Objectives

Goal:

- Show how Aleo is uniquely positioned to enable compliant private payments and transactions

Topics:

- Tokens and tokenomics
  - Stablecoins (as specific implementation of tokens)
  - Compliance
- Build-a-token
  - Tokens as Record-type
  - Public Swaps
  - Private Swaps
  - Public/Private conversions
- Drawback to how `credits.aleo` is implemented and how Aleo is currently enabling private payments

Takeaways:

- Devs should learn:
  - Why Aleo is uniquely positioned to enable private payments and transactions
  - How to create a basic token in Leo
  - How to implement a public and private swap function with that token

Aleo | Workshop

Compliant Private Tokens

Agenda

1 Logistics

2 What is a Token?

3 Privacy & Compliance

4 What is Aleo?

5 Tokens on Aleo

6 Compliance In-Practice

7 Q&A

8 Hands-on Challenge

What You'll Need

Leo Playground: https://**play.leo-lang.org/**

OR

IDEs: VSCode / Sublime Text / IntelliJ

- Install Rust
- Install Leo
- Install Leo Extension for your IDE

Workshop: https://github.com/**alex-aleo/private-token-workshop**

Testnet Faucet: https://**test-faucet.aleo.org/**

Other Resources

Aleo Developer Docs: https://**developer.aleo.org**

Leo Docs: https://**docs.leo-lang.org**

Discord: https://discord.gg/**aleo**

Devs Telegram:

What is a Token?

- Digital assets representing value or utility onchain
- **Fungible Tokens**:
  - Units are interchangeable with each other
  - **Stablecoins**, ERC-20 tokens, Bitcoin
- **Non-Fungible Tokens (NFTs)**:
  - Units are unique and possess distinct characteristics or value
  - Digital art, collectibles, virtual real estate, event tickets

What is a Token?

- **Tokenomics:**
  - The economics of a token, governing its creation, distribution, and usage
  - **Minting:**
    - Process of creating new tokens
  - **Burning:**
    - Removing tokens from circulation, often to reduce supply
  - **Transferring:**
    - Sending a token to another user or service

Privacy

- **Major problem:**
  - Blockchains are fully public ledgers
  - All holdings and transactions are publicly visible
    - Surveillance in **perpetuity**
- Example:
  - Etherscan

Privacy

- **Why is privacy important?**
  - Payments with stablecoins necessitate privacy
    - Users shouldn’t have to share everything they pay for
    - Business shouldn’t be able to see their competitors’ expenditures
- **Caveat:**
  - Any payments must also maintain regulatory compliance

Compliance

- **KYC / AML**
  - KYC = Know Your Customer
  - AML = Anti-Money Laundering
  - Set of regulations designed to prevent illicit activities
    - Fraud, money laundering, terrorist financing, etc.
  - Affected businesses must verify the identity of their clients
    - ex) Coinbase requires ID upload to open an account

