Unlocking Privacy on Solana

Token2022 Confidential Transfer Extension

SOLANA

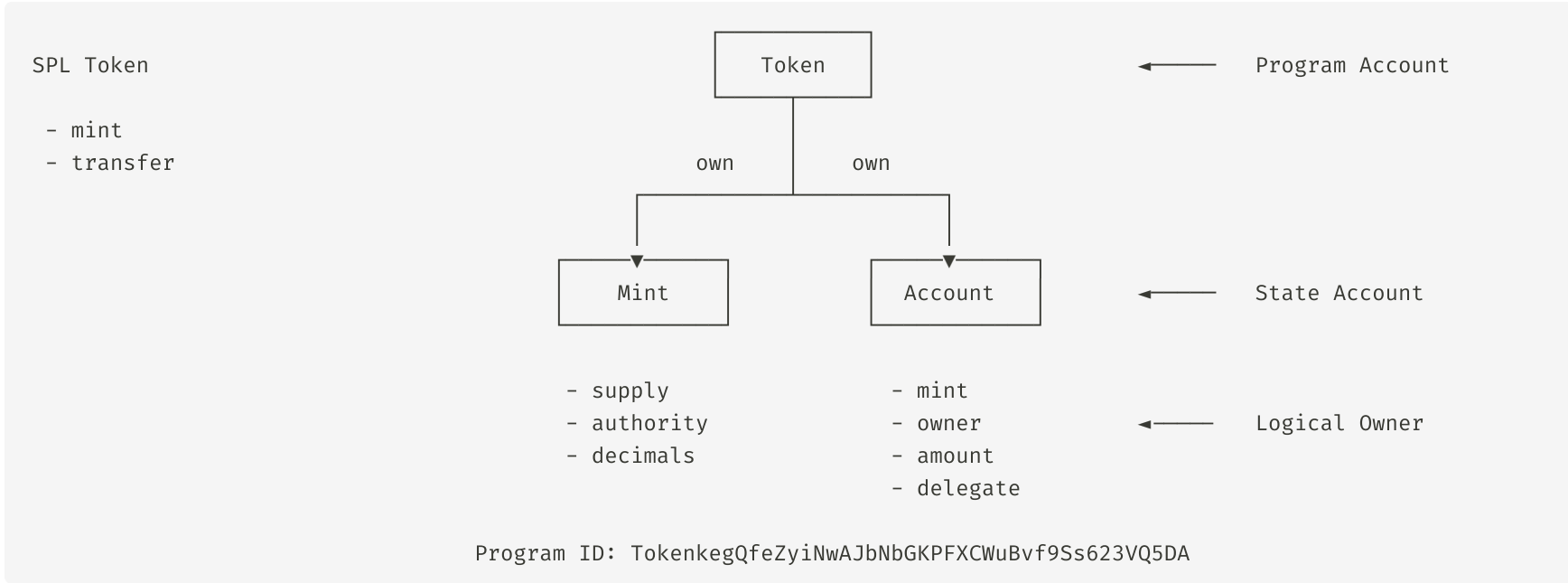
Privacy for Transactions, Compliance for All

Presented by pupplecat | Solana Enthusiast | May 5, 2025

Agenda

```
1. Solana & Tokens

2. Why Privacy?

3. Token 2022

4. Confidential Transfers

5. Cryptography

6. Use cases

7. Live Demo

8. Q & A
```

SPL Token


SPL Token
- mint
- transfer
Token
Program Account
own
own
Mint
Account
State Account
- supply
- authority
- decimals
- mint
- owner
- amount
- delegate
Logical Owner
Program ID: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA

Why Privacy in Blockchain?

- **Public Ledgers**: All amounts visible
- **Need for Confidentiality**:
  - Protect sensitive payments
  - Enable institutional adoption
  - Meet regulatory compliance
- **Confidential Transfer**: Hides amounts, not identities

"Privacy for payments, transparency for regulators."

