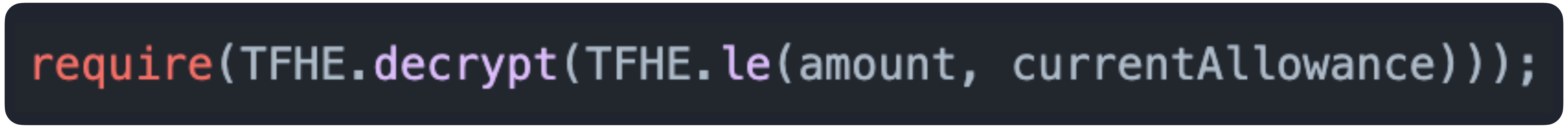


```
return TFHE.reencrypt(balances[msg.sender], publicKey);
```

Securely re-encrypt
from global FHE public key
to user NaCl public key

Optional authentication
token to trust identity of
sender (EIP-712)

# TFHE.decrypt



```
require(TFHE.decrypt(TFHE.le(amount, currentAllowance)));
```

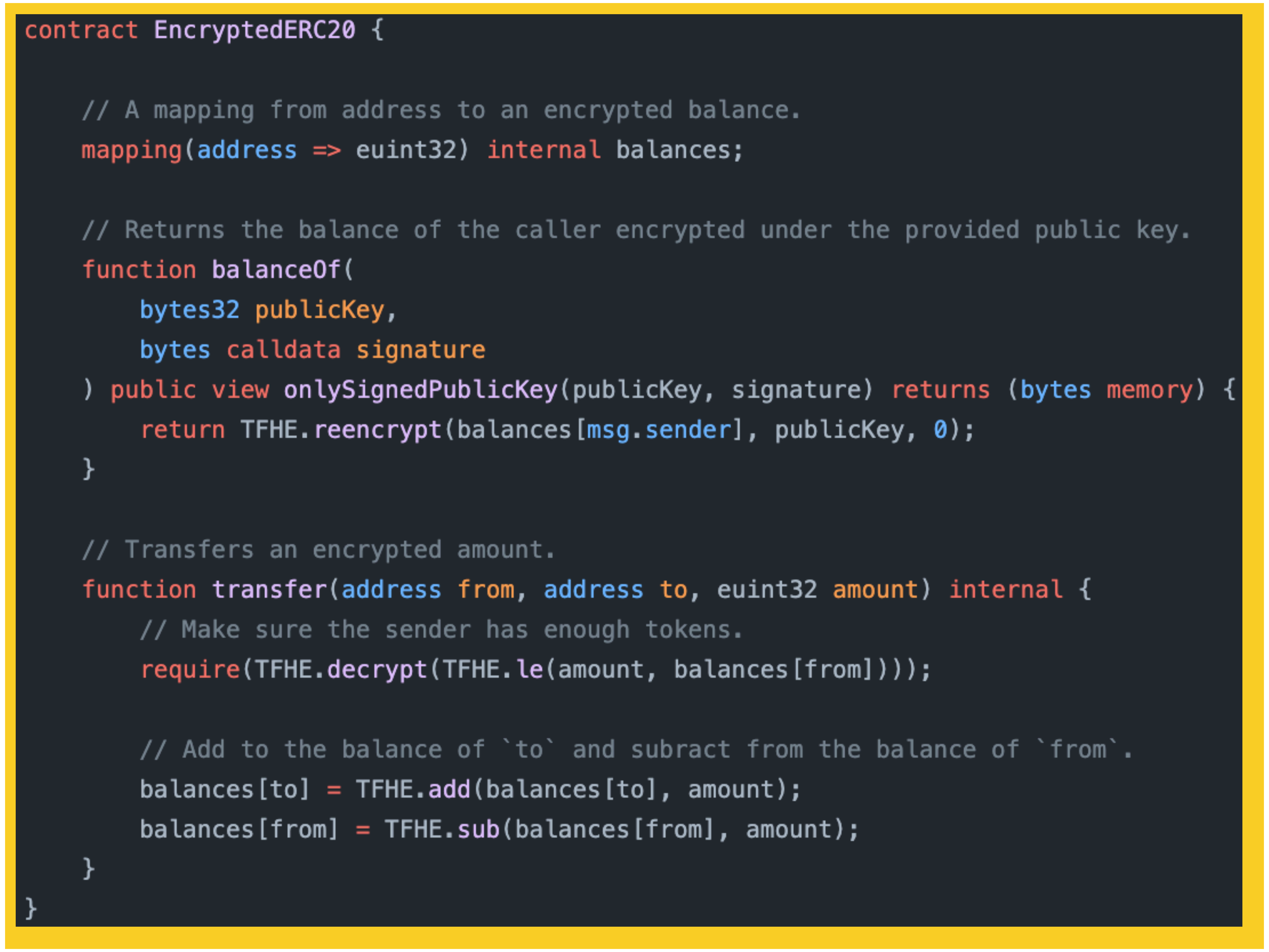
Evaluate condition
homomorphically

Decrypt boolean,
and abort if false

Leaks *something*!

Alternative is
TFHE.cmux(eCondition, eTrueValue, eFalseValue)

# Developers can write confidential contracts without learning cryptography



```
contract EncryptedERC20 {

    // A mapping from address to an encrypted balance.
    mapping(address => euint32) internal balances;

    // Returns the balance of the caller encrypted under the provided public key.
    function balanceOf(
        bytes32 publicKey,
        bytes calldata signature
    ) public view onlySignedPublicKey(publicKey, signature) returns (bytes memory) {
        return TFHE.reencrypt(balances[msg.sender], publicKey, 0);
    }

    // Transfers an encrypted amount.
    function transfer(address from, address to, euint32 amount) internal {
        // Make sure the sender has enough tokens.
        require(TFHE.decrypt(TFHE.le(amount, balances[from])));

        // Add to the balance of `to` and subract from the balance of `from`.
        balances[to] = TFHE.add(balances[to], amount);
        balances[from] = TFHE.sub(balances[from], amount);
    }
}
```

## Solidity Integration

fhEVM contracts are simple solidity contracts that are built using traditional solidity toolchains.

## Simple DevX

Developers can use the `euint` data types to mark which part of their contracts should be private.

## SC-defined ACL

All the logic for access control of encrypted states is defined by developers in their smart contracts.

# Try the fhEVM yourself today

Github

Documentation

White Paper

# Questions?