Account and Transaction Model in Klaytn
Lead of Platform & SDK, Ground X
- Account model
- Transaction model
- Improving dev/test environment

Software Engineer, Samsung Electronics
- Improving Tizen development environment
- Developing AI software stack for mobile

Ph.D. in Computer Science and Engineering
- System software
- Computer architecture
- Parallel programming model
- GPGPU
· What are accounts and transactions?

· Usability considerations
  - User’s perspective
  - Service provider’s perspective
  - Platform developer’s perspective

· Account model

· Transaction model

· Conclusion
Account and Transaction

- Account
  - A data structure storing information of users and contracts
    - Nonce
    - Balance
    - CodeHash
    - StorageRoot
Account and Transaction

- **Account**
  - A data structure storing information of users and contracts
    - Nonce
    - Balance
    - CodeHash
    - StorageRoot

- **Transaction**
  - A unit of changing states of Klaytn blockchain platform
  - Various functions
    - Value transfer
    - Smart contract deploy
    - Smart contract execution
Account and Transaction

- Account
  - A data structure storing information of EOAs and contracts
    - Nonce
    - Balance
    - CodeHash
    - StorageRoot

- Transaction
  - A unit of changing states of Klaytn blockchain platform
  - Various functions
    - Value transfer
    - Smart contract deploy
    - Smart contract execution

For mass adoption, need better usability!
For better usability, need better acc/tx model!
Usability Considerations

- User’s perspective
- Service provider’s perspective
- Platform developer’s perspective
Usability Considerations for Users
Usability Considerations for Users

- User’s perspective
  - Exposed private key
  - Increasing security of the account
Relation between Key Pair and Address

Private Key: secp256k1 with ECDSA
Public Key: 0xA29a0AEBb4C537945699A4EF712b83981141a79
Address: 0xA29a0AEBb4C537945699A4EF712b83981141a79
Exposed Private Key

---

What if your private key is exposed?

---

Exosed Private Key
Exposed Private Key

What if your private key is exposed?

Exposed Private Key

you need to create another account

NEW
Exposed Private Key

What if your private key is exposed?

Exrose Private Key → you need to create another account → Tell the new address to your friends.
Exposed Private Key

What if your private key is exposed?

1. Exposed Private Key
2. You need to create another account
3. Tell the new address to your friends. Transfer your assets on BApps
Exposed Private Key

What if your private key is exposed?

? ?
Exposed Private Key

you need to create another account

NEW

Tell the new address to your friends. Transfer your assets on BApps

Lost your transaction history

NEW
Exposed Private Key

What if your private key is exposed?

- Exposed Private Key
- You need to create another account
- Tell the new address to your friends. Transfer your assets on BApps
- Lost your transaction history
Exposed Private Key

Address == Bank account number
Private key == Password

What if your private key is exposed?

Exposed Private Key

You need to create another account

Lost your transaction history

Tell the new address to your friends. Transfer your assets on BApps
Exposed Private Key

Solution: Make private key changeable

Address == Bank account number
Private key == Password

What if your private key is exposed?

Exposed Private Key

? ?

you need to create another account

NEW

Lost your transaction history

Tell the new address to your friends.
Transfer your assets on BApps

Tell

+
Decoupling Key Pair from Address

Private Key → Public Key → Address

secp256k1 with ECDSA

0xA29a0AEBb4cC53794569
9A4Ef712b83981141a79
Public key in Account

<table>
<thead>
<tr>
<th>Nonce</th>
<th>Balance</th>
<th>Root</th>
<th>CodeHash</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Nonce</th>
<th>Balance</th>
<th>Root</th>
<th>CodeHash</th>
<th>Pubkey</th>
</tr>
</thead>
</table>
Usability Considerations for Users

- User’s perspective
  - Exposed private key
  - Increasing security of the account
Increasing the Security of Your Account

- Traditional solution
  - Multisig smart contract
Increasing the Security of Your Account

● Traditional solution
  ○ Multisig smart contract

● Problems of multisig smart contracts
  ○ What is smart contract?
  ○ How to deploy it?
  ○ How to execute it?
  ○ How to guarantee the contract is secured?
Increasing the Security of Your Account

- Traditional solution
  - Multisig smart contract

- Problems of multisig smart contracts
  - What is smart contract?
  - How to deploy it?
  - How to execute it?
  - How to guarantee the contract is secured?

