ackee blockchain

Summer School of Solana

LECTURE 5

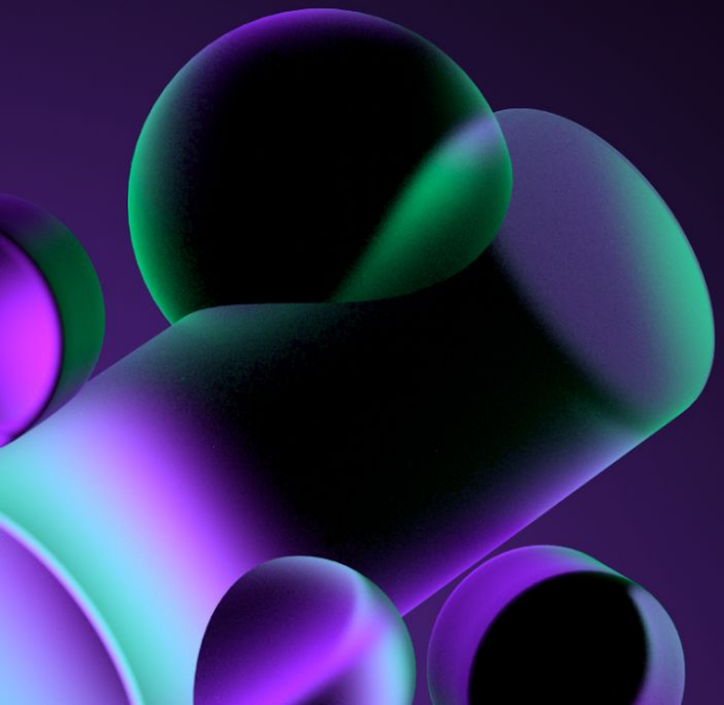
Solana Programming model 3/3

Tibor Tribus

About this lecture

- **Solana Programming model - recap**
  - Recap
  - Program Derived Addresses (PDAs)
  - Tokens
- **Hands on Example**
  - Turnstile program (fixes)

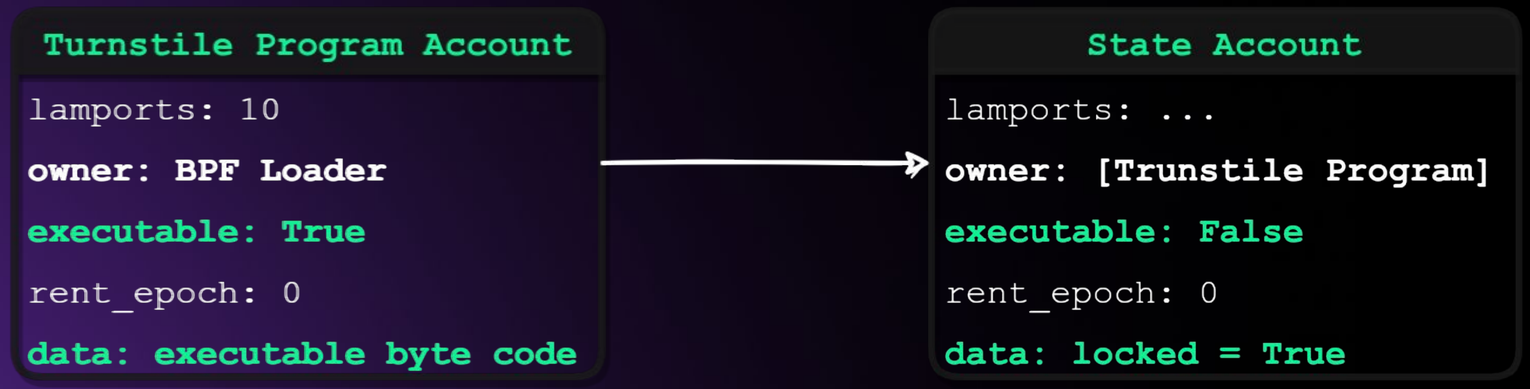
Recap




ACCOUNTS
LIGHTYEAR
ACCOUNTS EVERYWHERE
imgflip.com

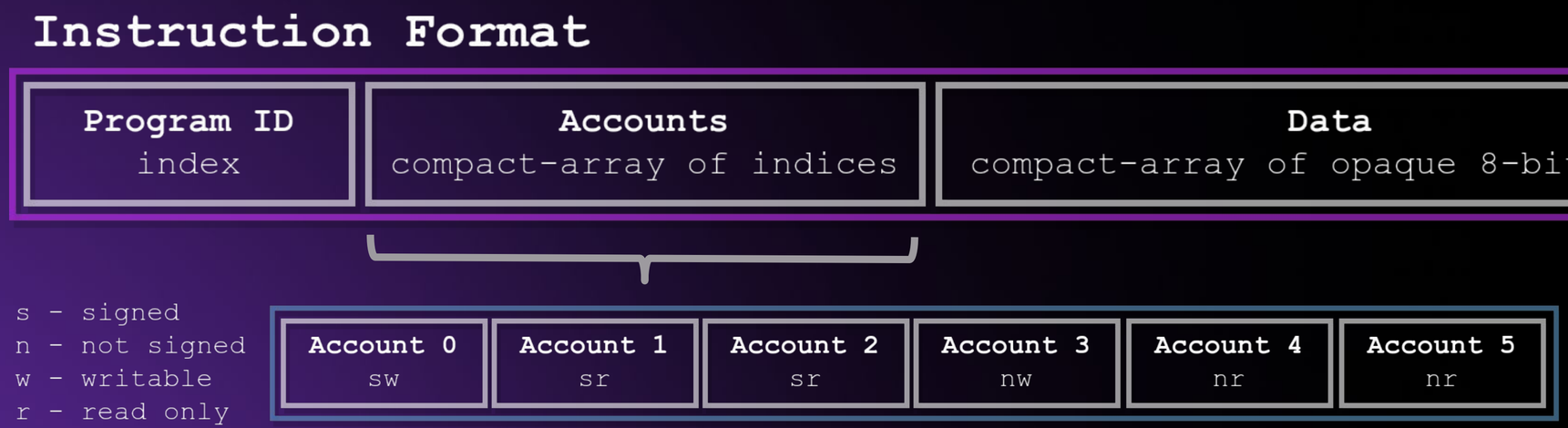
Accounts

- Solana accounts:
  - Program accounts
  - Data accounts
- Program accounts do not store state!
- Only a data account's owner can modify its data and subtract lamports.
- To prevent an account from being deleted, you must pay rent.