Compliance

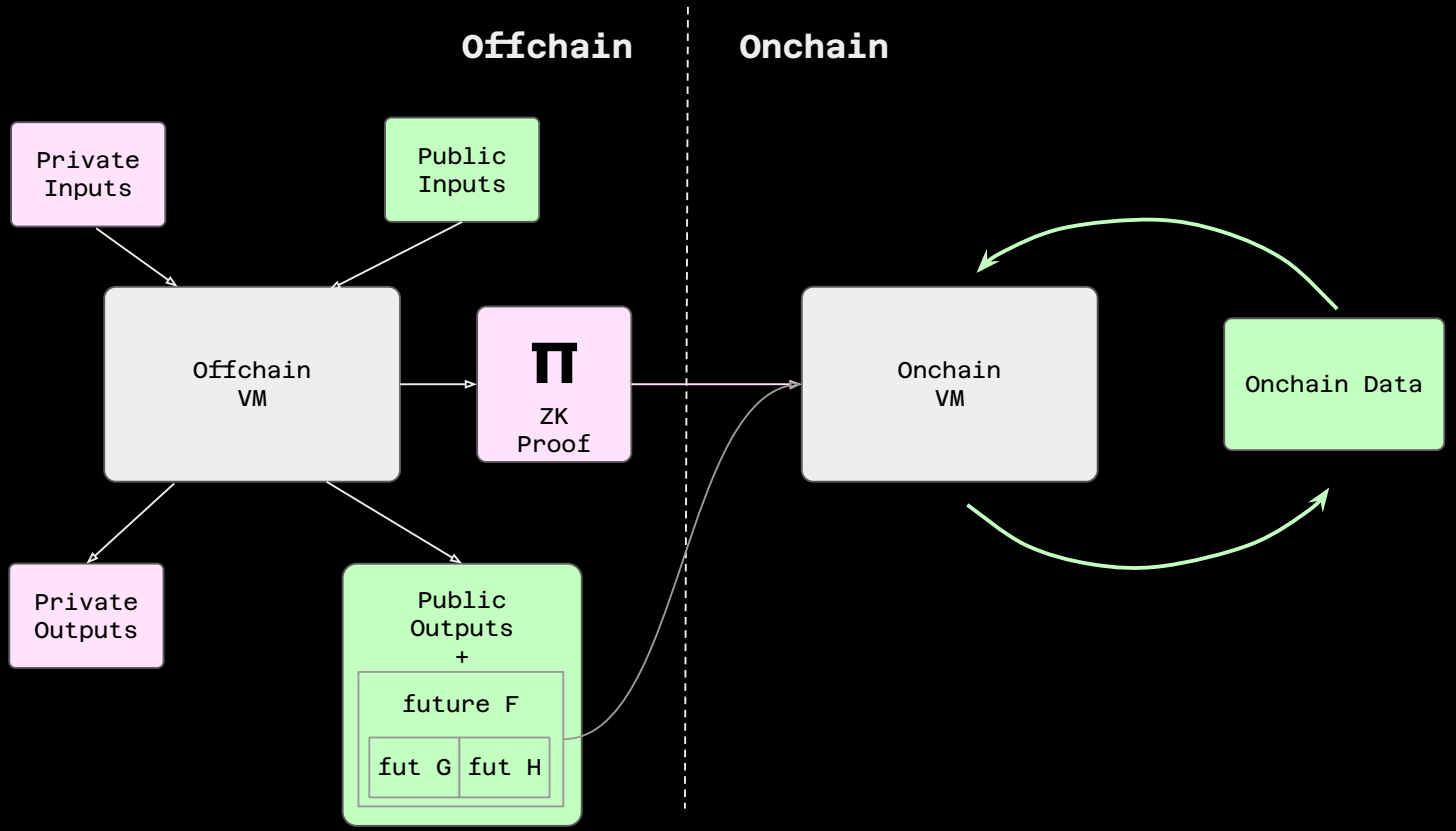
- **Sanctioned Address List**
  - Maintained by the Office of Foreign Assets Control (OFAC)
    - U.S. Treasury Department
  - List of cryptocurrency addresses associated with sanctioned entities

How can we maximize privacy while maintaining regulatory compliance?

Aleo

- Layer 1 blockchain with privacy as a first-class citizen
  - Powered by zero-knowledge proofs
- Privacy is **programmable**, so developers can choose what gets revealed
  - Bake-in compliance without sacrificing user privacy

Aleo Model


Offchain
Onchain
Private Inputs
Public Inputs
Offchain VM
π
ZK Proof
Onchain VM
Onchain Data
Private Outputs
Public Outputs +
future F
fut G
fut H

Tokens on Aleo

- Offchain State

record

```
L1  record Token {
L2      owner: address,
L3      balance: u64,
L4  }
```