- With Klaytn
  - Native support of multisig
Multisig in Account

- Nonce
- Balance
- Root
- CodeHash

- Nonce
- Balance
- Root
- CodeHash

- Key

- Public key
- Multiple public keys
SUMMARY: Usability Considerations for Users

● User's perspective
  ○ Exposed private key
  ○ Increasing security of the account

● Solution
  ○ Changeable private keys
  ○ Native support of multisig
Usability Considerations for Service Providers
Usability Considerations for Service Providers

● Service provider’s perspective
  ○ Transaction fee
  ○ Separation of permission
Fee Delegation

- Transaction fee
  - Paid on every action of a user

- Normal services
  - No fee for common actions
  - Trial period
Fee Delegation

- Transaction fee
  - Paid on every action of a user

- Normal services
  - No fee for common actions
  - Trial period

- With Klaytn
  - Transaction fee can be paid by service providers
  - Services can take various user acquisition strategies
## Fee Delegated Transactions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountNonce</td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td></td>
</tr>
<tr>
<td>GasLimit</td>
<td></td>
</tr>
<tr>
<td>Recipient</td>
<td></td>
</tr>
<tr>
<td>Amount</td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
<tr>
<td>Sender address</td>
<td></td>
</tr>
<tr>
<td>Sender signatures</td>
<td></td>
</tr>
<tr>
<td><strong>Fee payer address</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Fee payer signatures</strong></td>
<td></td>
</tr>
</tbody>
</table>
Fee Delegated Transaction Execution

Sender → FeePayer → Klaytn
Usability Considerations for Service Providers

- Service provider’s perspective
  - Transaction fee
  - Separation of permission
Permissions

- Transferring KLAY
- Deploying a smart contract
- Executing a smart contract
- Updating the account’s data
- Paying transaction fee
Permissions and Roles

- Transferring KLAY
- Deploying a smart contract
- Executing a smart contract
- Updating the account’s data
- Paying transaction fee
Role-based Key Use Case - Fee Delegation

0xA1B2...C3D4

RoleTransaction

Admin

RoleFeePayer

Operator
Role-based Key Use Case - Fee Delegation

0xA1B2...C3D4

RoleTransaction  →  Admin can transfer KLAY.

RoleFeePayer  →  Operator
Role-based Key Use Case - Fee Delegation

0xA1B2...C3D4

- RoleTransaction
- RoleFeePayer

Admin

Operator can pay tx fee.
Role-based Key Use Case - Fee Delegation

0xA1B2...C3D4

- RoleTransaction
- RoleFeePayer

Admin

Operator can pay tx fee.
Operator cannot transfer KLAY.
Role-based Key Use Case - User Account Recovery

0xAAAA...BBBB

RoleTransaction → User

RoleAccountUpdate → Service provider
Role-based Key Use Case - User Account Recovery

0xAAAA...BBBB

RoleTransaction

User can transfer KLAY.

Service provider

RoleAccountUpdate

User can transfer KLAY.
Role-based Key Use Case - User Account Recovery

0xAAAA...BBBB

RoleTransaction

User

RoleAccountUpdate

Service provider can update RoleTransaction Key.
Role-based Key Use Case - User Account Recovery

Service provider can update RoleTransaction Key. Service provider can give a new key to the user.
Role-based Key in Account

- Nonce
- Balance
- Root
- CodeHash

- Nonce
- Balance
- Root
- CodeHash

Key

- Public key
- Multiple public keys
- Role-based keys
  - Transaction
  - Account update
  - Fee payer
SUMMARY: Usability Considerations for Service Providers

- Service provider’s perspective
  - Transaction fee
  - Separation of permission

- Solution
  - Fee Delegated transactions
  - Native support of role-based keys
Usability Considerations for Platform Developers
Usability Considerations For Platform Developers

- Platform developer’s perspective
  - Easy to extend
  - Easy to analyze
<table>
<thead>
<tr>
<th>Account Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonce</td>
<td></td>
</tr>
<tr>
<td>Balance</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td></td>
</tr>
<tr>
<td>Root</td>
<td></td>
</tr>
<tr>
<td>CodeHash</td>
<td></td>
</tr>
</tbody>
</table>
# Account Type

