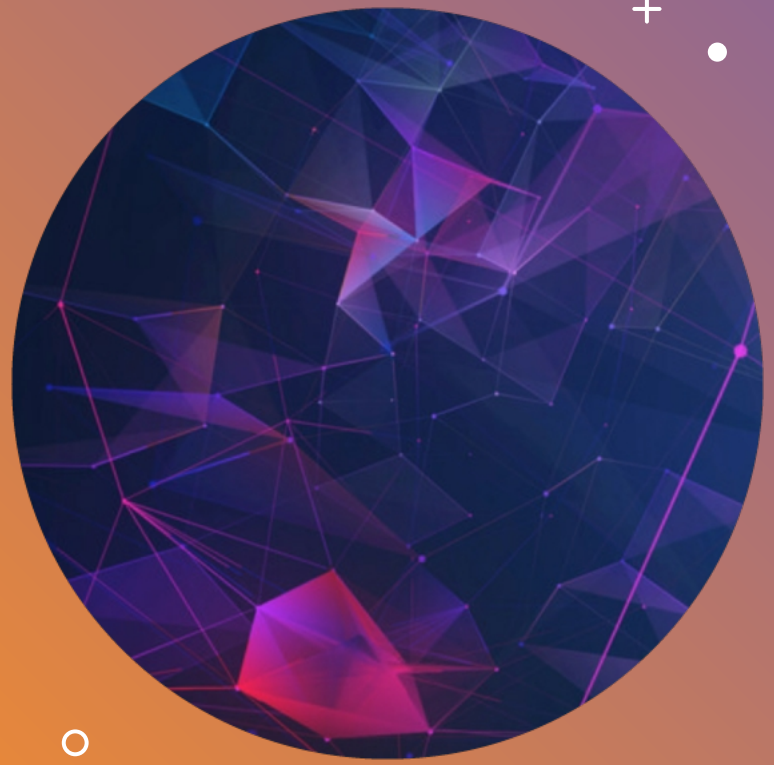
CRYPTOGRAPHY IN BLOCKCHAIN:

