Session II : Service Chain

김경업 / Ground X

Horizontal Scaling through Service Chain in Klaytn

김경업, Ethan Kim

Software Engineer, Platform & SDK Team, Ground X
- Klaytn Service Chain 개발
- Klaytn Main-net (Cypress) 개발
- Klaytn Test-net (Aspen, Baobab) 개발

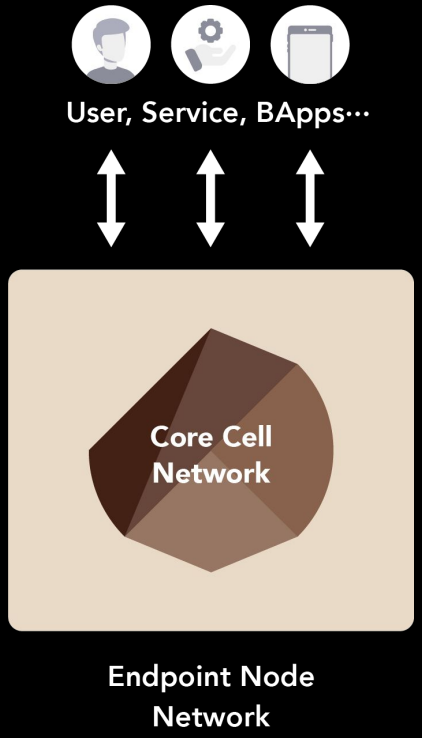
Senior Software Engineer, Samsung Electronics
- Expert Programmer
- 무선 사업부
  - AI Robot / Speaker 선행 개발
- 생활 가전 사업부
  - 가전 IoT 전략 기획, 선행 개발
  - 냉장고 소프트웨어 개발

Seoul National University
- 기계 항공 공학 전공, 석사 (공대공 미사일 유도/제어 세부 전공)

Hanyang University
- 전자 전기 컴퓨터 공학 전공, 학사

TABLE OF CONTENTS

Klaytn Cypress (Mainnet)



· Public Blockchain

누구나 data를 read/write(transaction 발생) 가능

· 성능

1 second Block Time

Instant Finality

4,000 TPS (KLAY transfer 기준)

· Transaction Fee

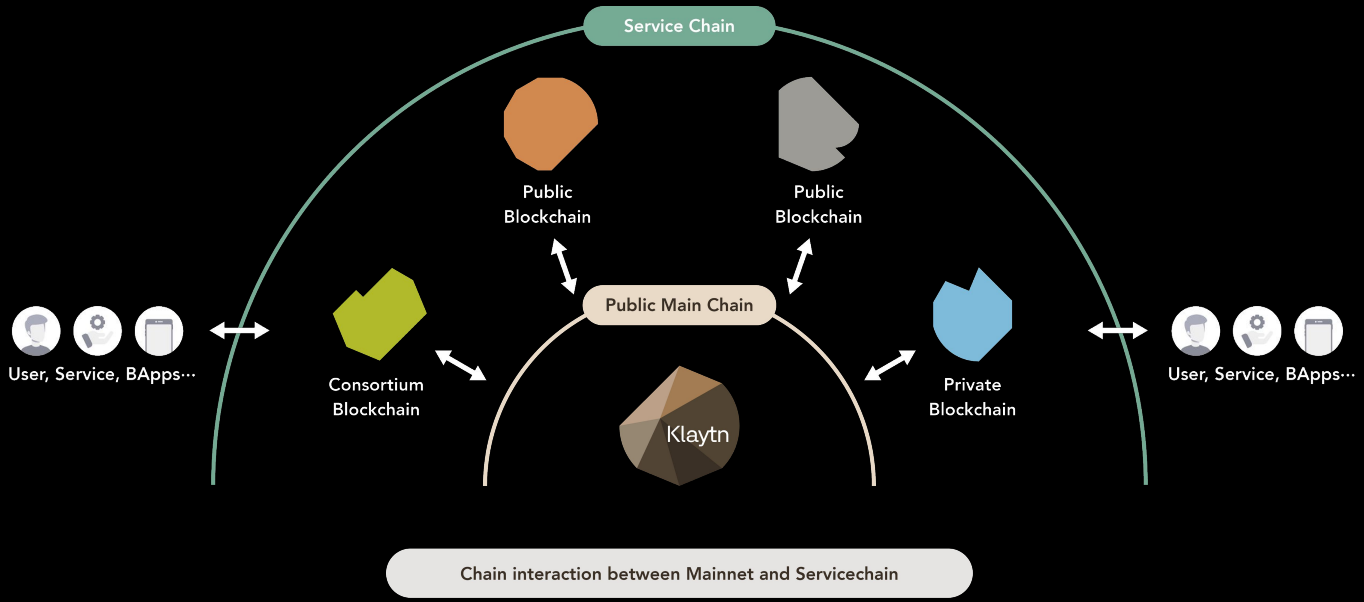
0.000525 KLAY (21,000 gas)

0.11원 (11/6 기준)