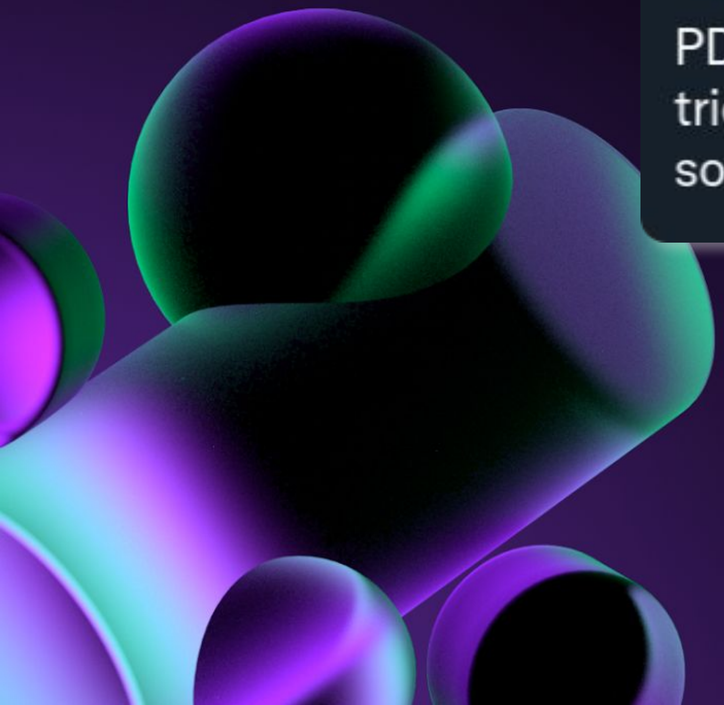


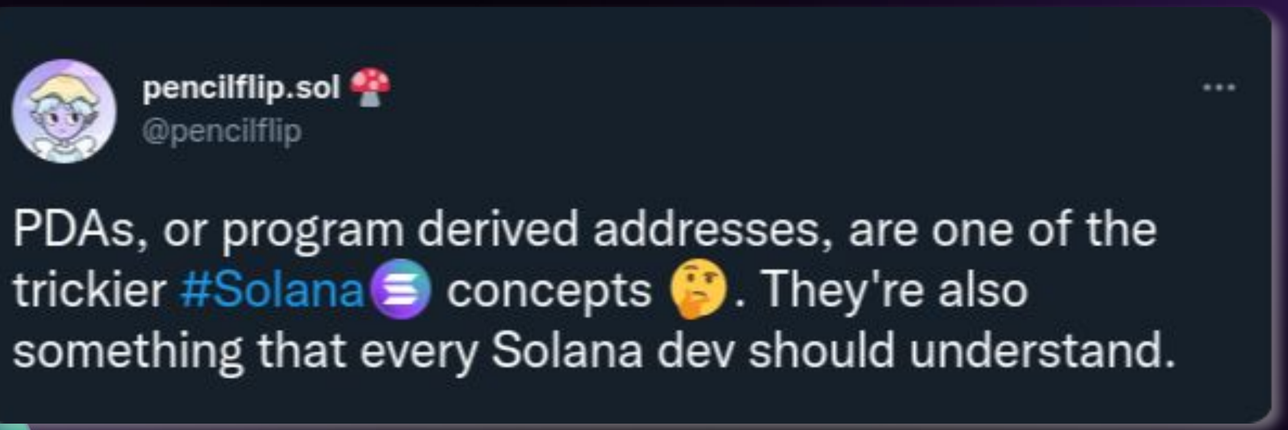
Transactions

- The basic operational unit on Solana is an instruction.
- One or more instructions can be bundled into a transaction.
- Instructions in one transaction are processed in order and atomically.
- You must forward-declare every account you intend to read from or write to.
  - be aware of potential **SECURITY RISKS**