SECRET SHARING AND AN IMPLEMENTATION IN OPENETHEREUM

Francesco Rambaldi

Blockchain and Cryptocurrencies

Academic Year: 2020/2021



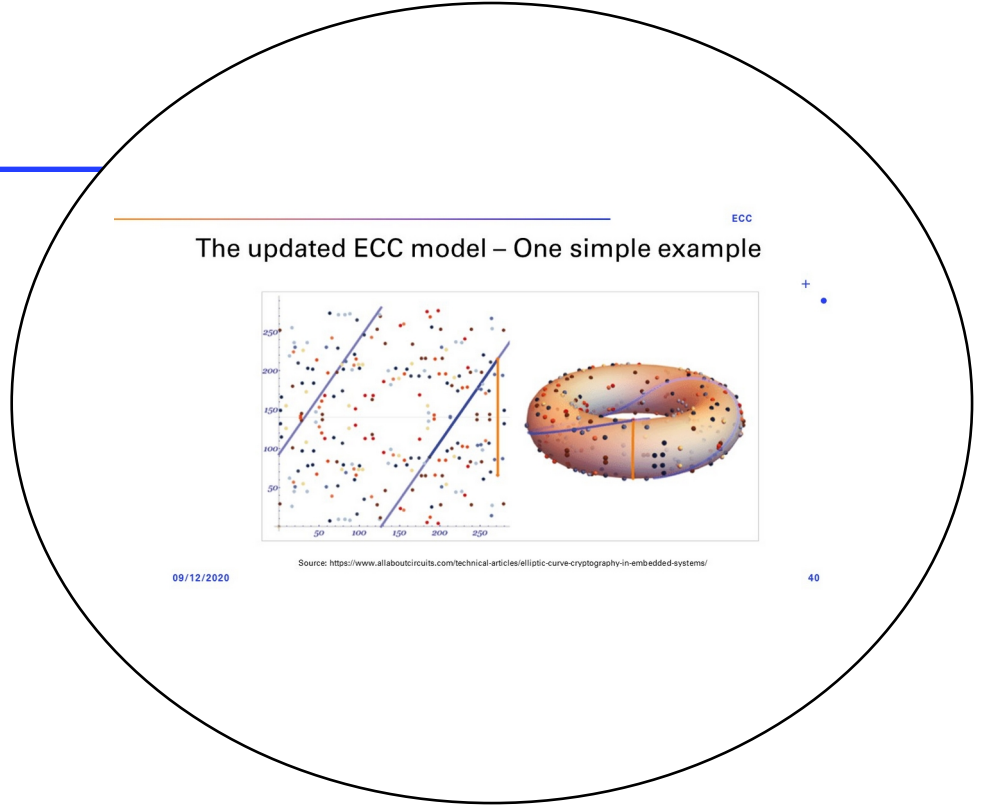
AGENDA

Secret Sharing

Open Ethereum and the Secret Store

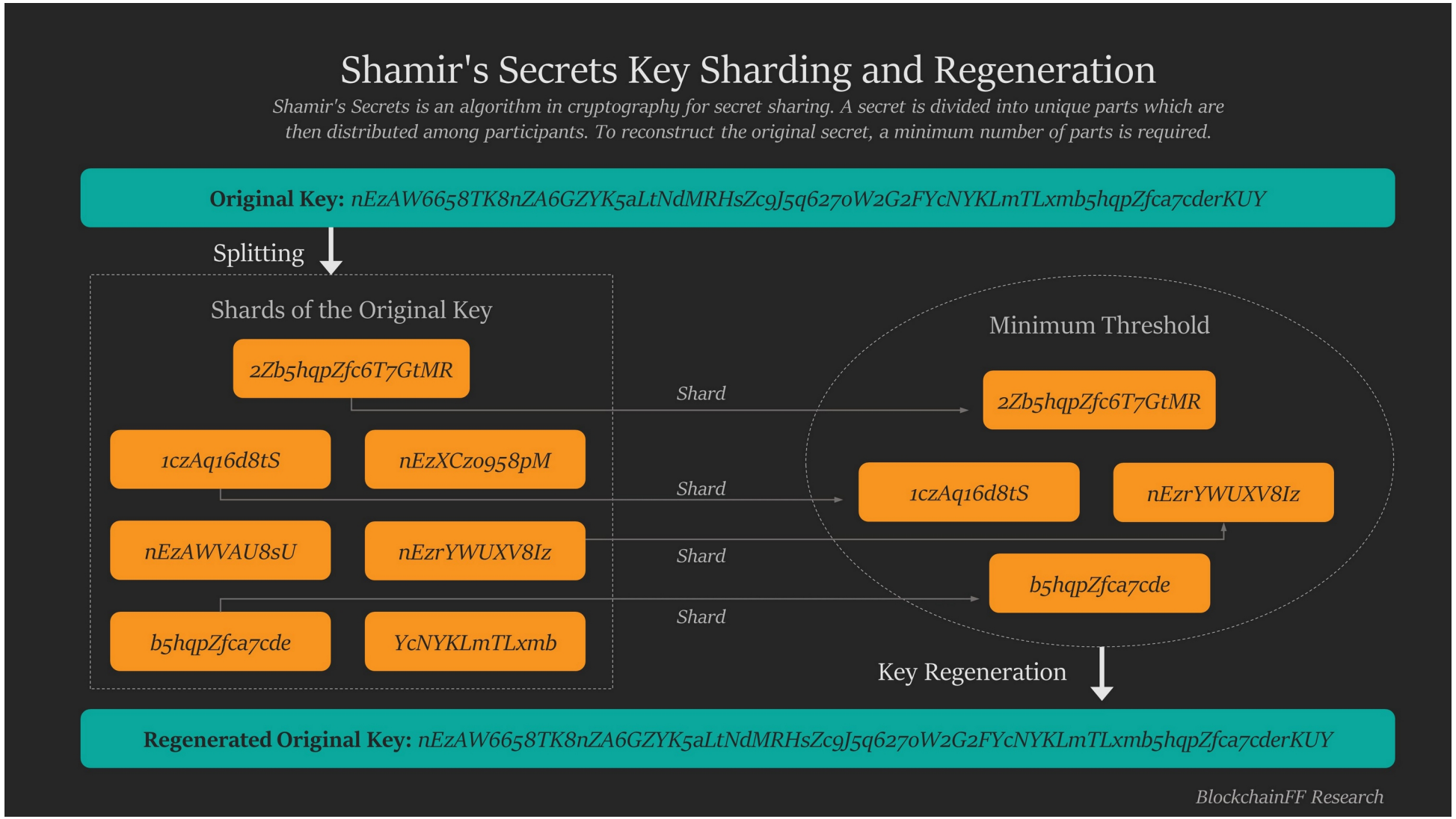
A Practical Use Case