Klaytn Service Chain

· 특정 서비스 전용으로 다양한 구성 (Public, Private, Consortium)이 가능한 블록체인

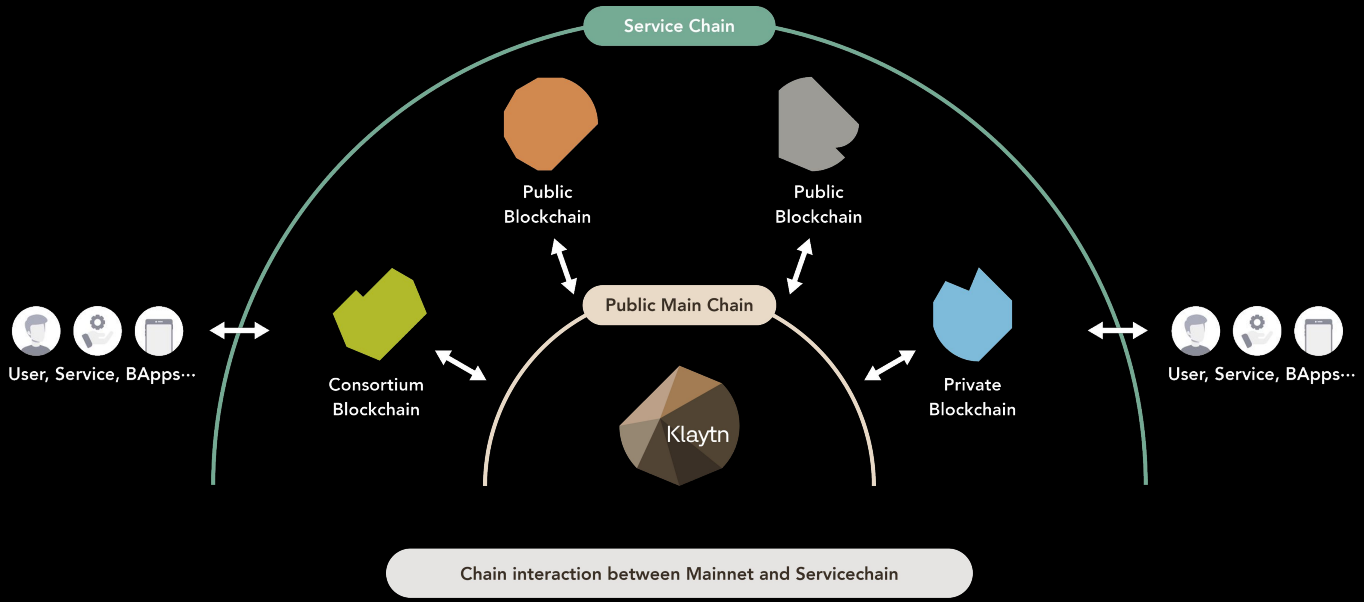
· Klaytn Cypress와 **interaction**을 할 수 있는 독립된 블록체인



Klaytn Service Chain

· 특정 서비스 전용으로 다양한 구성 (Public, Private, Consortium)이 가능한 블록체인

· Klaytn Cypress와 **interaction**을 할 수 있는 독립된 블록체인



Why Service Chain?

·**테스트 용도** : Main-net/Test-net과는 별도로 자체 테스트 시

·**Token 전용 DB** : token 정보 관리에 최적화된 database로 사용하고 싶을 때

·**비용** : Public Block chain의 사용 비용(Gas)이 부담될 때

·**신뢰** : 자체 블록체인이지만, Main-net과 연결하여 User의 신뢰를 얻고 싶을 때

·**보안** : Chain data를 공개하고 싶지 않을 때 / Network 접근권한을 한정하고 싶을 때

·**성능** : 높은 TPS 가 요구되는 특정 서비스를 위해서

·**커스터 마이징**

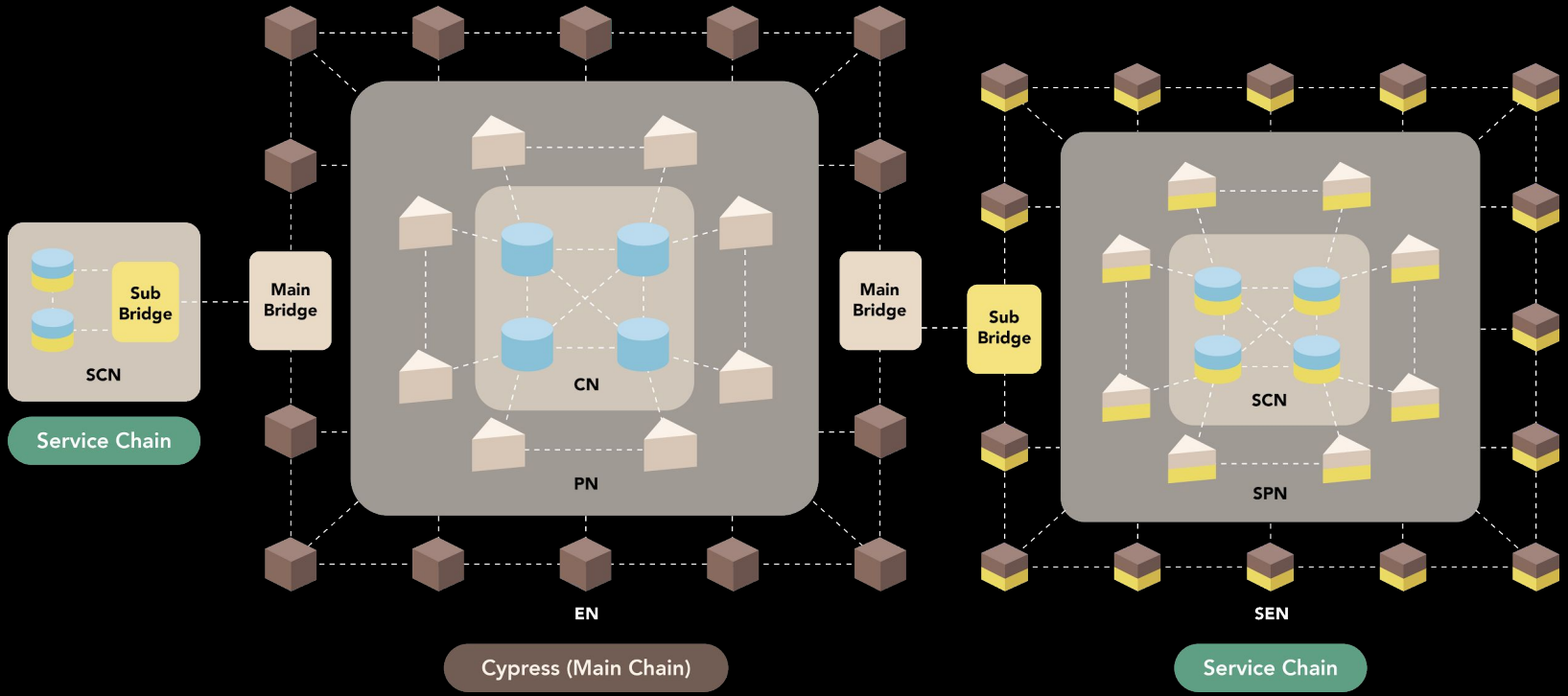
- Network topology (SCN/SPN/SEN, BN)
- Block Gas Limit / Chain ID / Gas Price
- Consensus/Reward => Consortium