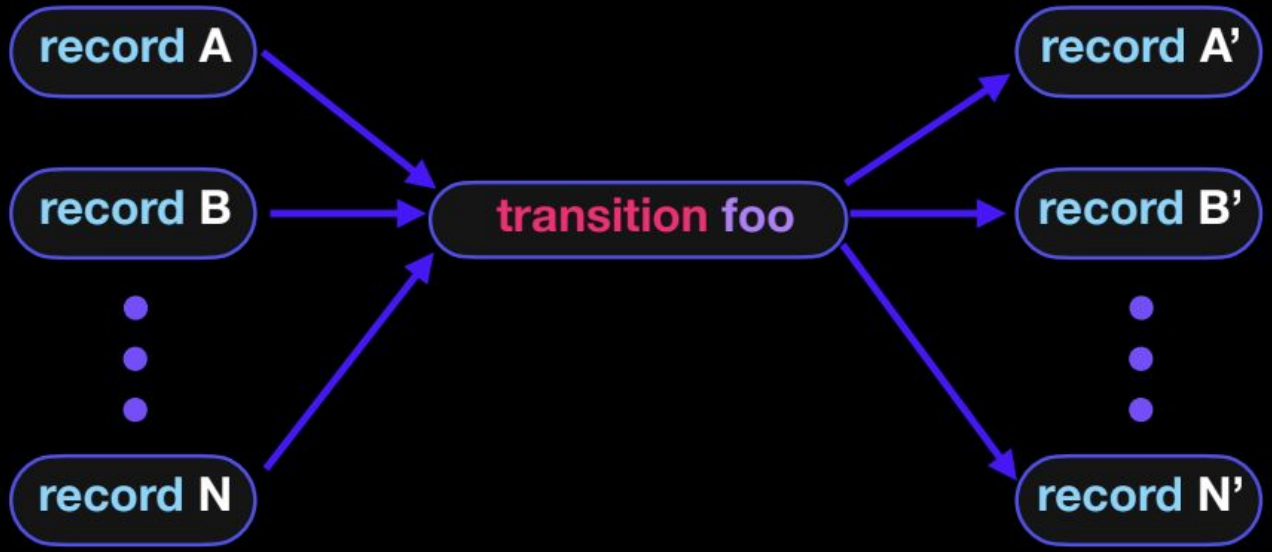
Application state is encoded in records

Users exclusively own their records

records enable concurrency and privacy

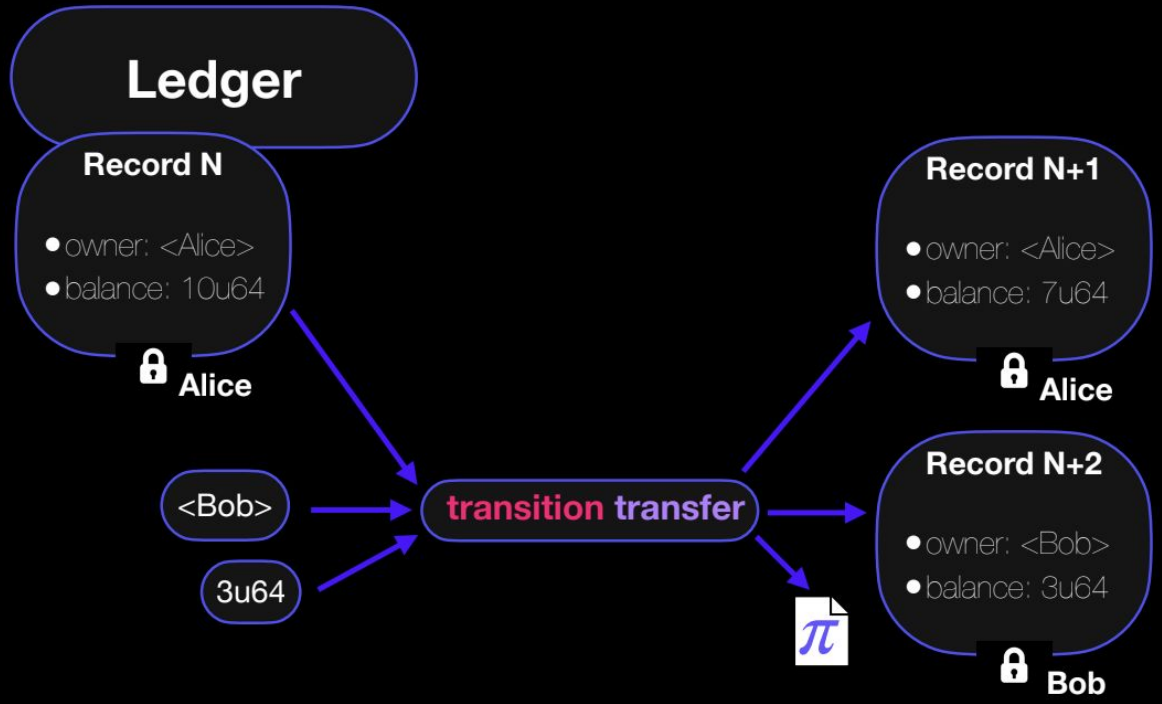
Tokens on Aleo

- Using records



Tokens on Aleo

- Using records



Tokens on Aleo

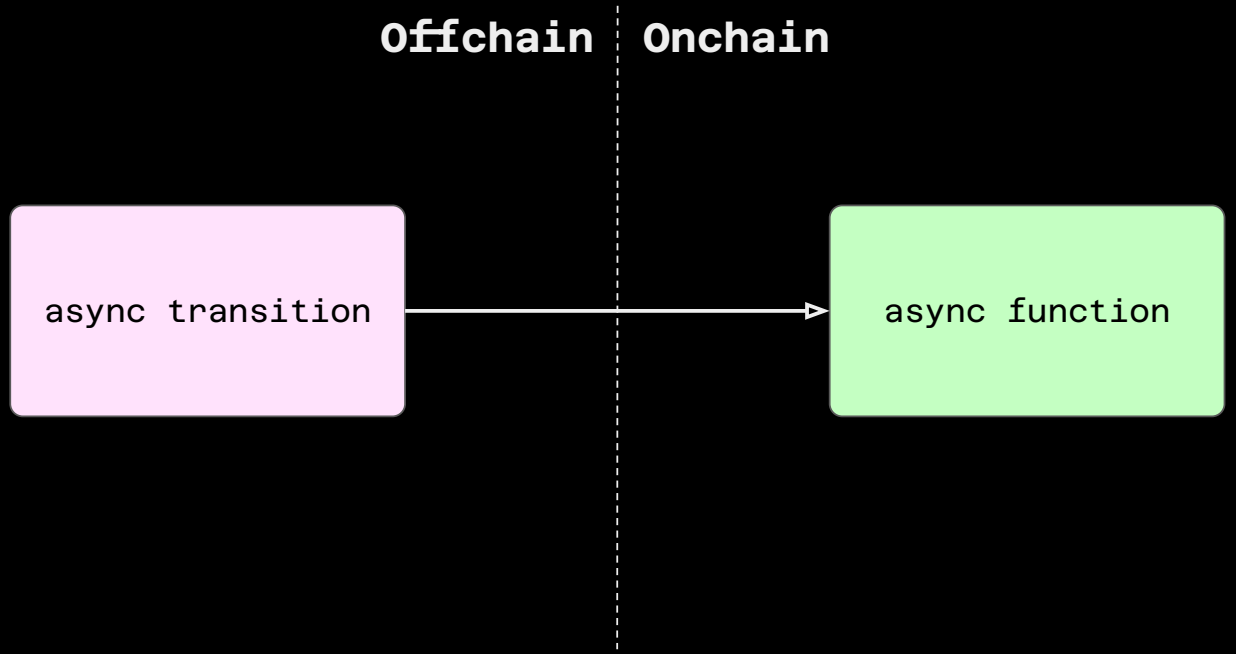
- Onchain State

mapping

```
L1 program token.aleo {
L2     mapping balances: address => u64;
L3     ...
L4 }
```

Tokens on Aleo

- Modifying Onchain State
  - The async model



Tokens on Aleo

- Modifying Onchain State
  - async transition
    - Offchain computation with ZK proof of execution
    - async keyword signals additional onchain computation to follow
      - Otherwise acts same as regular transition
    - Must return at least a Future
      - Call to an async function

Tokens on Aleo

- Modifying Onchain State
  - `async function`
    - Onchain computation
    - All inputs are public
    - Can only be called by `async transition`, not standalone