Token2022: Beyond SPL Token

```
SPL Token 2022

 - confidential transfer
 - fees
 - hooks
 - metadata
 - etc.

                                          Token 2022                    <------   Program Account
                                                |
                                          own       own
                                                |
                                       ---------------
                                       v             v
                                    Mint          Account               <------   State Account

                              -     *   Ext #1   *     *   Ext #1   *     -
                              |     **************     **************     |
                              |     *    ...     *     *    ...     *     |
                   not extendable   **************     **************     |   extendable
                              |     *   Ext #n   *     *   Ext #n   *     |
                              -     **************     **************     -

                                   Program ID: TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb
```

Confidential Transfer Extension

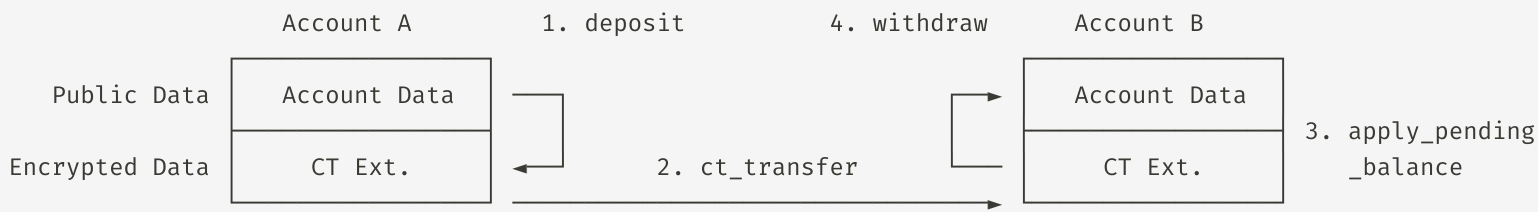
```
Confidential Transfer


  What It Does :
       - Hides transfer amounts
       - Keeps sender/receiver public
       - Optional auditor key

  Workflow :
       1. Create mint with extension
       2. Configure accounts
       3. Deposit to encrypted balance
       4. Transfer confidentially
       5. Apply/withdraw balance
```

Mint

Account Data
CT Ext.

Confidential mint is optional

- confidential_mint
- confidential_burn



Cryptography Behind It

Zero-Knowledge Proofs in CT

- **CiphertextValidityProof**: Validates encrypted transfer
- **EqualityProof**: Matches amounts via commitments (not direct compare)
- **RangeProof**: Ensures new balance is 0 to 2^{64}

Role of ElGamal Key

- **Encryption**: Hides balances (asymmetric)
- **Homomorphic**: Enables encrypted additions
- **Stored**: Public key in token account

Role of AE Key

- **Owner Access**: Encrypts decryptable balance
- **Updates**: Refreshed on key operations
- **Privacy**: Only owner decrypts; auditors cannot

Auditor Role

- **Decrypt Amount**: Views transfer amount
- **Validate Proofs**: Checks ZK proof integrity
- **No Balance Access**: Cannot see full balances

CT Transfer Step-by-Step (Unencrypted View)

```
Zero Knowledge Proof in Confidential Transfer


   Account A has 100 USDC and is transfering 20 USDC to Account B


                Account A                                                    Account B

before :    - available_balance = 100    ⟹   transfer_amount = 20    ⟹       - available_balance = 0
                                                                             - pending_balance = 0
                                          ............. ...................
                                          . Zero-knowledge proof          .
                                          .   - Ciphertext validity proof .
                                          .   - Equality proof            . ⟵ Verified by zk elgamal program
                                          .   - Range proof               .
                                          ..................................

after :     - available_balance = 80                                         - available_balance = 0
                                                                             - pending_balance = 20  ✅


ZK ElGamal Proof Program ID: ZkE1Gama1Proof11111111111111111111111111111
```

Use Cases & Benefits

- **Use Cases**:
  - Stablecoins (Paxos USDP)
  - Payroll (hidden salaries)
  - B2B settlements
  - Compliance (auditor keys)
- **Benefits**:
  - Native integration
  - Scalable on Solana
  - Compliance-friendly
- **Challenges**:
  - Limited wallet support
  - Extension conflicts

Live Demo: Hiding the Amount

- **Objective**: Create a token and transfer it confidentially, showing the amount is encrypted.
- **Tools**: Solana CLI, local validator
- **Github**: pupplecat/token-2022-confidential-transfer-example
- **Steps**:
  1. Create confidential mint
  2. Set up accounts
  3. Deposit to encrypted balance
  4. Transfer (amount hidden)
  5. Check Solana Explorer

Watch the amount disappear!

Book a free demo

Key Takeaways

```
- Privacy:    Hides amounts with Confidential Transfer
- Secure:     ElGamal encryption, Bulletproofs
- Practical:  Stablecoins, payroll, compliance
- Accessible: Easy with Solana CLI
```

Q&A and Resources

Ask Away!

What's on your mind? Use the chat!

Resources

- Solana Docs
- GitHub
- CLI Guide