<table>
<thead>
<tr>
<th>Nonce</th>
<th>Balance</th>
<th>Key</th>
<th>Root</th>
<th>CodeHash</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Type: Contract**

<table>
<thead>
<tr>
<th>Nonce</th>
<th>Balance</th>
<th>Key</th>
<th>Root</th>
<th>CodeHash</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Type: User**

<table>
<thead>
<tr>
<th>Nonce</th>
<th>Balance</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © Ground X. All rights reserved.
Account Type

Type: User
Nonce
Balance
Key
Root
CodeHash

Type: Contract
Nonce
Balance
Key
Root
CodeHash

Easy to extend
Easy to analyze
Account Type

- Type:Contract
  - Nonce
  - Balance
  - Key
  - Root
  - CodeHash

- Type:User
  - Nonce
  - Balance
  - Key
  - Root
  - CodeHash

Easy to extend
Easy to analyze

Storage cost reduced
# Transaction Type

<table>
<thead>
<tr>
<th>Transaction Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountNonce</td>
</tr>
<tr>
<td>Price</td>
</tr>
<tr>
<td>GasLimit</td>
</tr>
<tr>
<td>Recipient</td>
</tr>
<tr>
<td>Amount</td>
</tr>
<tr>
<td>Payload</td>
</tr>
<tr>
<td>Signature(V,R,S)</td>
</tr>
</tbody>
</table>

Copyright © Ground X. All rights reserved.
Transaction Type

<table>
<thead>
<tr>
<th>AccountNonce</th>
<th>Price</th>
<th>GasLimit</th>
<th>Recipient</th>
<th>Amount</th>
<th>Payload</th>
<th>Signature(V,R,S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type: ValueTransfer

<table>
<thead>
<tr>
<th>AccountNonce</th>
<th>Price</th>
<th>GasLimit</th>
<th>Recipient</th>
<th>Amount</th>
<th>Payload</th>
<th>Signature(V,R,S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type: SmartContractExecution

<table>
<thead>
<tr>
<th>AccountNonce</th>
<th>Price</th>
<th>GasLimit</th>
<th>Recipient</th>
<th>Amount</th>
<th>Payload</th>
<th>Signature(V,R,S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © Ground X. All rights reserved.
## Transaction Type

<table>
<thead>
<tr>
<th>Type: Value Transfer</th>
<th>Type: Smart Contract Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountNonce</td>
<td>AccountNonce</td>
</tr>
<tr>
<td>Price</td>
<td>Price</td>
</tr>
<tr>
<td>GasLimit</td>
<td>GasLimit</td>
</tr>
<tr>
<td>Recipient</td>
<td>Recipient</td>
</tr>
<tr>
<td>Amount</td>
<td>Amount</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
<tr>
<td>Signature(V,R,S)</td>
<td>Signature(V,R,S)</td>
</tr>
</tbody>
</table>

- Easy to extend
- Easy to analyze
# Transaction Type

**Type:** ValueTransfer  
**Type:** SmartContractExecution

<table>
<thead>
<tr>
<th>AccountNonce</th>
<th>Price</th>
<th>GasLimit</th>
<th>Recipient</th>
<th>Amount</th>
<th>Payload</th>
<th>Signature(V,R,S)</th>
</tr>
</thead>
</table>

- **Easy to extend**
- **Easy to analyze**

**Storage cost reduced**
SUMMARY: Usability Considerations For Platform Developers

● Platform developer’s perspective
  ○ Easy to extend
  ○ Easy to analyze

● Solution
  ○ Introduce explicit type fields to
    ■ accounts
    ■ transactions
Account Model
Account Model

**Contract**
- Nonce
- Balance
- Key
  - Root
  - CodeHash

**User**
- Nonce
- Balance
- Key

- Public key
- Multiple public keys
- Role-based keys
  - Transaction
  - Account update
  - Fee payer
Transaction Model
## Transaction Model