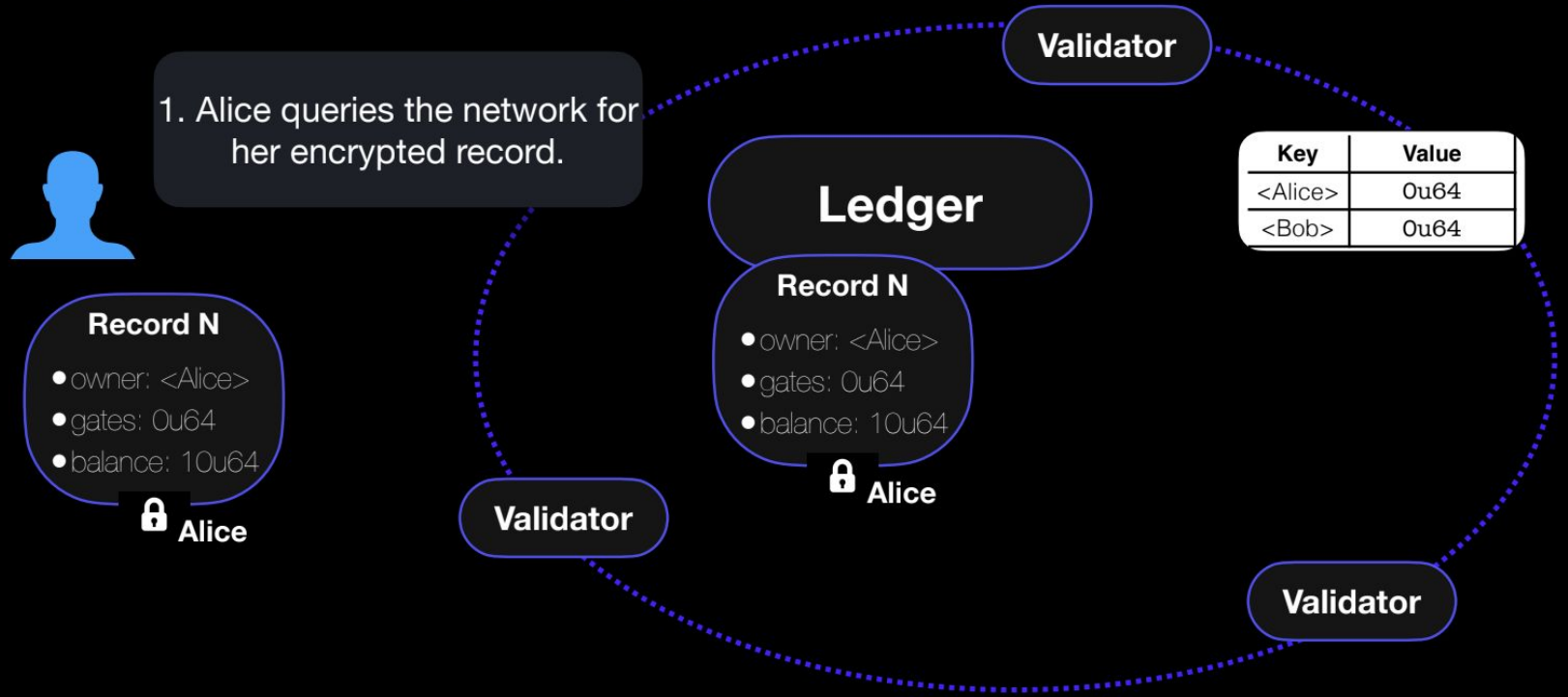
Tokens on Aleo

- Modifying Onchain State

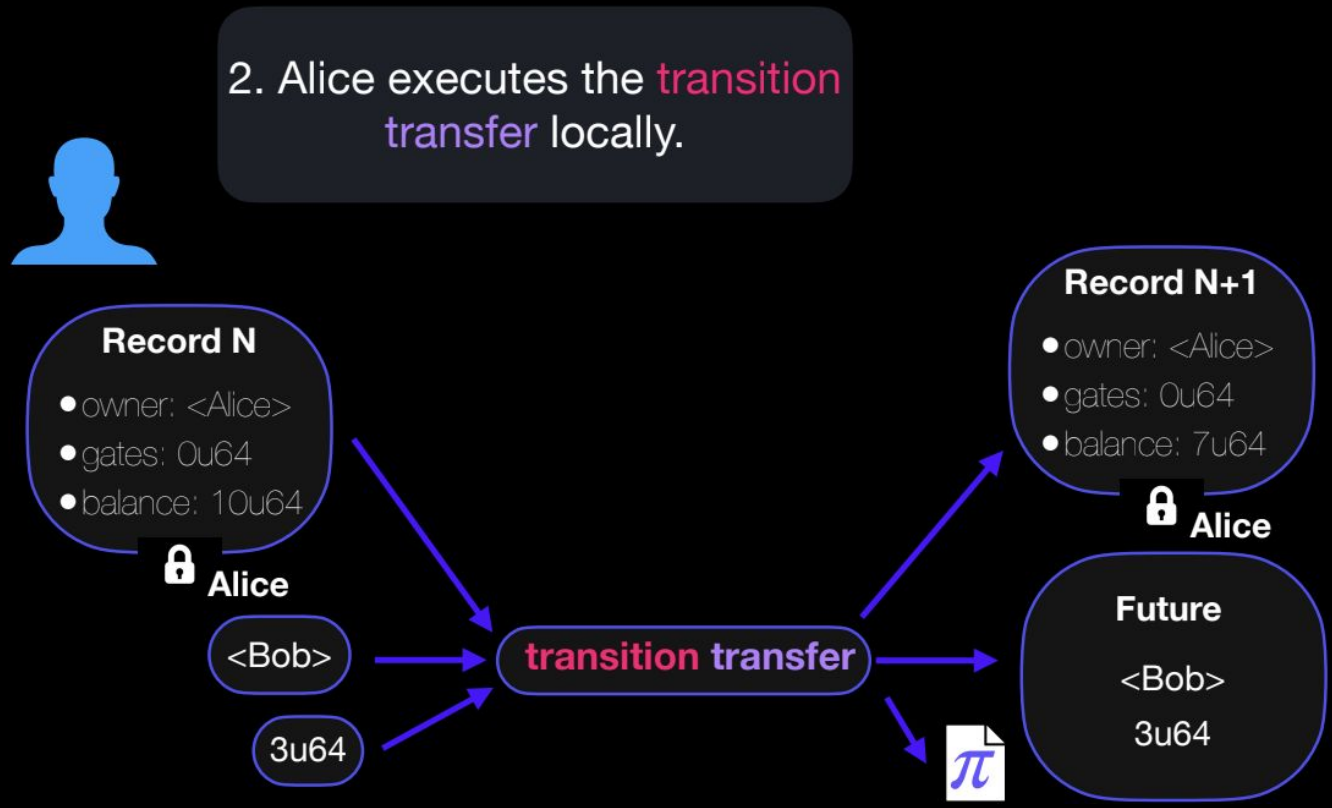
async

```
L1 async transition foo() -> Future {
L2     return bar();
L3 }
L4 async function bar() { // On-chain code }
```

Example


1. Alice queries the network for her encrypted record.
Validator
Ledger
Key
Value
<Alice>
0u64
<Bob>
0u64
Record N
owner: <Alice>
gates: 0u64
balance: 10u64
Alice
Record N
owner: <Alice>
gates: 0u64
balance: 10u64
Alice
Validator
Validator