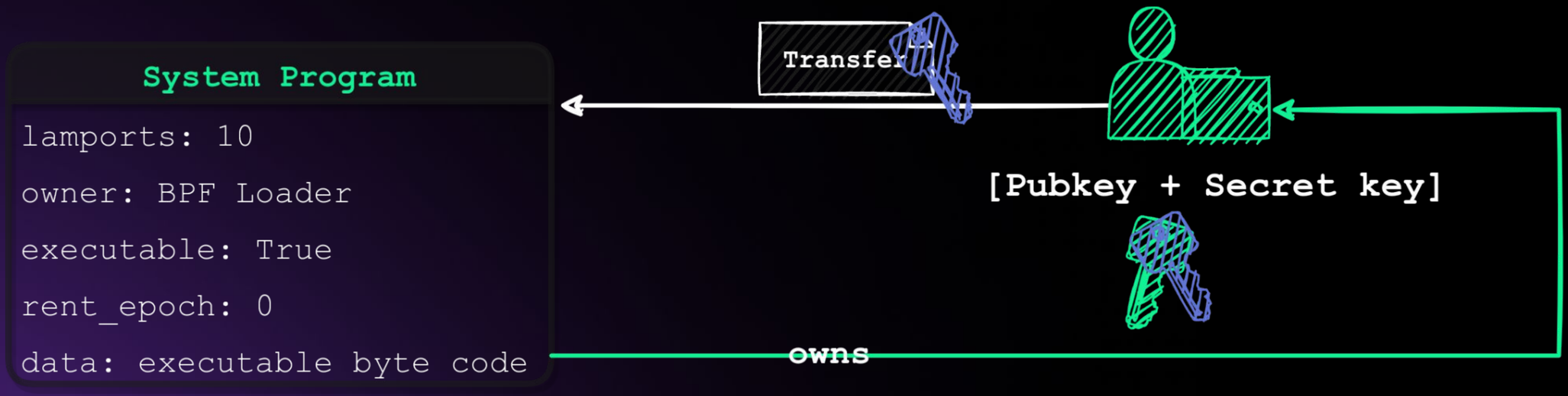


PDAs

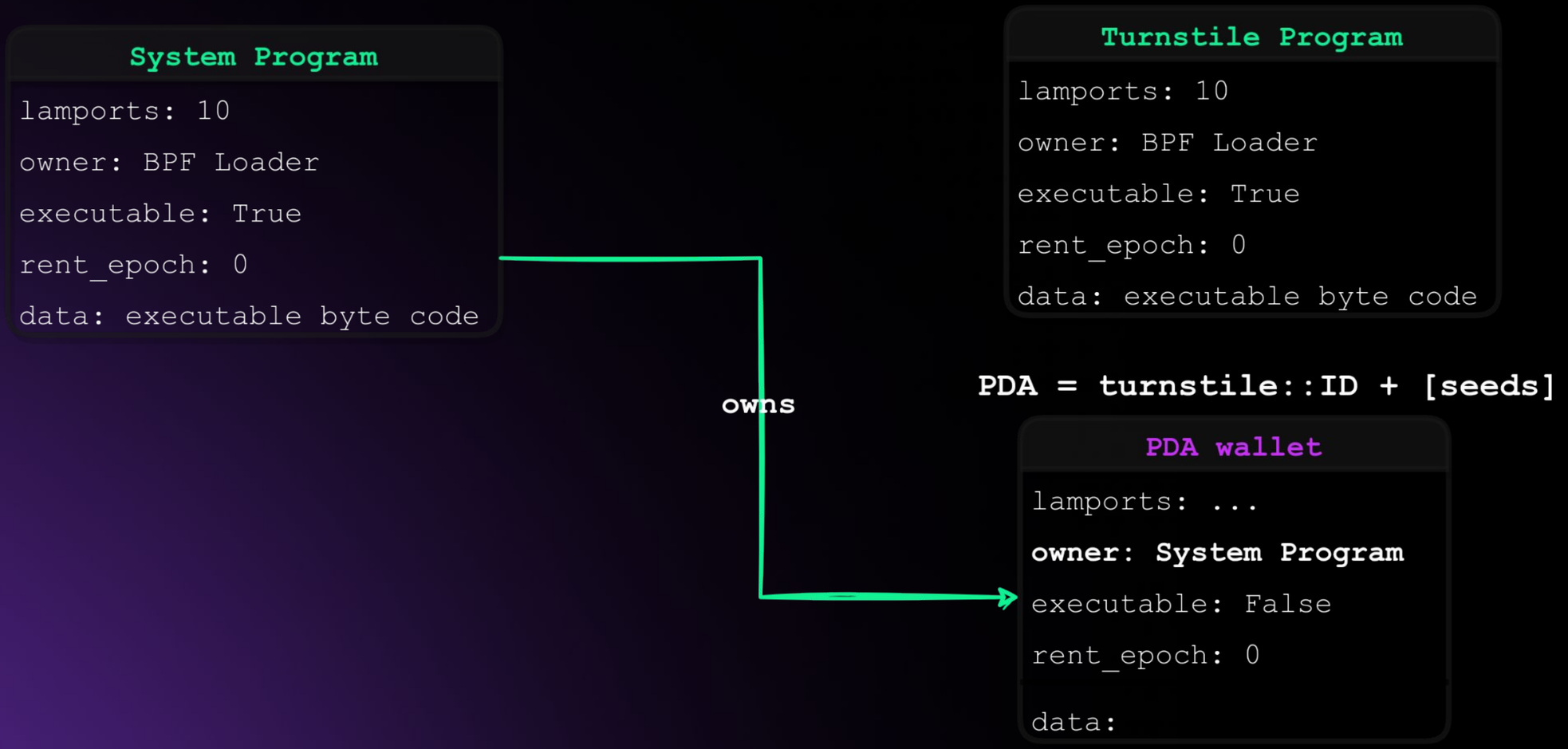



pencilflip.sol
@pencilflip
PDAs, or program derived addresses, are one of the trickier #Solana concepts. They're also something that every Solana dev should understand.

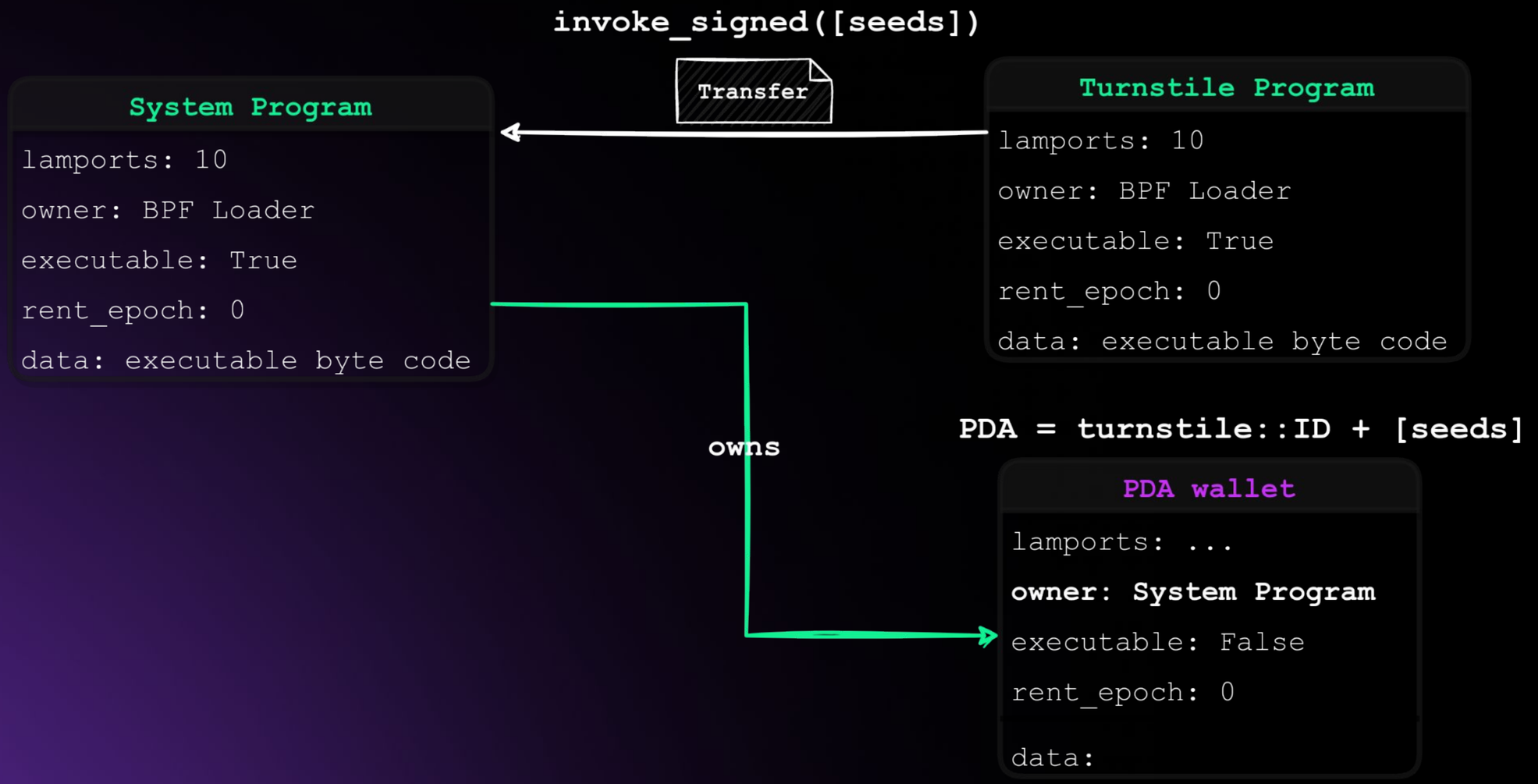
PDAs motivation



PDAs (invoke_signed)



PDAs (invoke_signed)

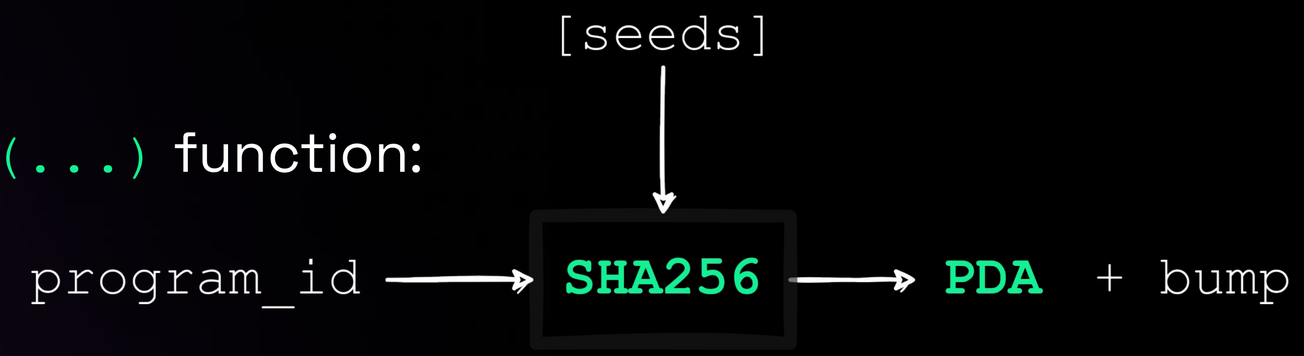


PDA

- **Programs want to be able to sign accounts for cross-program invocation.**

- PDA = program derived address
- PDA account = an account whose address is a PDA.

- How does a PDA get derived?
  - `Pubkey::find_program_address(...)` function:



- PDA is bumped off the Ed25519 elliptic curve. Hence, there is no private key.