A Local Implementation

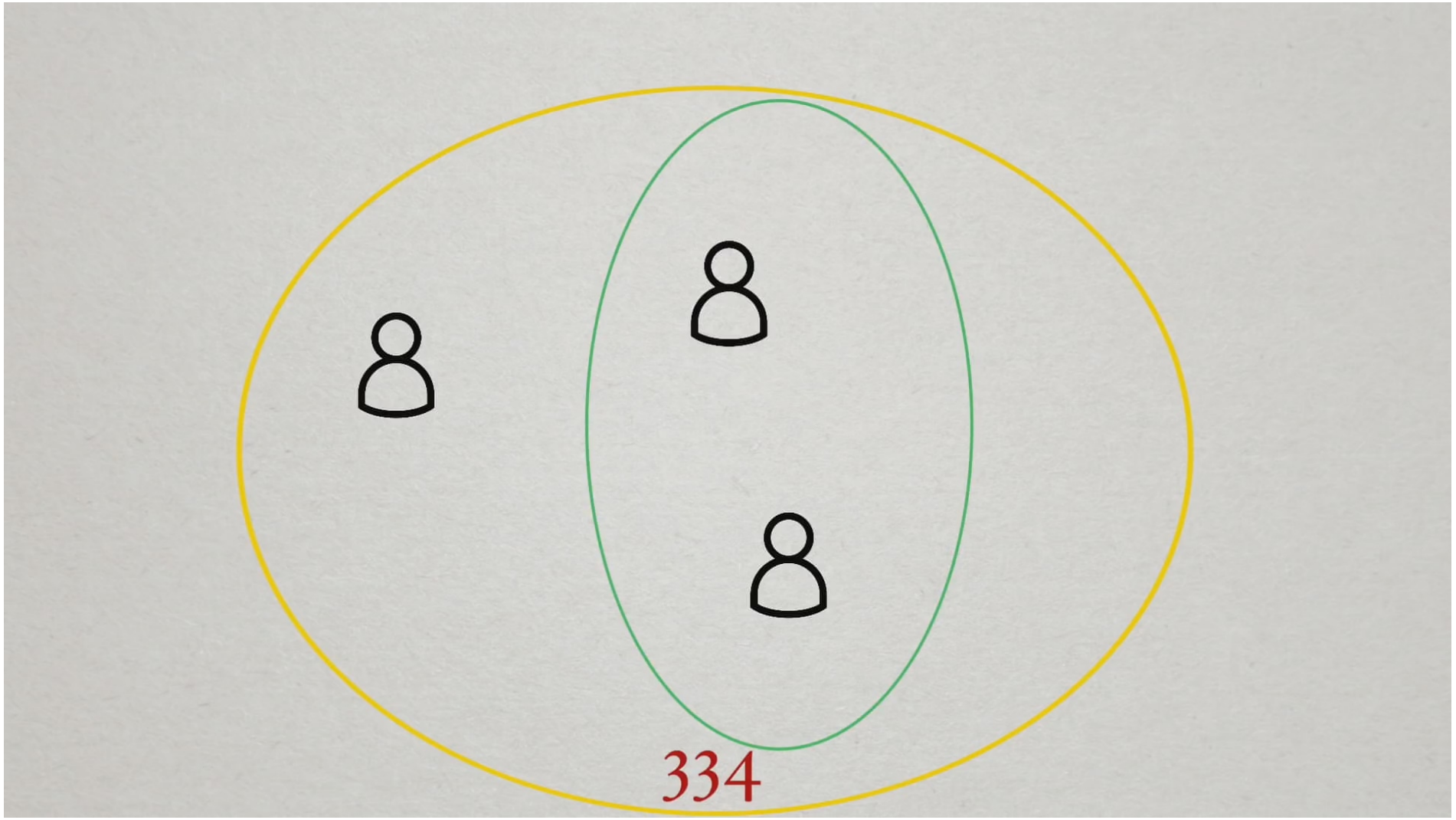

ECC
The updated ECC model – One simple example
250
200
150
100
50
50
100
150
200
250
Source: https://www.allaboutcircuits.com/technical-articles/elliptic-curve-cryptography-in-embedded-systems/
09/12/2020
40

What's next?