Network Architecture

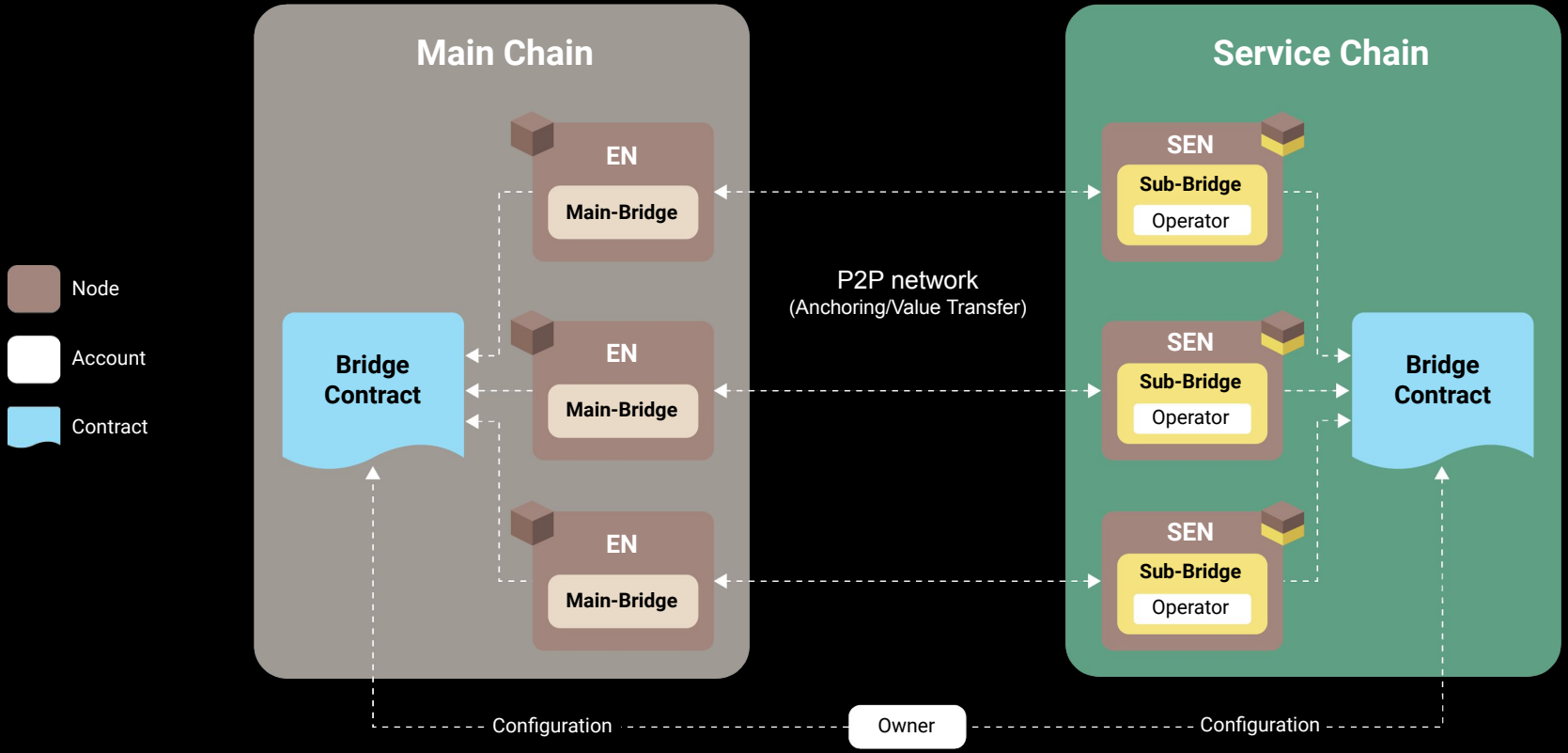
CN : Consensus Node
SCN: Service Chain Consensus Node

PN: Proxy Node
SPN: Service Chain Proxy Node

EN: Endpoint Node
SEN: Service Chain Endpoint Node



Detail Architecture



Service Chain 주요 기능

·**Anchoring**

- Integrity
- Statistics

·**Value Transfer**

- KLAY
- KCT 1-Step Transfer
- KCT 2-Step Transfer
- High availability (HA)
- Multisignature

Feature 1: Anchoring

·Service Chain의 신뢰 확보를 위함.