PDAs (deterministic addresses)

```rust
pub fn push(
  turnstile_program: Pubkey,
  state: Pubkey
) -> Instruction {
    Instruction {
        program_id: turnstile_program,
        accounts: vec![AccountMeta::new(state, false)],
        data: TurnstileInstruction::Push.try_to_vec().unwrap()
    }
}
```

PDAs (deterministic addresses)

```rust
// function which builds the Push instruction
pub fn push(
  turnstile_program: Pubkey,
) -> Instruction {
    let (state, bump) =
        Pubkey::find_program_address(
          &[b"STATE"],
          turnstile_program
        );
    Instruction {
        program_id: turnstile_program,
        accounts: vec![AccountMeta::new(state, false)],
        data: TurnstileInstruction::Push
                .try_to_vec()
                .unwrap(),
    }
}
```

ackee blockchain

PDAs (hash map)

[Pubkey1]

[Pubkey2]

[Pubkey3]

Program

lamports: 10

owner: BPF Loader

executable: True

rent_epoch: 0

data: executable byte code

[Pubkey1] + Program ID

User Account

lamports: ...

owner: Program

executable: False

rent_epoch: 0

data:

```
User {
 balance: [u64],
 ...
}
```

[Pubkey2] + Program ID

User Account

lamports: ...

owner: Program

executable: False

rent_epoch: 0

data:

```
User {
 balance: [u64],
 ...
}
```

User Account

lamports: ...

owner: Program

executable: False

rent_epoch: 0

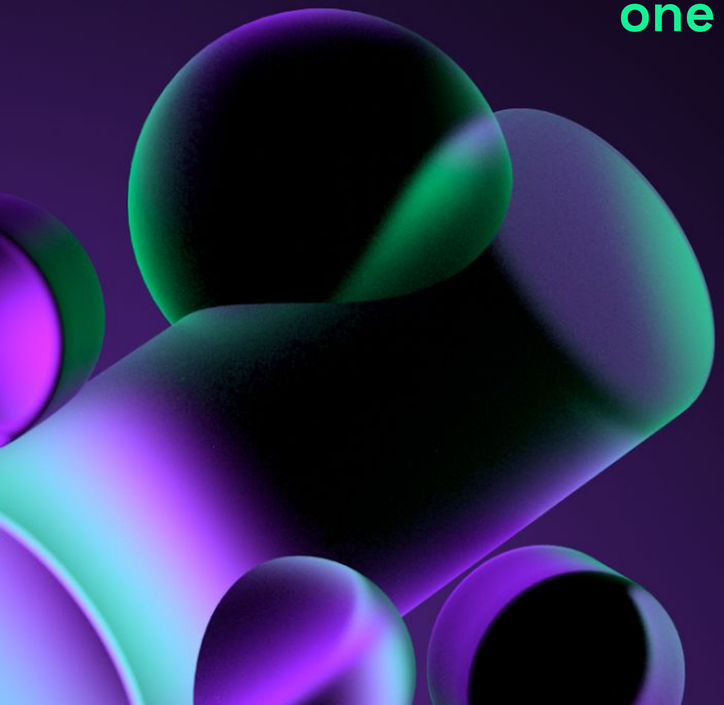
data:

```
User {
 balance: [u64],
 ...
}
```

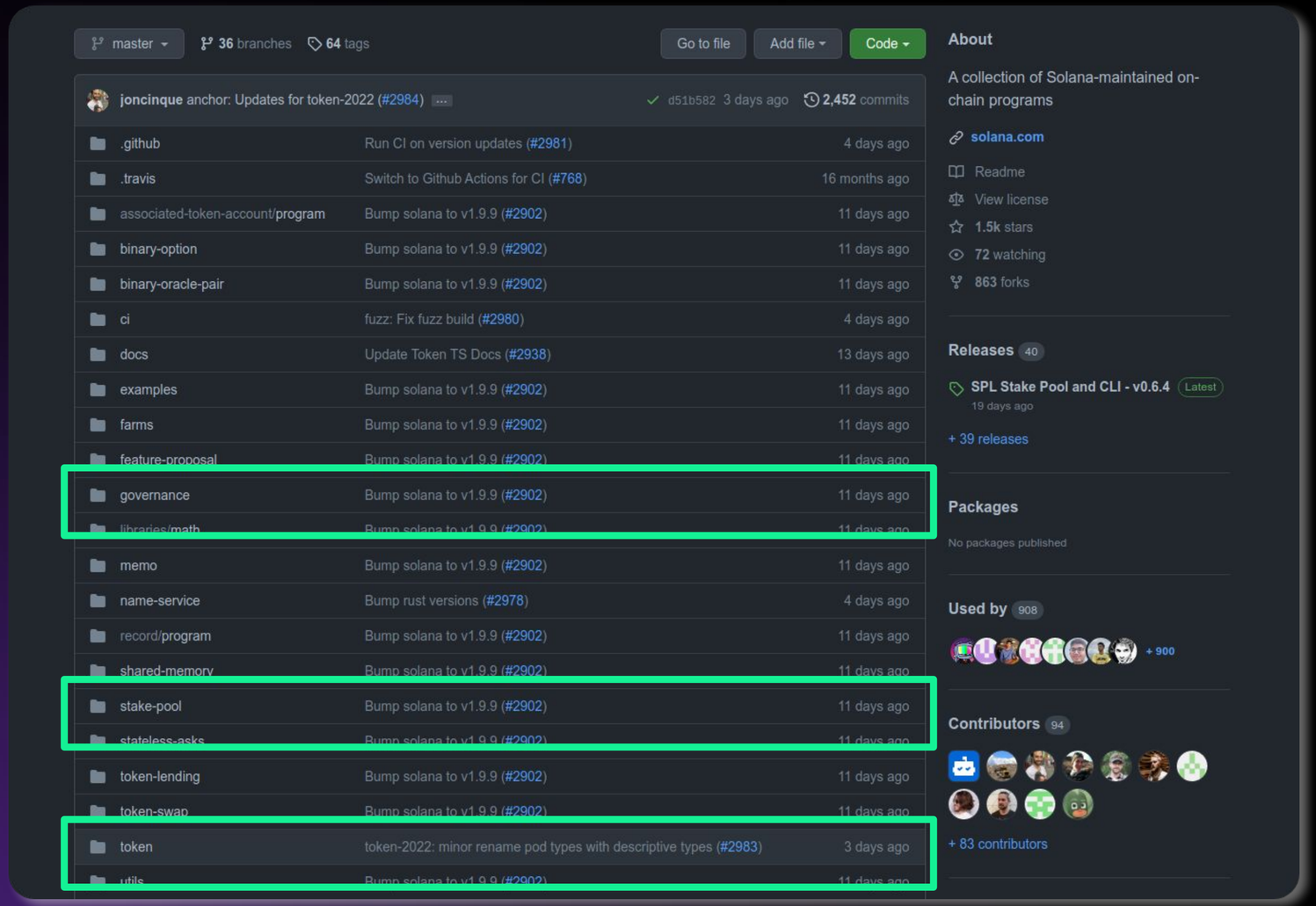
[Pubkey3] + Program ID

Solana Program Library

The advantage of the program/account model is that you can have **one generic program** that operates on **various data**.



Solana Program Library (SPL)



https://github.com/solana-labs/solana-program-library

Solana Tokens

ERC20 token = a token whose smart contract uses the ERC20 standard

- Fungible tokens use the ERC20 standard.
- Smart contract template to create a new token.
- To create a new token, you deploy ERC20 smart contract on chain.

