Project presentation

Python SDK for NexRes

By: Arindaam Roy, UC Davis
Guided By: Prof. Mohammad Sadoghi, UC Davis
01

Motivation and Objectives
SDK: a precursor to NFT Marketplace

- What is NexRes?
  - Next generation of ResilientDB (ResDB)
    - High Throughput Yielding Permissioned Blockchain Fabric
  - A consensus engine
    - core consensus protocol is based on a highly optimized PBFT
  - A key-value store with durable storage
  - Written in C++
What is needed for NFT Marketplace?
- Easy way to create and transfer asset
- Validation of transactions
- Support for modern backend languages like python
Project Contributions
UTXO model on NexRes

1. Python SDK
   - Prepares the Tx
   - Signs/fulfills the Tx
   - Sends the Tx over REST endpoints

2. KV interface
   - REST endpoints in C++
   - To post a Tx
   - To get a Tx

3. Tx validation
   - Signature
   - Double spend
   - Duplicate Tx
   - etc
Core concepts
Transactions (Tx)

- Enforces a UTXO model
- Encodes information such as:
  - Public keys (Owners)
  - Fulfillment of previous Tx
  - Asset info
- Inspired from BigChainDB transaction spec (BEP-13)
## Tx Structure

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset</td>
<td>Digital or Physical item that cannot be changed once registered. Tokens, Crypto-Currencies, Digital certificates etc.</td>
</tr>
<tr>
<td>Metadata</td>
<td>Mutable part of the asset. Eg. price, age etc.</td>
</tr>
<tr>
<td>Inputs</td>
<td>Contains the signature with the private key of the owner as proof that the owner has created/transfered the asset</td>
</tr>
<tr>
<td>Outputs</td>
<td>Contains the condition that needs to be fulfilled for a transfer transaction. The condition encodes the public key of the owner to imply that its corresponding private key is required for verification</td>
</tr>
<tr>
<td>Operation</td>
<td>CREATE or TRANSFER</td>
</tr>
<tr>
<td>Id</td>
<td>sha256 hash of the tx: 61f7385eda0633220d3d6ad170b740...</td>
</tr>
</tbody>
</table>
$\tau_1$  \quad type(\tau_1) = CREATE

$\tau_2$  \quad type(\tau_2) = CREATE

$\tau_3$  \quad type(\tau_3) = TRANSFER
Tx Validation

- NexRes validates the inputs of aTx
- Ed25519 public-key signature to validate if the output of a tx is fulfilled by the correct owner
- Prevent double spend by checking if (id(Tx_i), index) is part of the input of any committed or enqueued Tx
04

Demo
05

Architecture & Tx Flow
NexRes Architecture
Tx Flow

Tx Preparation → Tx Fulfillment → Tx Verification → Tx Commitment
Challenges and Future work
Challenges

- Securely storing private keys
  - The SDK can generate private and public keys but they need to be secure stored
- Validation is a python binding (makes it slow)
  - C++ does not have well maintained cryptoconditions libs
Future work

- Validation in C++ (ongoing work)
- Using a persistence storage which allows for complex queries
- Requiring the signatures of both current and future owners for creation and transfer of assets
- Explore ResDB network with the sdk
- Package the SDK to PyPI
Thank You!