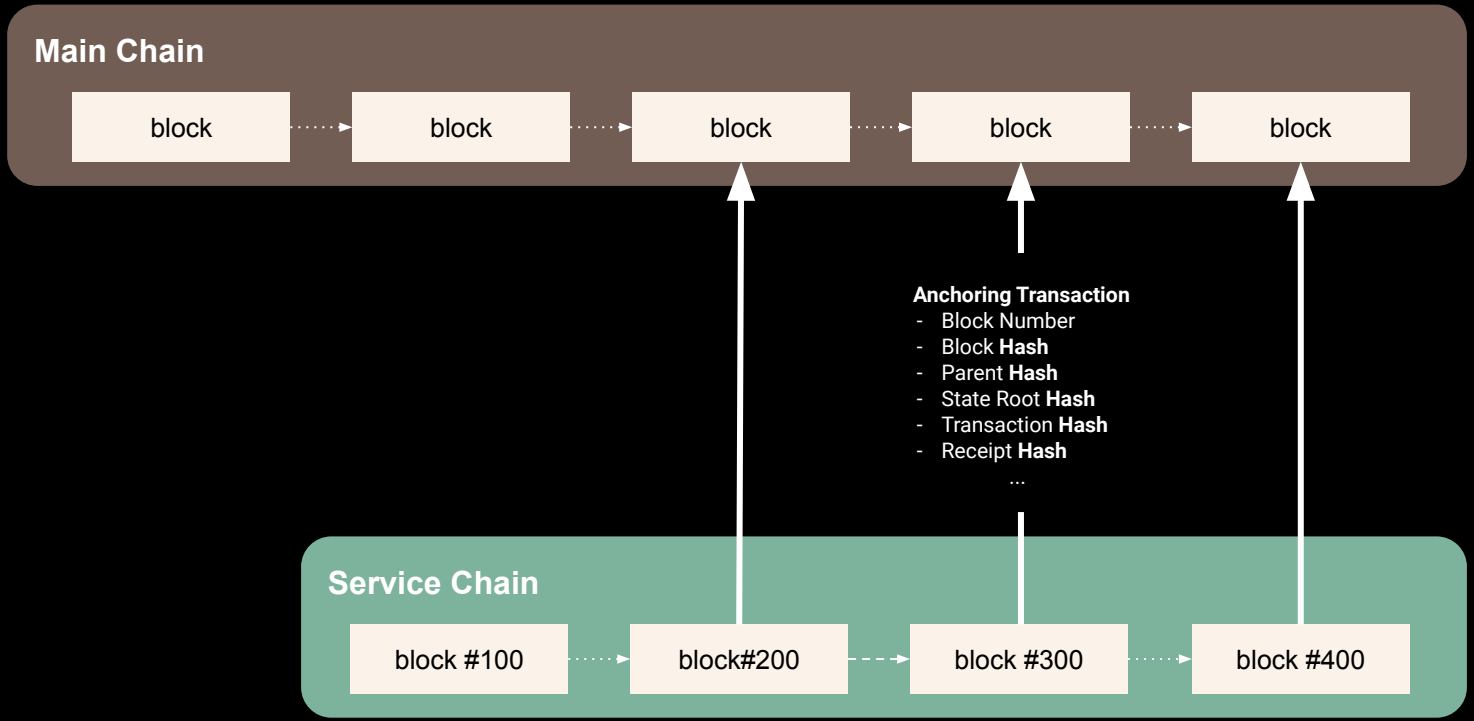
·자체 Private/Public/Hybrid chain을 운영하고 싶지만, 사용자들에게 신뢰를 얻고 싶다.

·주기적인 Service Chain Block의 주요 hash들을 Main Chain에 기록

→ Anchoring 된 데이터까지는 향후 위변조가 불가능

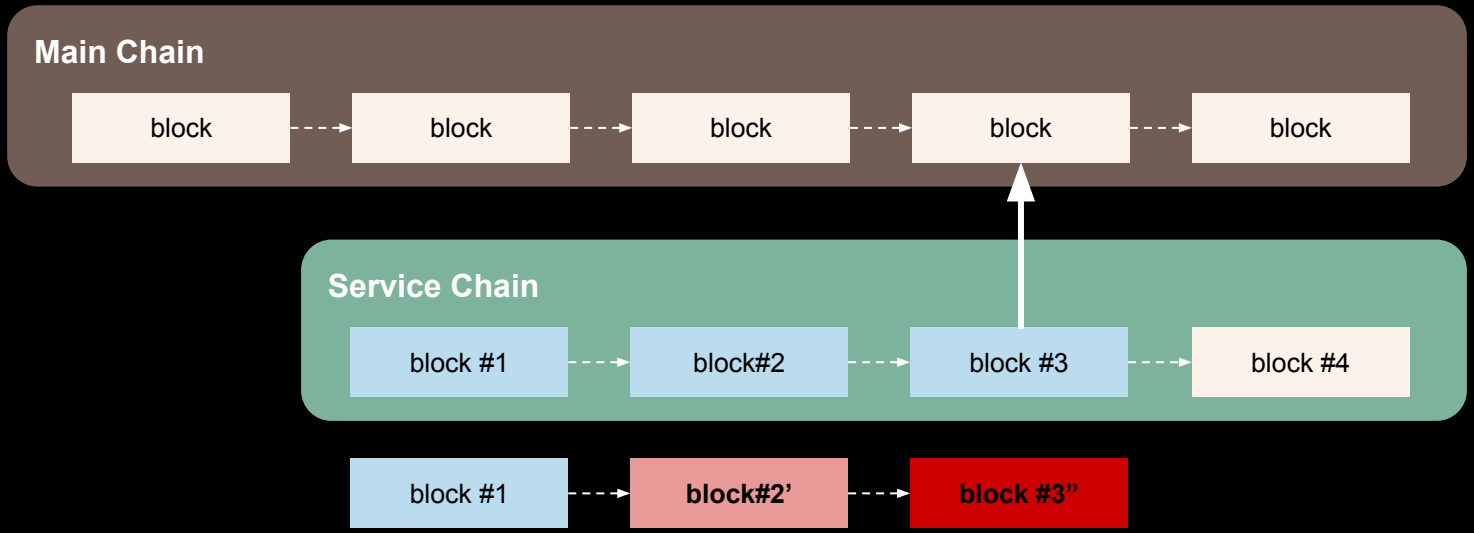
- SP가 User에서 주는 의미 “현재 block의 hash값을 이미 제출 했기 때문에 이후에 내용을 변경하지 않겠다.”