SPL token = a token created with the Solana token program

- Every SPL token has a standard set of functionality.
- On Solana, there is a single token program (spl-token-program).
- To create a new token, you just call a `create_mint` instruction of the token program.

Solana Tokens

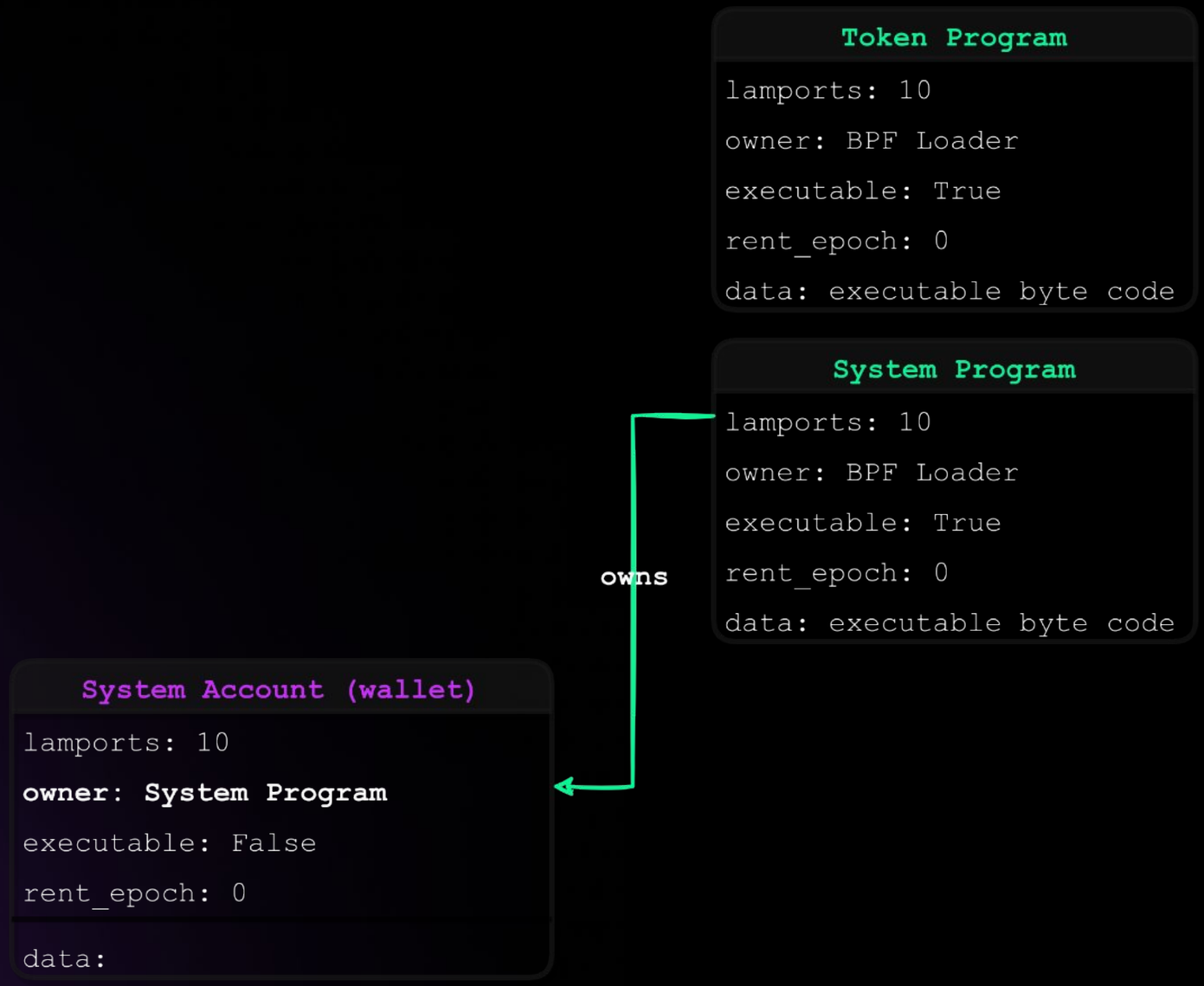
Token Program

lamports: 10
owner: BPF Loader
executable: True
rent_epoch: 0
data: executable byte code

System Program

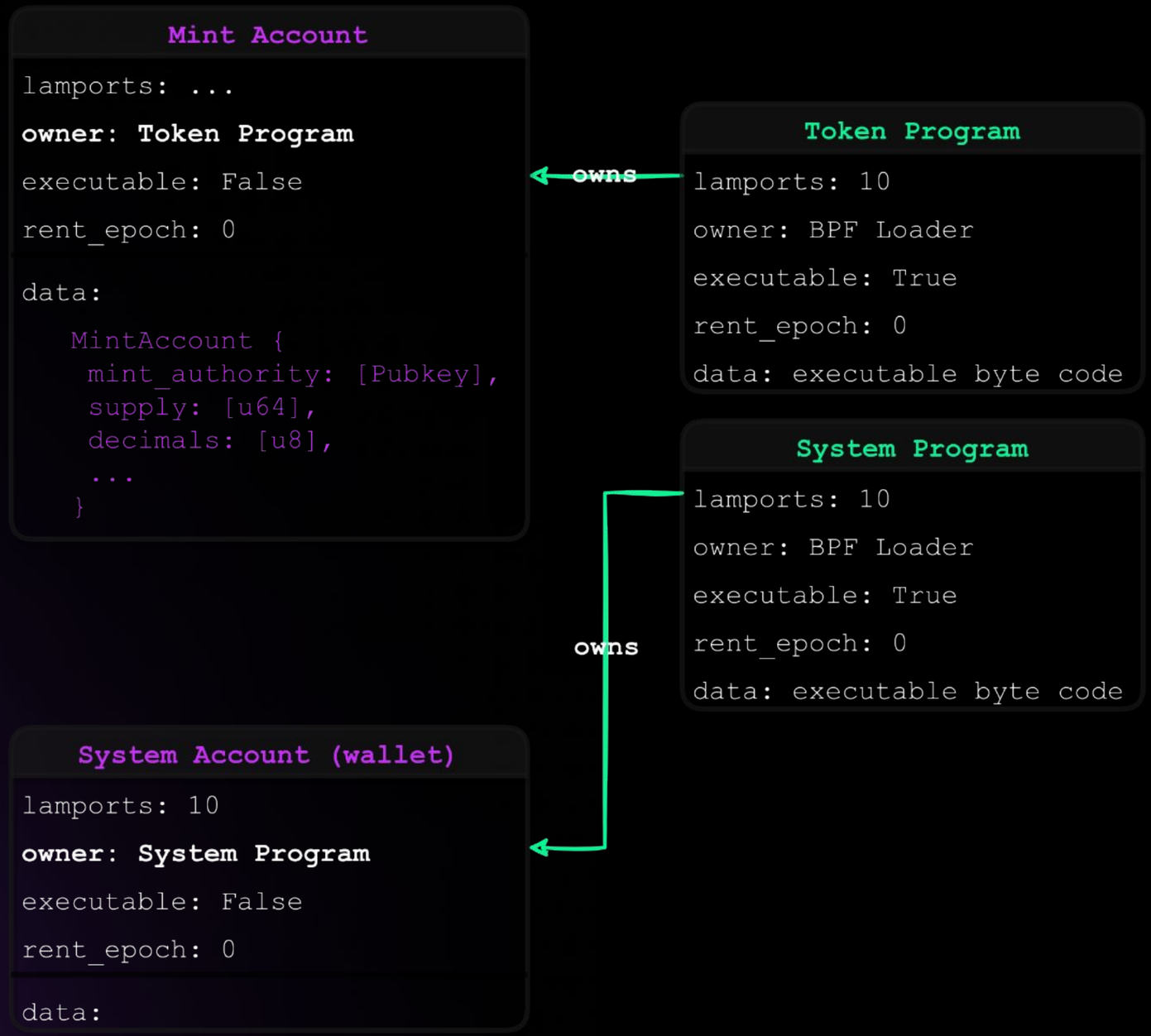
lamports: 10
owner: BPF Loader
executable: True
rent_epoch: 0
data: executable byte code