<table>
<thead>
<tr>
<th></th>
<th>Basic</th>
<th>Fee Delegation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>TxTypeLegacyTransaction</td>
<td>N/A</td>
</tr>
<tr>
<td>ValueTransfer</td>
<td>TxTypeValueTransfer</td>
<td>TxTypeFeeDelegatedValueTransfer</td>
</tr>
<tr>
<td>ValueTransferMemo</td>
<td>TxTypeValueTransferMemo</td>
<td>TxTypeFeeDelegatedValueTransferMemo</td>
</tr>
<tr>
<td>SmartContractDeploy</td>
<td>TxTypeSmartContractDeploy</td>
<td>TxTypeFeeDelegatedSmartContractDeploy</td>
</tr>
<tr>
<td>SmartContractExecution</td>
<td>TxTypeSmartContractExecution</td>
<td>TxTypeFeeDelegatedSmartContractExecution</td>
</tr>
<tr>
<td>AccountUpdate</td>
<td>TxTypeAccountUpdate</td>
<td>TxTypeFeeDelegatedAccountUpdate</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[https://docs.klaytn.com/klaytn/design/transactions](https://docs.klaytn.com/klaytn/design/transactions)
## Transaction Model

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Basic</th>
<th>Fee Delegation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>TxTypeLegacyTransaction</td>
<td>N/A</td>
</tr>
<tr>
<td>ValueTransfer</td>
<td>TxTypeValueTransfer</td>
<td>TxTypeFeeDelegatedValueTransfer</td>
</tr>
<tr>
<td>ValueTransferMemo</td>
<td>TxTypeValueTransferMemo</td>
<td>TxTypeFeeDelegatedValueTransferMemo</td>
</tr>
<tr>
<td>SmartContractDeploy</td>
<td>TxTypeSmartContractDeploy</td>
<td>TxTypeFeeDelegatedSmartContractDeploy</td>
</tr>
<tr>
<td>SmartContractExecution</td>
<td>TxTypeSmartContractExecution</td>
<td>TxTypeFeeDelegatedSmartContractExecution</td>
</tr>
<tr>
<td>AccountUpdate</td>
<td>TxTypeAccountUpdate</td>
<td>TxTypeFeeDelegatedAccountUpdate</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[https://docs.klaytn.com/klaytn/design/transactions](https://docs.klaytn.com/klaytn/design/transactions)
## Transaction Model

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Basic</th>
<th>Fee Delegation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>TxTypeLegacyTransaction</td>
<td>N/A</td>
</tr>
<tr>
<td>ValueTransfer</td>
<td>TxTypeValueTransfer</td>
<td>TxTypeFeeDelegatedValueTransfer</td>
</tr>
<tr>
<td>ValueTransferMemo</td>
<td>TxTypeValueTransferMemo</td>
<td>TxTypeFeeDelegatedValueTransferMemo</td>
</tr>
<tr>
<td>SmartContractDeploy</td>
<td>TxTypeSmartContractDeploy</td>
<td>TxTypeFeeDelegatedSmartContractDeploy</td>
</tr>
<tr>
<td>SmartContractExecution</td>
<td>TxTypeSmartContractExecution</td>
<td>TxTypeFeeDelegatedSmartContractExecution</td>
</tr>
<tr>
<td>AccountUpdate</td>
<td>TxTypeAccountUpdate</td>
<td>TxTypeFeeDelegatedAccountUpdate</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

https://docs.klaytn.com/klaytn/design/transactions
What’s Next?

● Human-readable address
  ○ 0x1234...CDEF -> colin.klaytn

● More account types

● More transaction types
Conclusion
Conclusion

- Design account and transaction model to enhance usability

- Users
  - Changeable private key
  - Native support of multisig

- Service providers
  - Fee delegation
  - Native support of role-based keys

- Platform Developers
  - Explicit types for accounts and transactions

Find more: https://medium.com/@klaytn.tech
Something More!
Contribute!

- Klaytn organization in Github: [https://github.com/klaytn](https://github.com/klaytn)

<table>
<thead>
<tr>
<th></th>
<th><a href="https://github.com/klaytn/klaytn">https://github.com/klaytn/klaytn</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>caver-js</td>
<td><a href="https://github.com/klaytn/caver-js">https://github.com/klaytn/caver-js</a></td>
</tr>
<tr>
<td>caver-java</td>
<td><a href="https://github.com/klaytn/caver-java">https://github.com/klaytn/caver-java</a></td>
</tr>
<tr>
<td>Klaytn Improvement Proposal (KIP)</td>
<td><a href="https://github.com/klaytn/kips">https://github.com/klaytn/kips</a></td>
</tr>
</tbody>
</table>
WE ARE HIRING!

https://www.groundx.xyz/careers
THANK YOU