Feature 1: Anchoring - Integrity



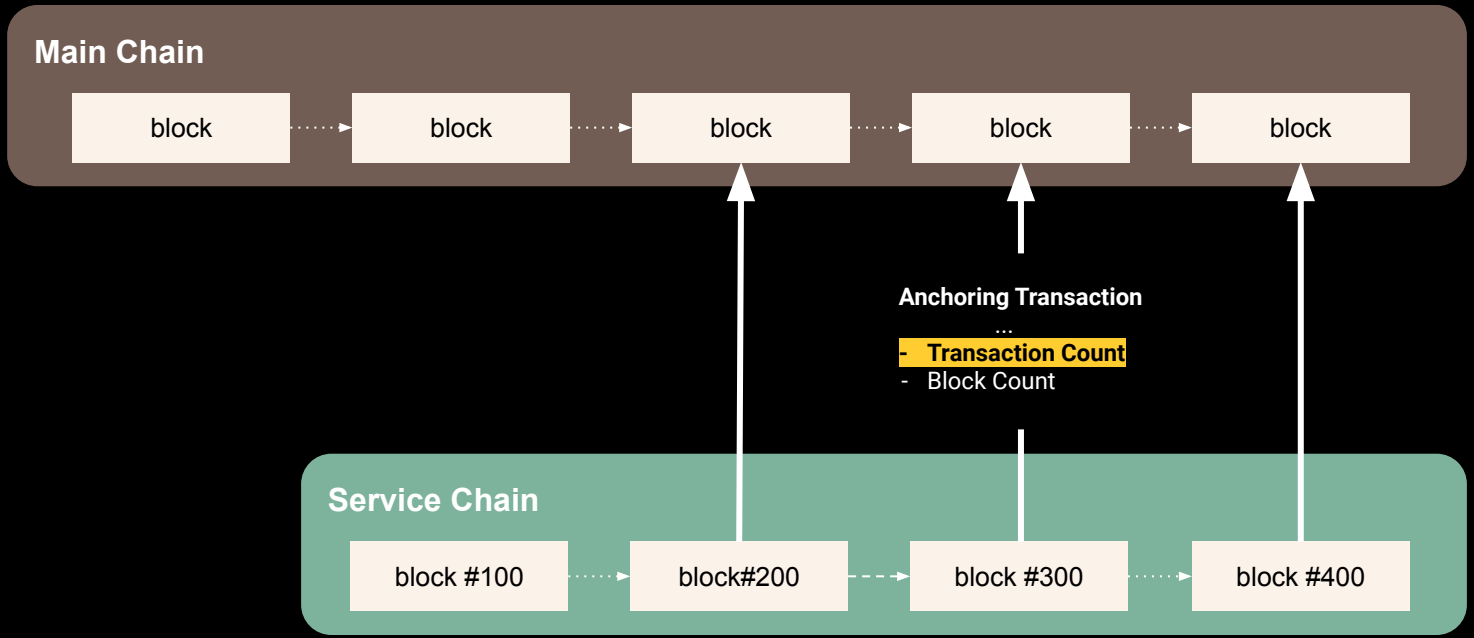
Feature 1: Anchoring - Integrity

Service Chain의 block#3까지 Anchoring 되었고, Block #2를 의도적으로 변조하였다면 Main Chain의 Anchoring 정보와 달라지므로 변조를 알 수 있음. (Hash 계산을 위한 데이터는 제공이 필요함.)



Feature 1: Anchoring - Statistics

Service Chain의 Transaction 개수에 대한 통계자료를 Main Chain에 기록 가능



Service Chain 주요 기능

·**Anchoring**

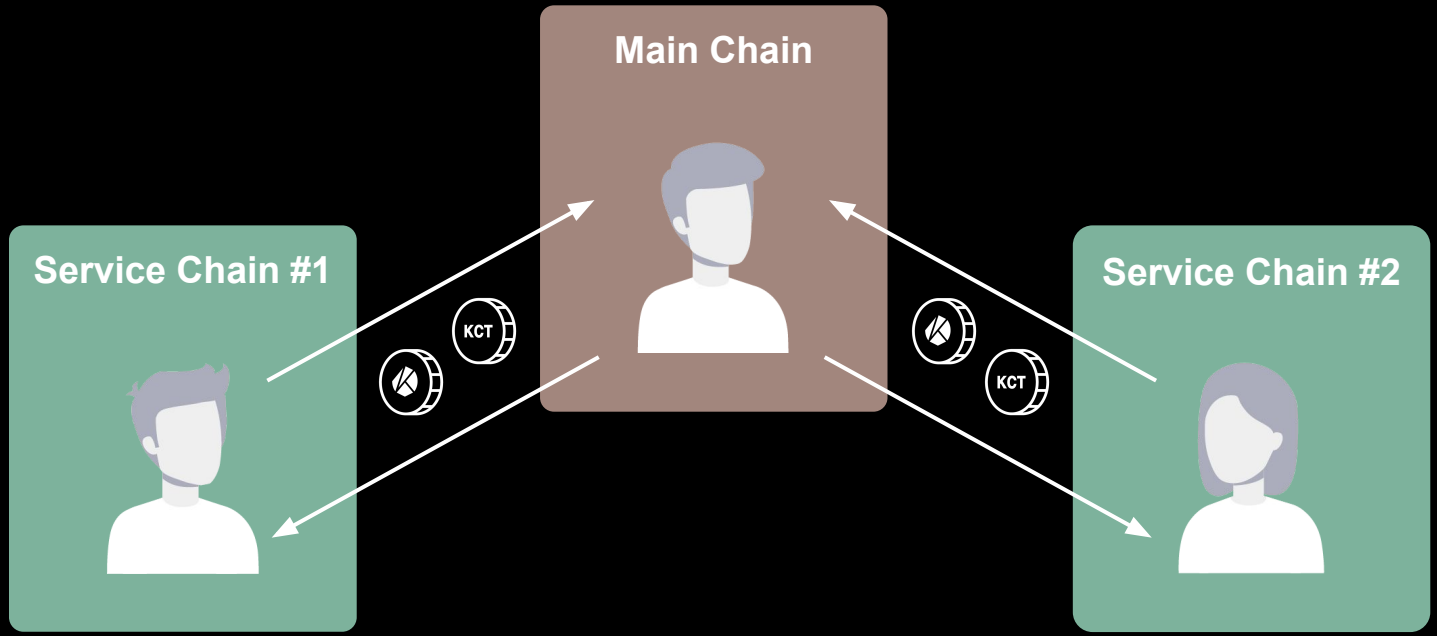
- Integrity
- Statistics

·**Value Transfer**

- KLAY
- KCT 1-Step Transfer
- KCT 2-Step Transfer
- High availability (HA)
- Multisignature