Example

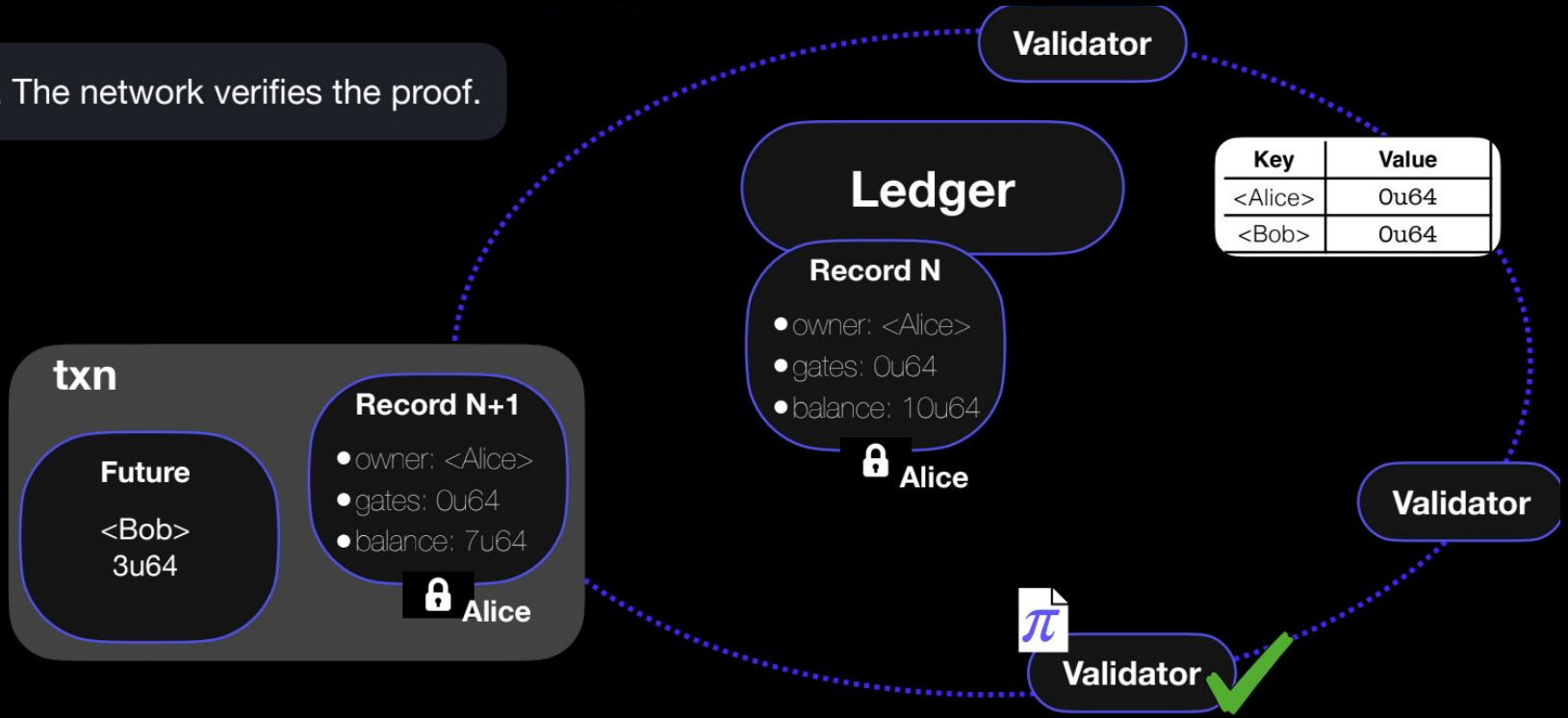

2. Alice executes the transition transfer locally.
Record N
owner: <Alice>
gates: 0u64
balance: 10u64
Alice
<Bob>
3u64
transition transfer
Record N+1
owner: <Alice>
gates: 0u64
balance: 7u64
Alice
Future
<Bob>
3u64
π