Solana Tokens



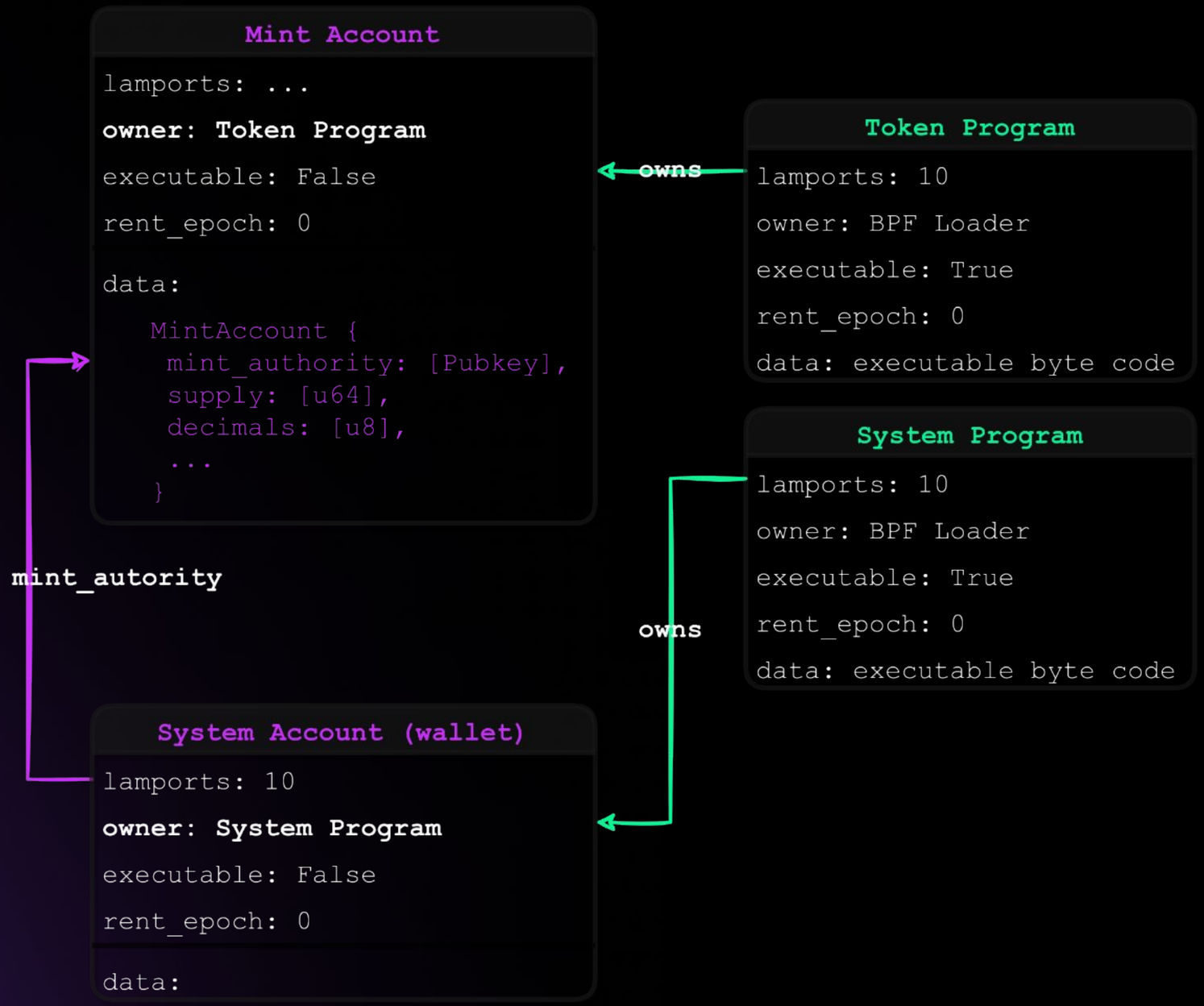
ackee blockchain

Solana Tokens



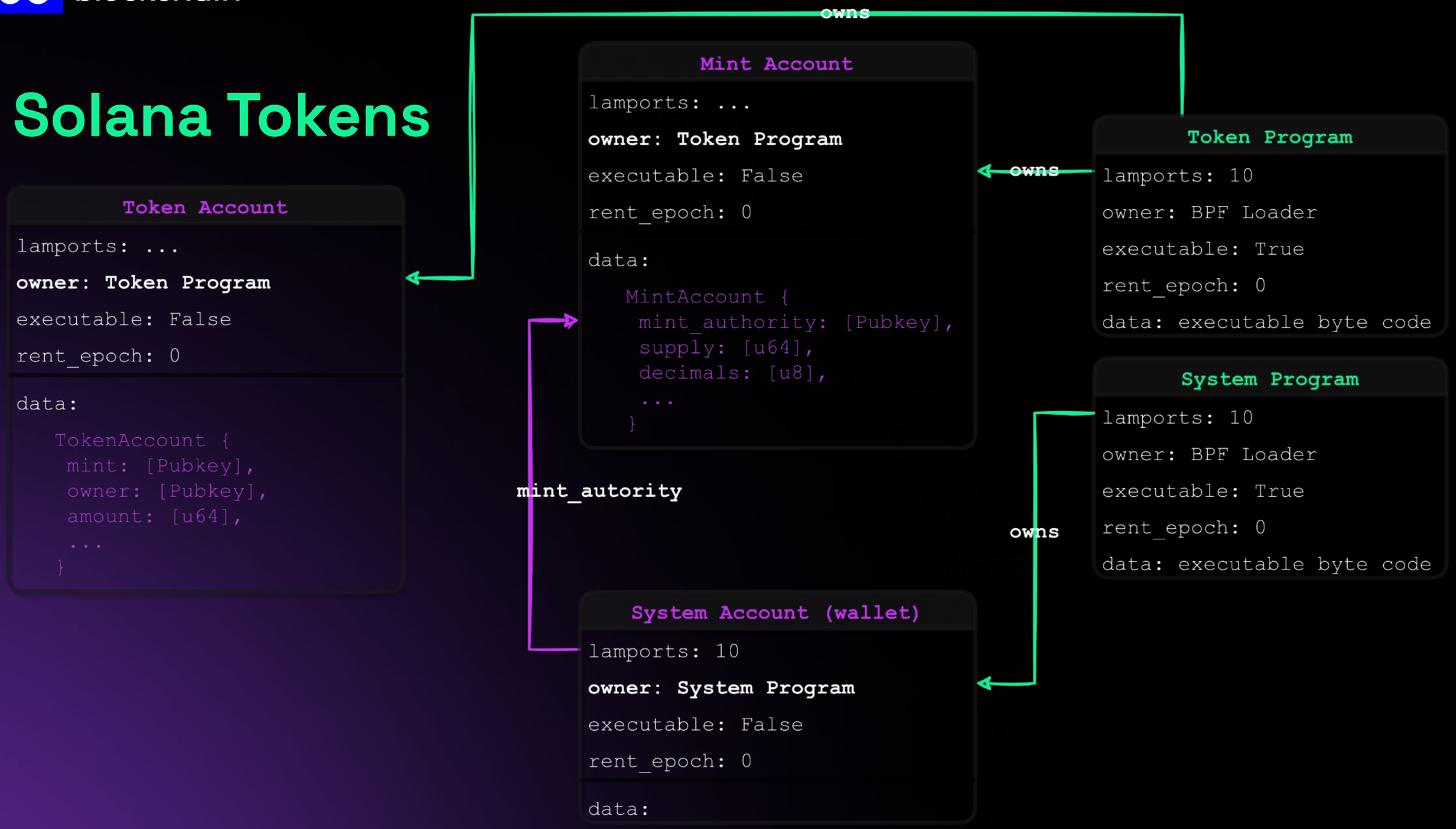
ackee blockchain

Solana Tokens