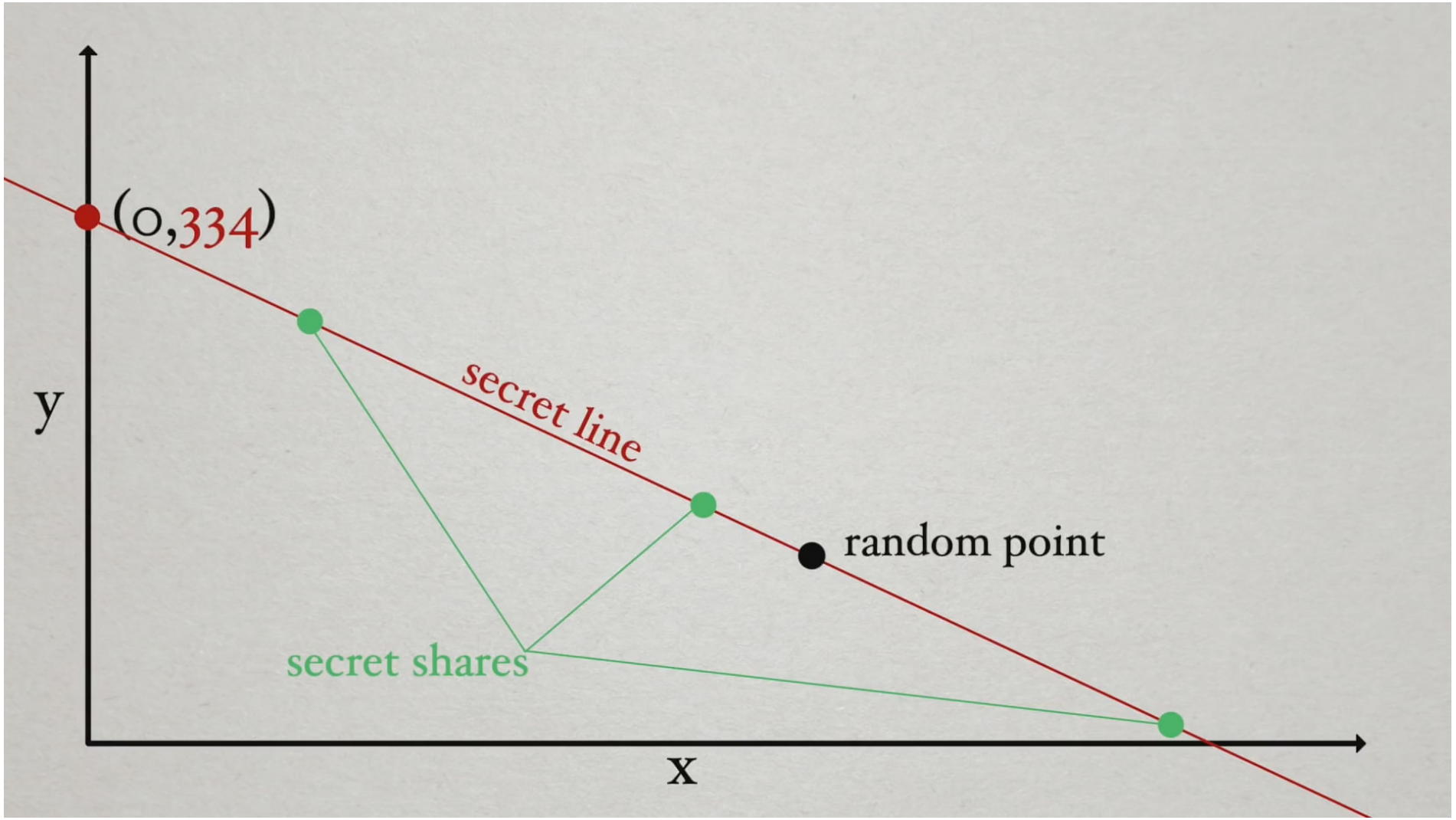
SECRET SHARING



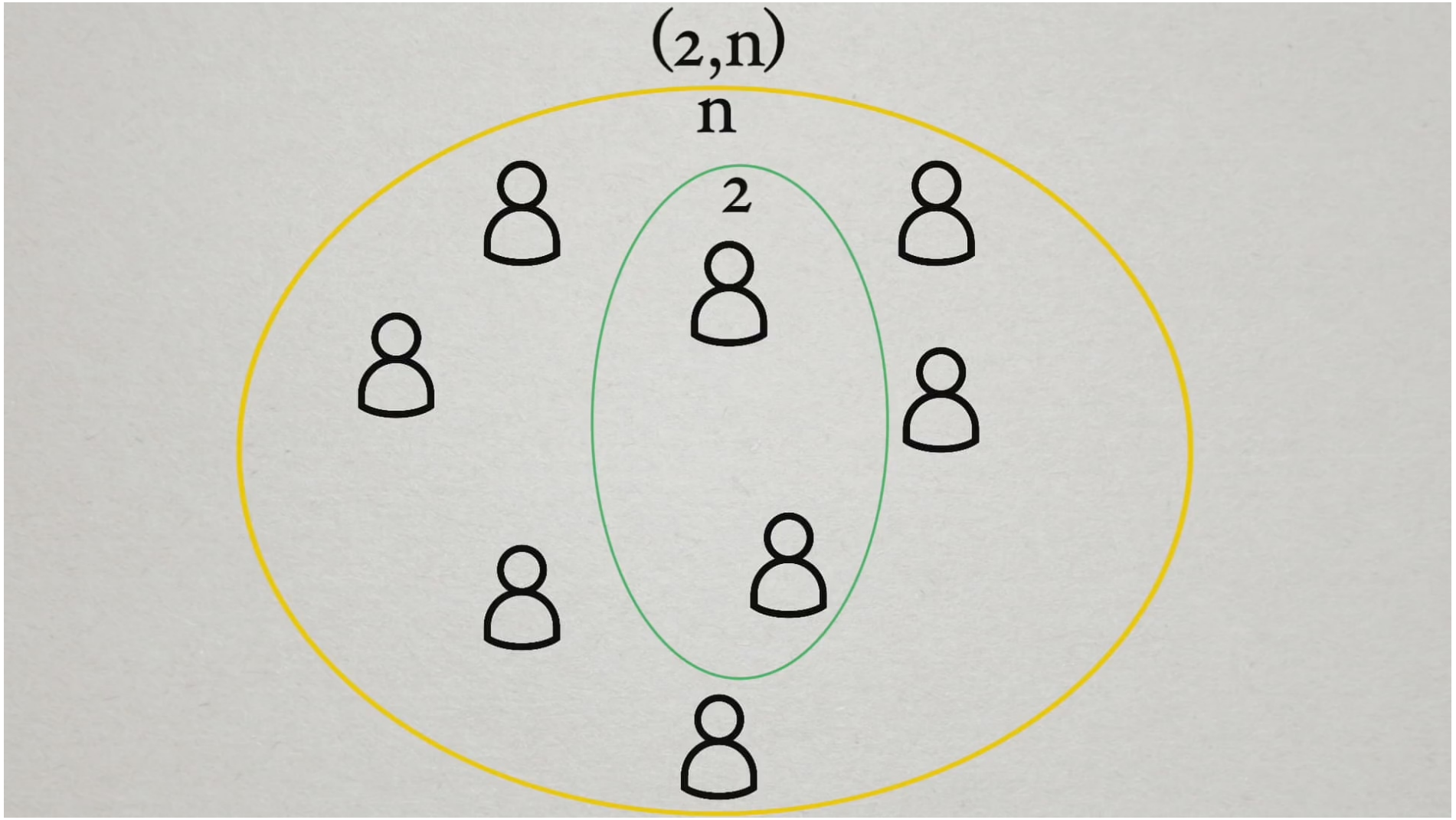
Source: https://hackernoon.com/the-evolution-of-the-public-private-key-encryption-in-blockchain-systems-b56a3xnn