Feature 2: Value Transfer

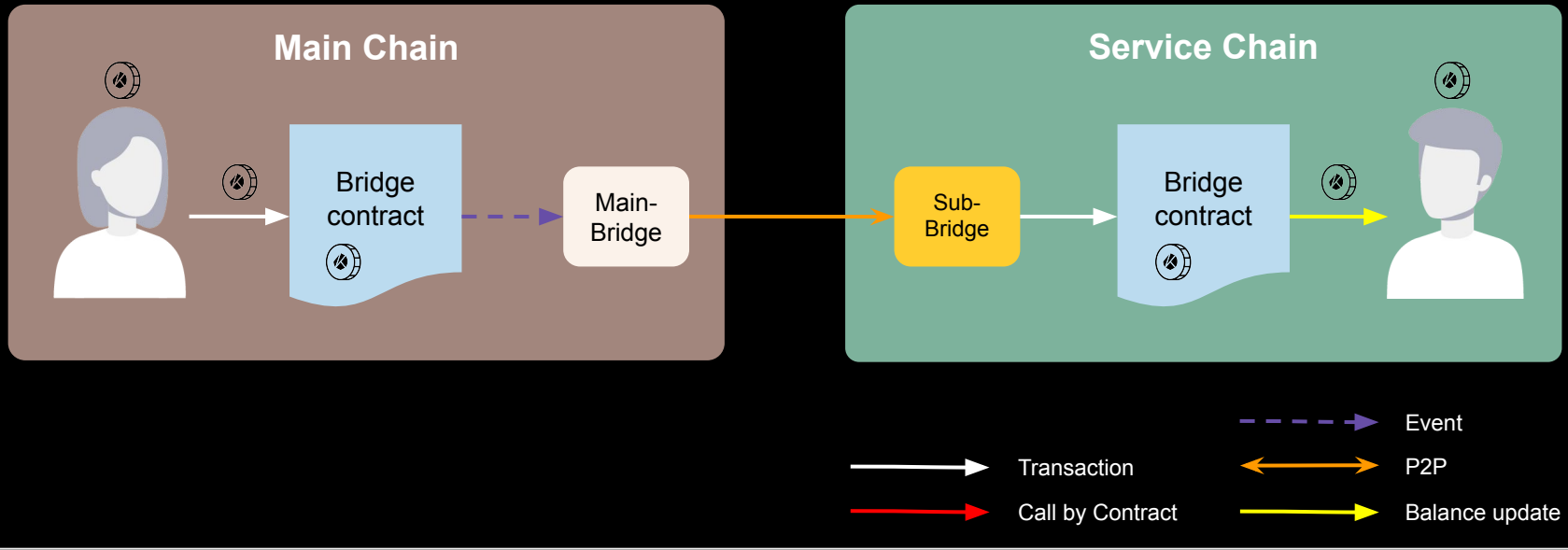
Service Chain 과 Main Chain 간의 KLAY/KCT (Klaytn Compatible Token) 전송 기능



Feature 2: Value Transfer - KLAY

·Main Chain의 User의 KLAY를 Service Chain의 유저에게 전달하는 과정

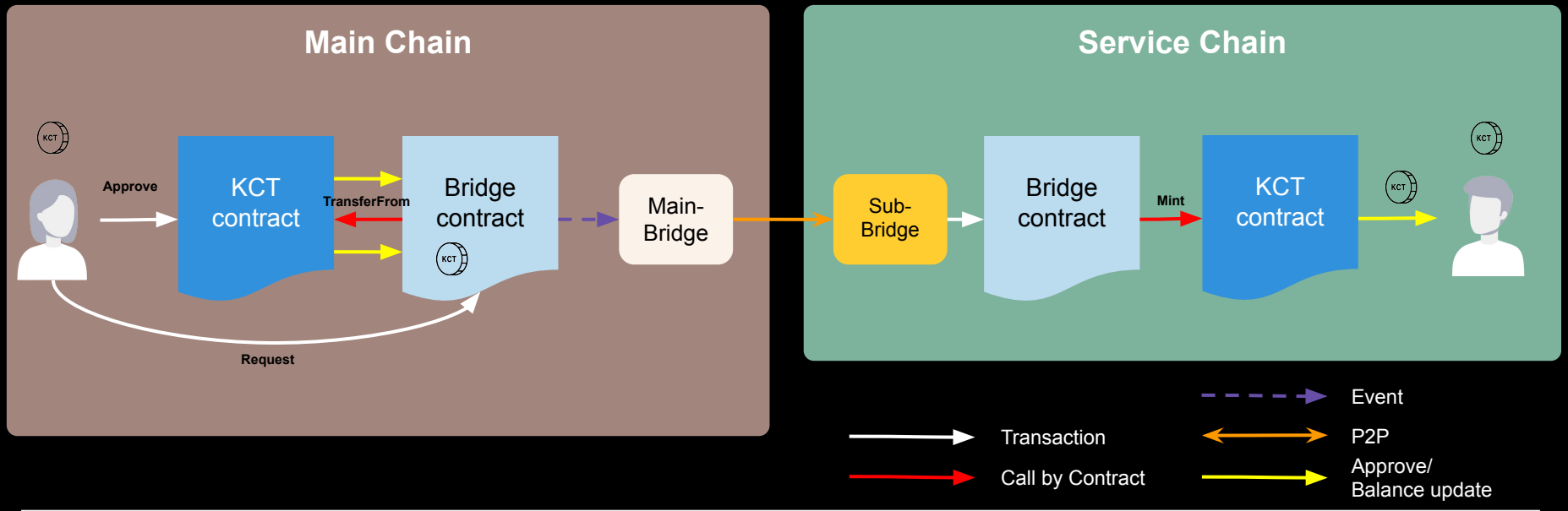
- Main Chain상 KLAY는 bridge contract에 보관.
- Service Chain상 KLAY는 Bridge contract → User에게 전달.



Feature 2: Value Transfer - KCT 2-Step

·Main Chain상 유저의 KLAY를 Service Chain의 유저에게 전달하는 과정

- User는 **2회**의 transaction 발생이 필요함. 1) Approve, 2) Request => Cypress상 KCT는 bridge contract 소유
- Service Chain에서는 Bridge contract가 Sub-Bridge의 처리로 요청된 양만큼의 KCT를 mint하여 User에게 전달됨



Feature 2: Value Transfer - KCT 1-Step w/ extended KCT

·**1회**의 transaction 발생으로 전송을 하기 위함.