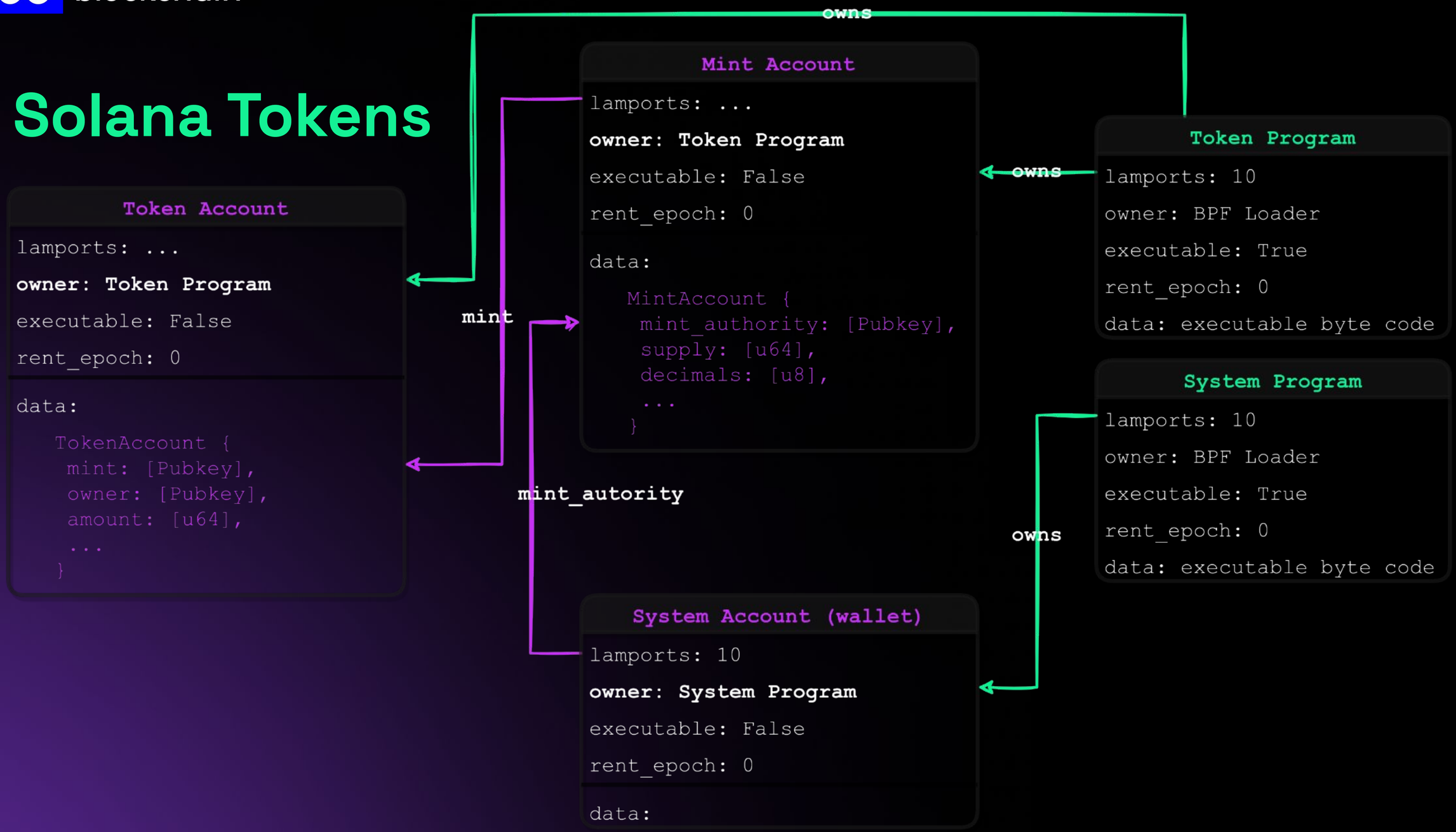
ackee blockchain

Solana Tokens



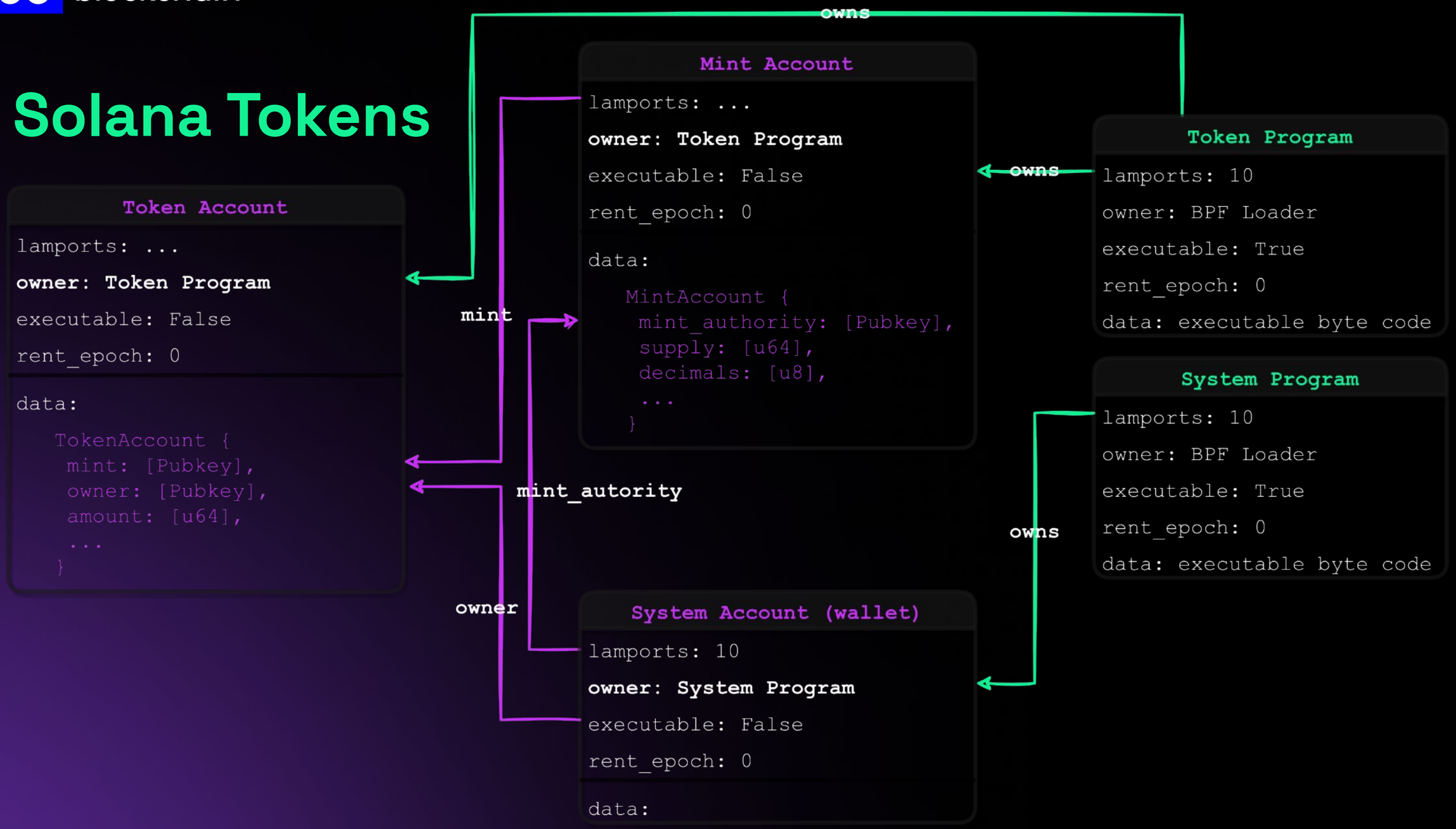
ackee blockchain

Solana Tokens