Example

3. Alice produces a transaction and sends it to the network.

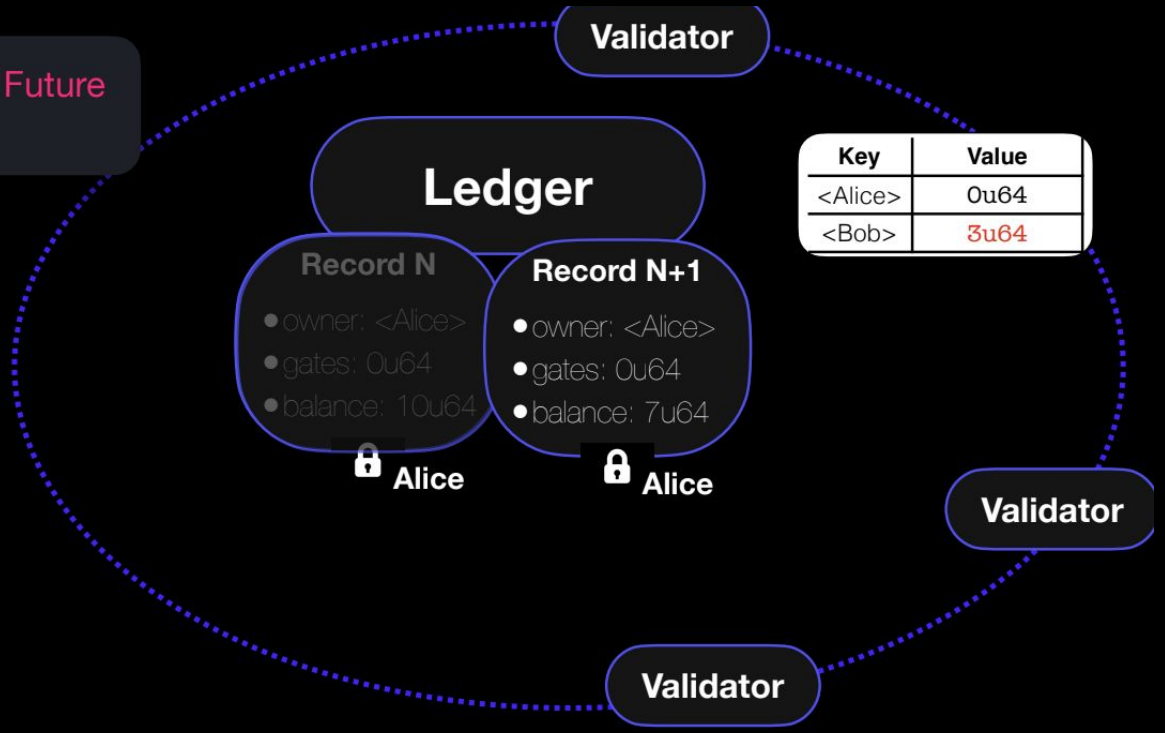
Example

4. The network verifies the proof.



Example

5. The network executes the Future and stores the new record.



Programmable Compliance

- Maintain updated list of OFAC sanctioned addresses in a mapping onchain
- Add assertions in every function that querying this mapping with transaction sender/recipients
  - Prevent any transfers or swaps of tokens from or to sanctioned addresses

Questions?


Let’s code!

Your Mission:

1. Build a token program in Leo using the provided template code. Your program must include:
    i. `mint_public` & `mint_private` functions
    ii. `transfer_public` & `transfer_private` functions
    iii. Compliance checks against `workshop_ofac.aleo` for all of the above
2. Deploy your program to Testnet.
3. Interact with your deployed program onchain:
    i. Publicly mint 100 tokens to your address
    ii. Publicly transfer those tokens to <WORKSHOP_ADDRESS>
    iii. Privately mint an additional 100 tokens to your address
    iv. Privately transfer those tokens to <WORKSHOP_ADDRESS>

Thank You!

Aleo | Workshop

TITLE HERE

Agenda

1. ITEM 1
2. ITEM 2
3. ITEM 3
4. ITEM 4
5. ITEM 5
6. ITEM 6
7. ITEM 7

Common errors and solutions

/ Version mismatches

/ Incorrect package installation

/ CSS/UI rendering issues

/ Ensure the connect button is functional

/ Check wallet selection and address retrieval

/ Validate Aleo Mainnet and Testnet connection strings

Rules & Eligibility

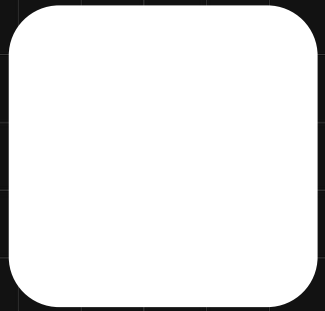
- Your program must introduce a new compliance feature (not just a fork of an existing one)
- The program should be functional and deployable on Aleo
- Code must be open-source and properly documented
- Only one reward per participant
- We reserve the right to reject submissions that are incomplete, plagiarized, or non-functional
- Publish your Aleo program code to a public GitHub repository

(Include a README.md with: A description of the compliance feature, how it works and its intended use case & deployment instructions)

Get your rewards

WIN!

What’s next?



More workshops