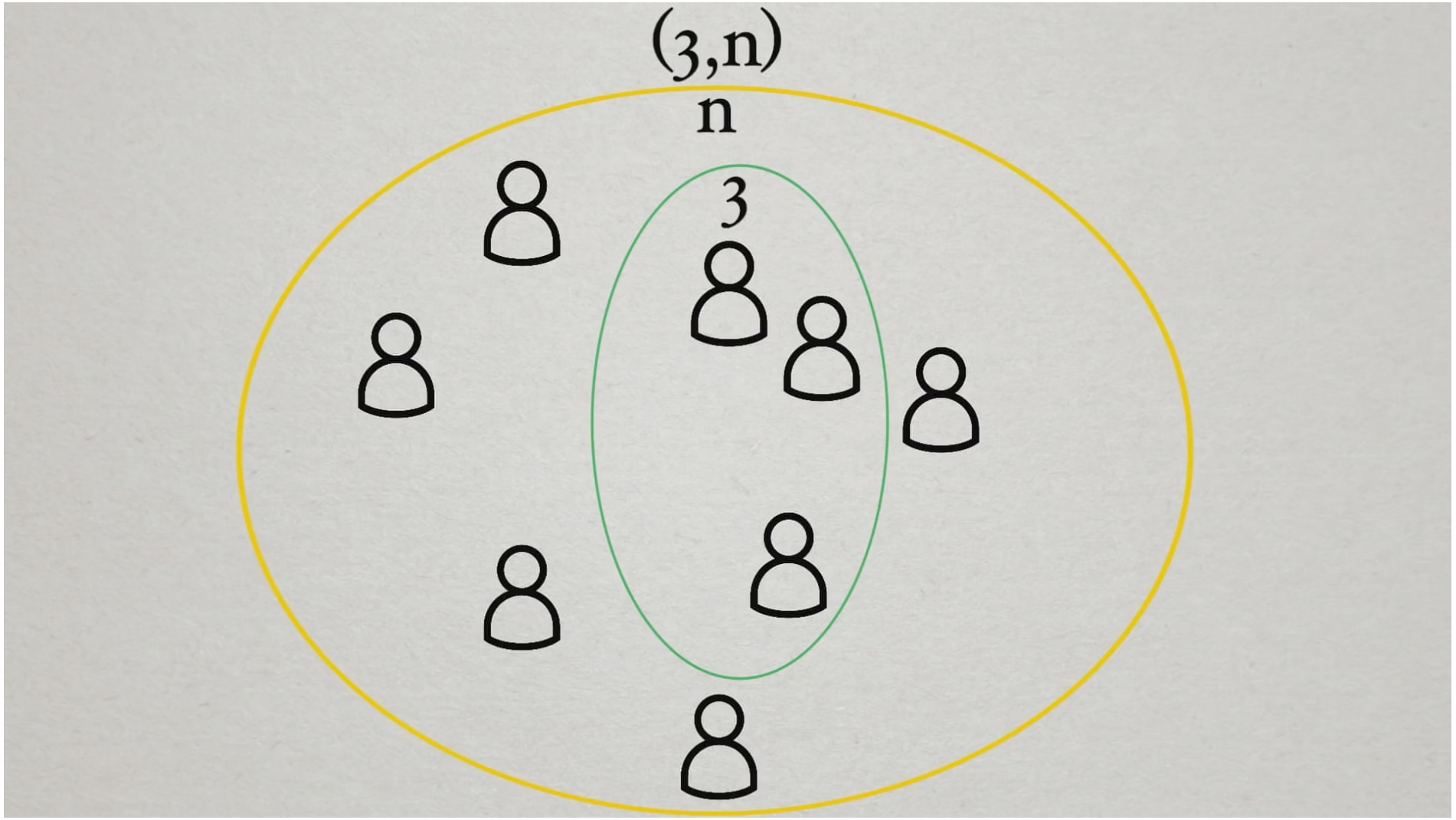
Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ



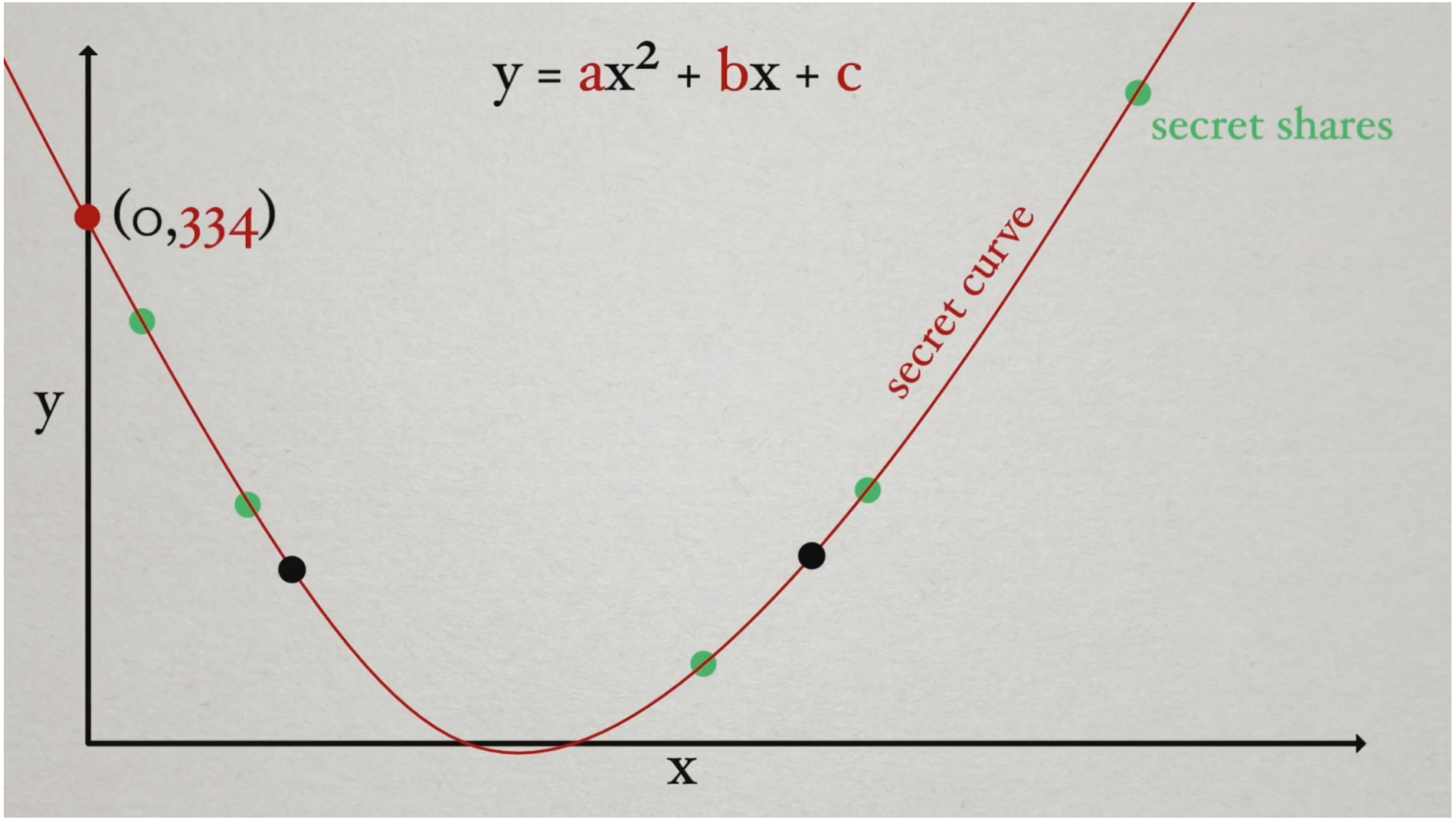
Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ