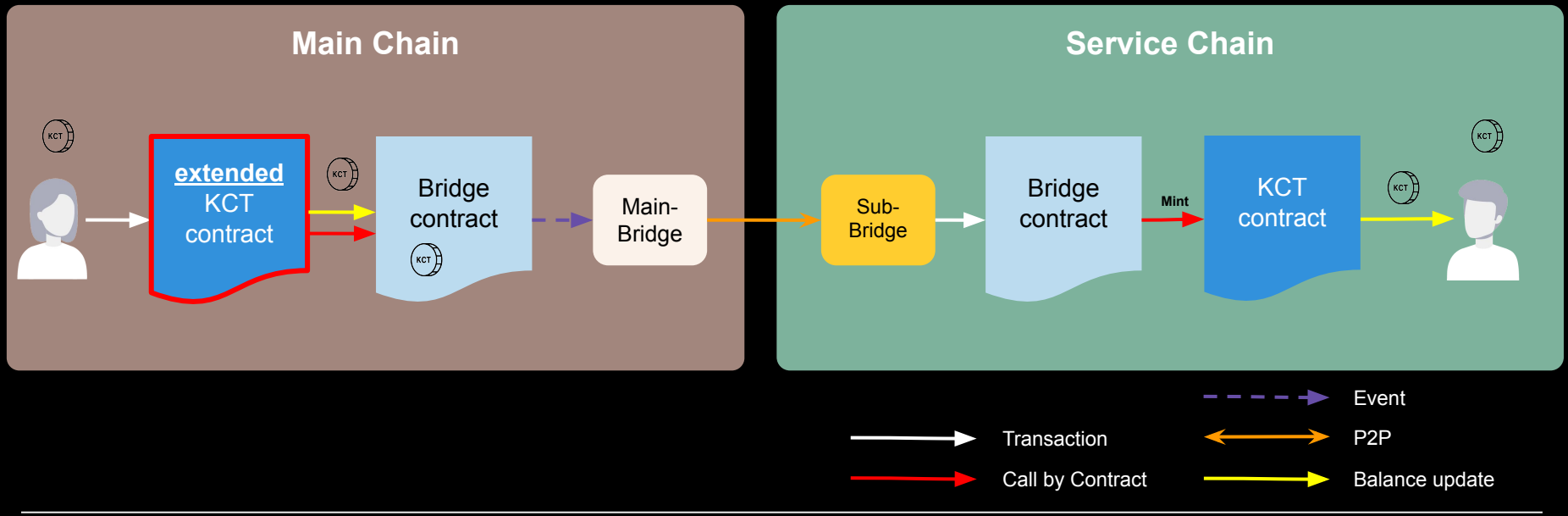
- 기존 KCT Contract에 추가 method가 필요함.
- Transfer를 하고 Bridge contract로 call을 해주면서 Approve/Request (TransferFrom) 기능을 한번에 수행함.

```
function requestValueTransfer(uint256 _amount, address _to, uint256 _feeLimit, bytes calldata _extraData) external {
    transfer(bridge, _amount.add(_feeLimit));
    IERC20BridgeReceiver(bridge).onERC20Received(msg.sender, _to, _amount, _feeLimit, _extraData);
}
```

Feature 2: Value Transfer - KCT 1-Step w/ extended KCT

·Cypress상 User의 KCT를 Service Chain (Child Chain)의 유저에게 전달하는 과정

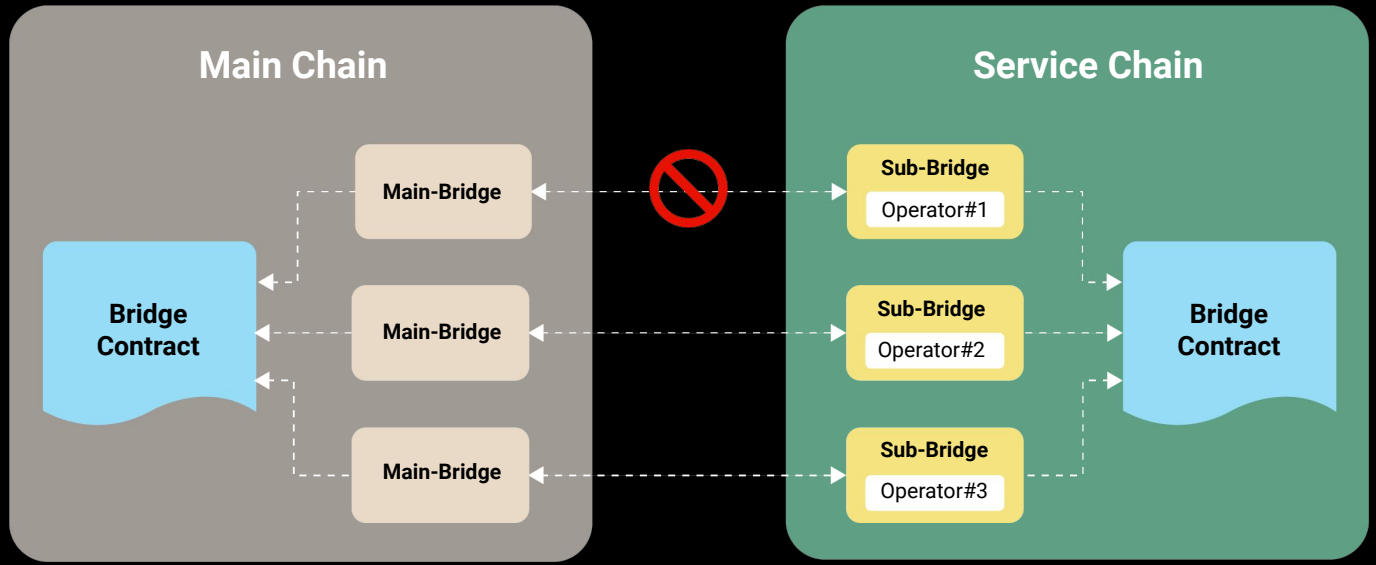
- User는 **1회**의 transaction 발생이 필요함. User의 요청으로 Cypress상 KCT는 bridge contract 소유
- Service Chain에서는 Bridge contract가 Sub-Bridge의 처리로 요청된 양만큼의 KCT를 mint하여 User에게 전달됨



Feature 2: Value Transfer - HA (고 가용성, High Availability)

·하나의 bridge (main, sub)로 서비스를 할 경우 장애가 발생시 연속적인 서비스 불가

→ 다수의 bridge 운영을 통하여 장애 발생시에도 연속적인 서비스가 가능할 수 있음.