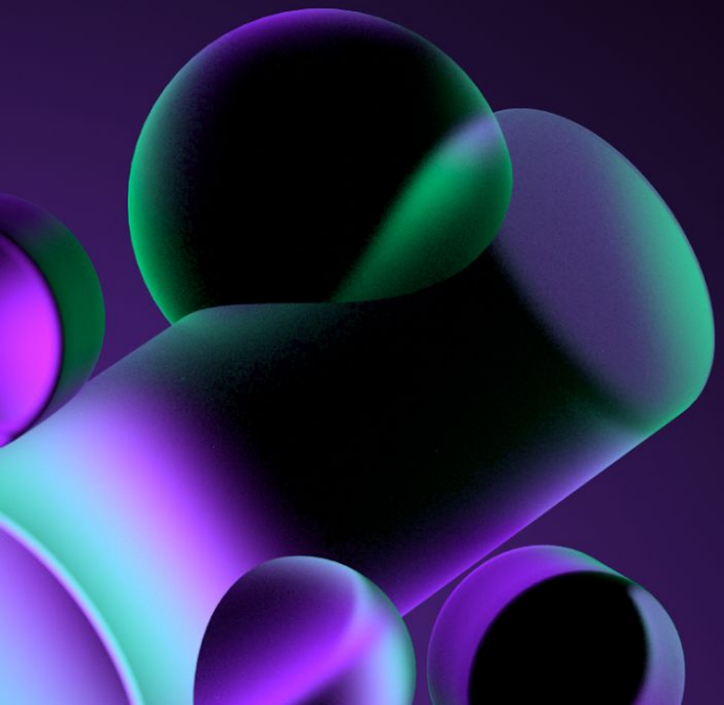


ackee blockchain

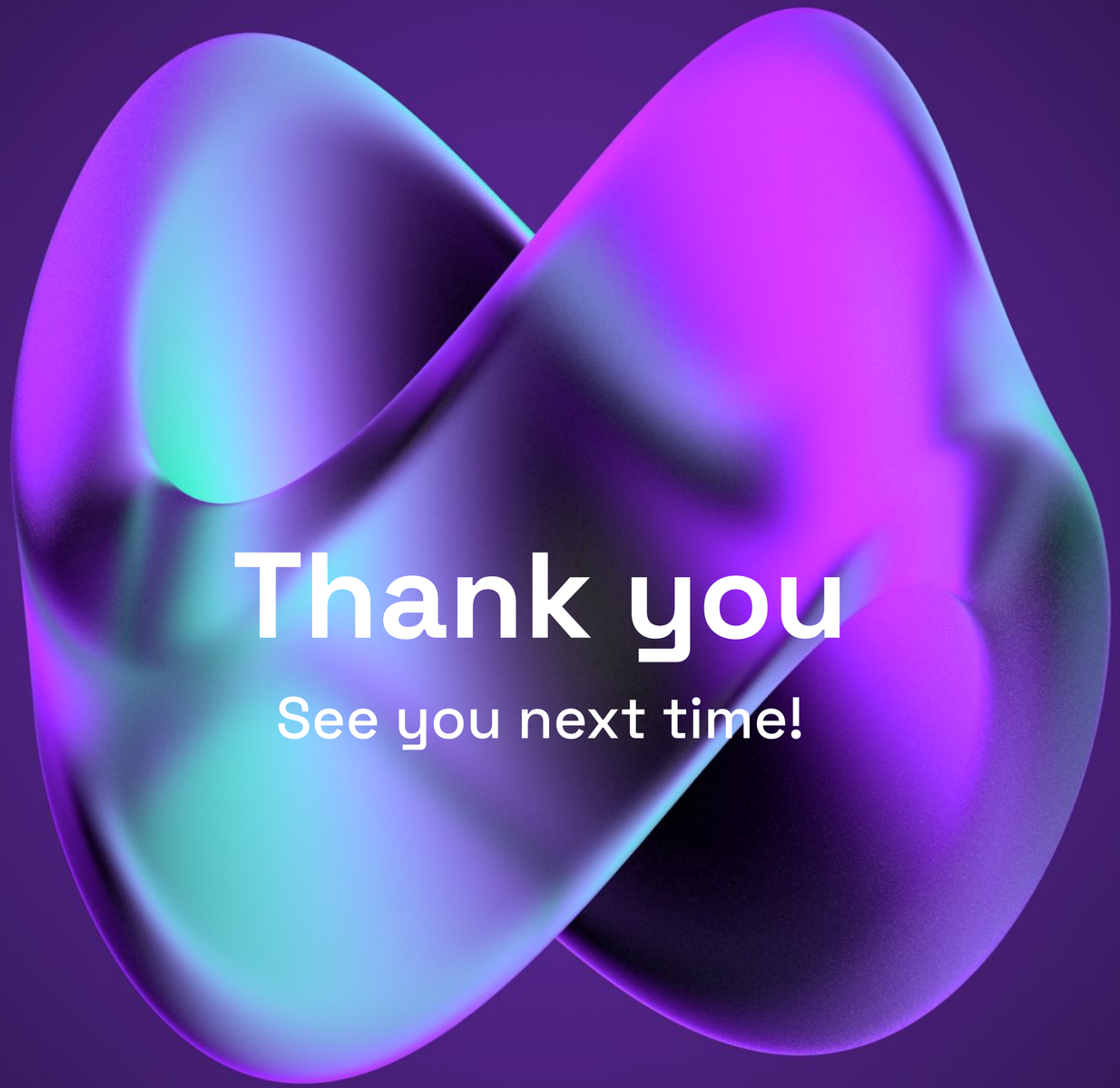
Solana Tokens



Hands on example



ackee blockchain



Thank you

See you next time!