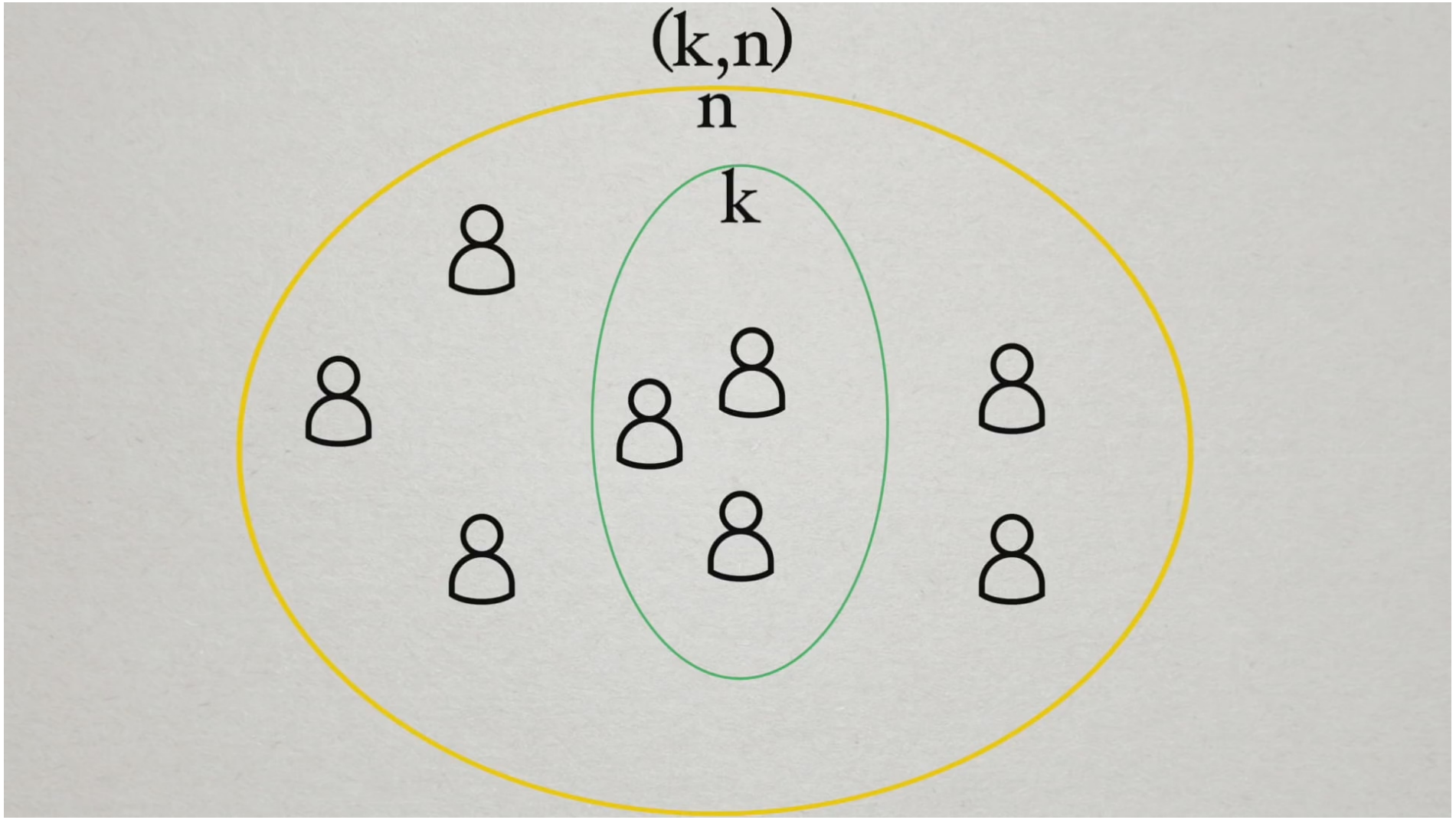
Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ



Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ



Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ



Source: https://www.youtube.com/watch?v=iFY5SyY3IMQ

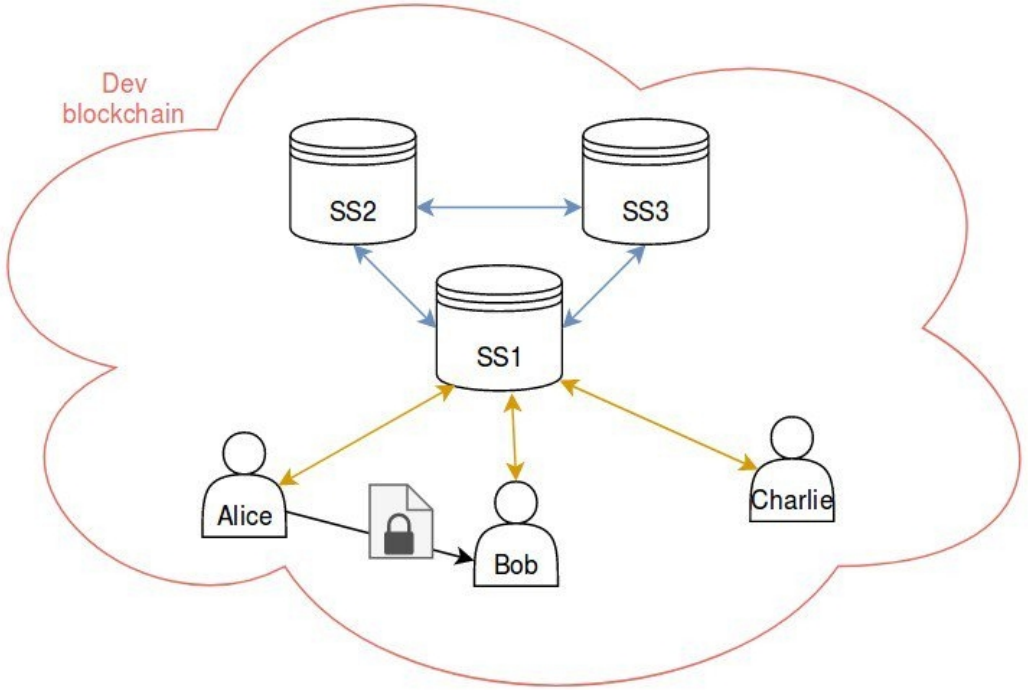
OPEN ETHEREUM AND THE SECRET STORE

Open Ethereum

- “OpenEthereum’s goal is to be the fastest, lightest, and most secure Ethereum client”

Secret Store

- Feature built on the OpenEthereum client that allows users to store on the blockchain a fragmented ECDSA key which retrievals are controlled by a SmartContract



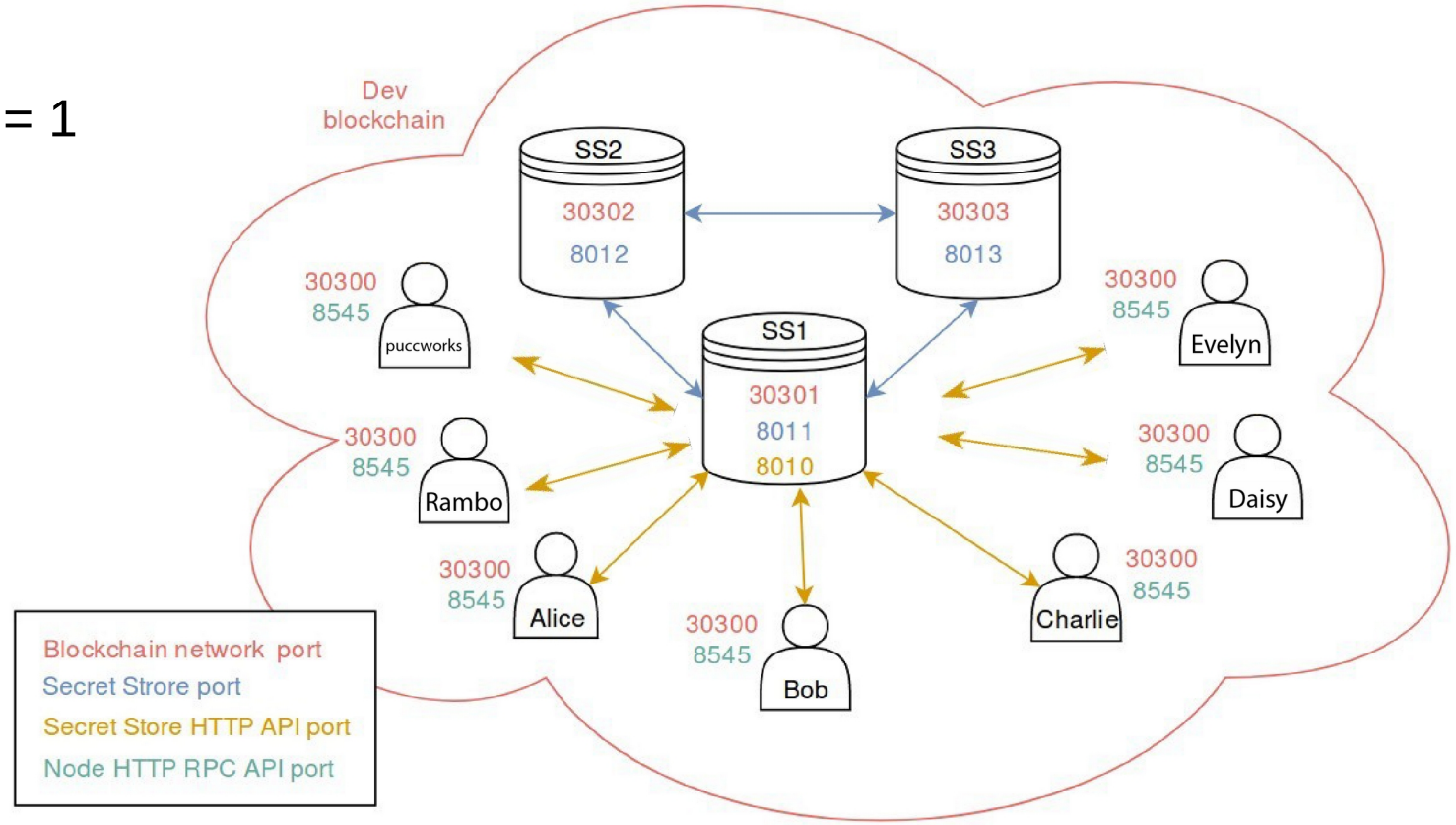
A PRACTICAL USE CASE

Welcome to the **puccworks** game company!

A LOCAL IMPLEMENTATION

Step 1: Setup the Private Dev Chain

- Threshold = 1



Step 2.1: Deploy the Smart Contracts

- Permissions
- CoolLeaderboard

- Python Script

Permissions Smart Contract

```
// allows owner only
modifier onlyOwner(){
    require(owner == msg.sender, "Sender not authorized");
    _;
}

// set the deployed CoolLeaderboard contract address
function setAddress(address _address) onlyOwner() public {
    leaderboard = CoolLeaderboard(_address);
}

// get the contest winners from the leaderboard
function getWinners() constant public returns (address[] memory) {
    return leaderboard.getWinners();
}

function checkPermissions(address user, bytes32 document) public view returns (bool) {  // view because the function will not mod
    address[] memory winners = getWinners();
    for(uint i = 0; i < winners.length; i++){
        if(document == documentKeyId && user == winners[i]){
            return true;
        }
    }
    return false;
}
```

CoolLeaderboard Smart Contract

```
// call to update leaderboard
function addScore(string memory user, uint score) public returns (bool) {

    // if the score is too low, don't update
    if (leaderboard[leaderboardLength-1].score >= score) return false;

    // store the message sender address
    address addr = msg.sender;

    // loop through the leaderboard
    for (uint i=0; i<leaderboardLength; i++) {

        // find where to insert the new score
        if (leaderboard[i].score < score) {

        // shift leaderboard
        User memory currentUser = leaderboard[i];
        for (uint j=i+1; j<leaderboardLength+1; j++) {
            User memory nextUser = leaderboard[j];
            leaderboard[j] = currentUser;
            currentUser = nextUser;
        }

        // insert
        leaderboard[i] = User({
            user: user,
            score: score,
            playerAddress: addr
        });

        // delete last from list
        delete leaderboard[leaderboardLength];

        return true;    // operation completed successfully!
        }
    }
}
```

CoolLeaderboard Smart Contract

```
// set the winners of the month
function setWinners() onlyOwner() public {
    for(uint i = 0; i < leaderboardLength; i++) {     // only the owner can perform this!
        winners[i] = leaderboard[i].playerAddress;
    }
}

// return the winners of the month (this will be called by the Permissions contract)
function getWinners() public view returns (address[] memory){
    return winners;
}
```

Step 2.2: Set Scores

```
"""
UPDATE THE LEADERBOARD WITH SOME USER SCORES
"""
# Alice scored 10000 in the wonderful "COOL Pizza!" Game
w3.parity.personal.unlock_account(alice_address_checksum, "alicepwd")
tx_hash = deployed_cool_leaderboard_contract.functions.addScore("Alice",10000).transact({"from": alice_address_checksum})
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)

# Bob scored 5000 in the wonderful "COOL Pizza!" Game
w3.parity.personal.unlock_account(bob_address_checksum, "bobpwd")
tx_hash = deployed_cool_leaderboard_contract.functions.addScore("Bob",5000).transact({"from": bob_address_checksum})
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)

# Charlie scored 200000 in the wonderful "COOL Pizza!" Game
w3.parity.personal.unlock_account(charlie_address_checksum, "charliepwd")
tx_hash = deployed_cool_leaderboard_contract.functions.addScore("Charlie",200000).transact({"from": charlie_address_checksum
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)

# Daisy scored 600000 in the wonderful "COOL Pizza!" Game. Daisy is our champion!
w3.parity.personal.unlock_account(daisy_address_checksum, "daisypwd")
tx_hash = deployed_cool_leaderboard_contract.functions.addScore("Daisy",600000).transact({"from": daisy_address_checksum})
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)

# Evelyn scored 50000 in the wonderful "COOL Pizza!" Game
w3.parity.personal.unlock_account(evelyn_address_checksum, "evelynpwd")
tx_hash = deployed_cool_leaderboard_contract.functions.addScore("Evelyn",50000).transact({"from": evelyn_address_checksum})
tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)
```

Step 3: Restart the BlockChain

- Now the TOML files are updated, thus we need to restart the development chain in order to take advantage of the configuration changes

Step 4: Have fun with COOL Pizza! (and win the contest!)