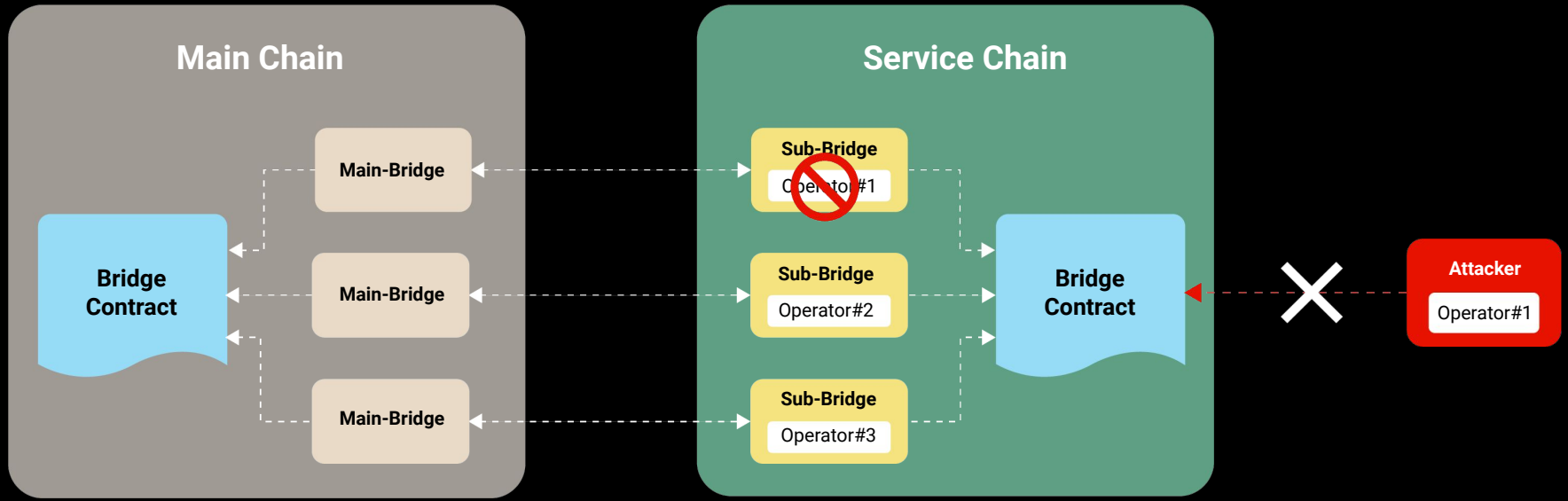


Feature 2: Value Transfer - Multisignature

·operator key 탈취시 Bridge Contract의 자산이 위험해짐.

→ multisig로 다수의 operator의 승인을 요구 할 수 있음. (2-of-3)

→ SP의 안정적인 서비스가 가능함.



Service Chain
Next Step / Future

- **Next Step**
  - Support KCT as native token in Service Chain
- **Future**
  - Platform Bridge contract, One Bridge

Next Step - KCT (Main Chain) ↔ Native Token (Service Chain)

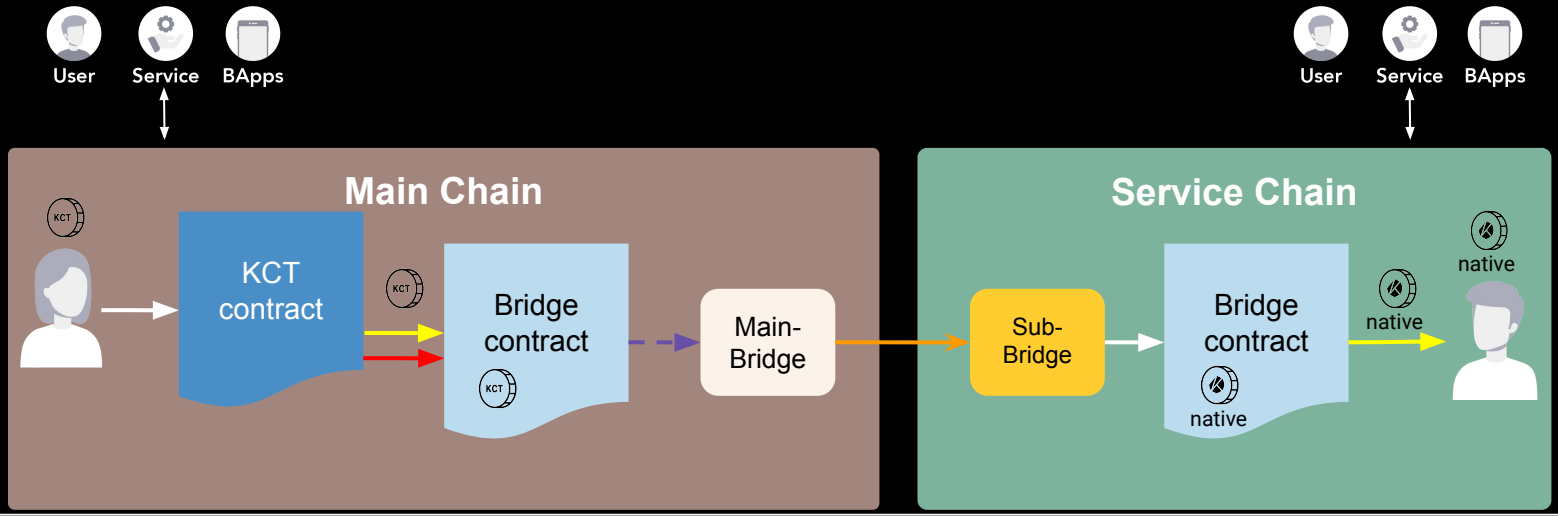
·현재 : Main Chain상의 KCT를 이용한 서비스 운용이 힘든 부분

- Transaction 수수료, Contract 개발의 상대적 어려움 (Native vs KCT), Contract Storage 의 비효율성 (느린 성능)

·향후: Main Chain상의 KCT를 자체 Service Chain에서는 KCT가 아닌 Native Token으로 사용

→ Main Chain에서 KLAY를 사용하듯이 서비스체인내에서는 bApp의 contract 개발이 용이해짐

→ 서비스체인에서 Database를 더 효율적으로 사용 하여 성능 증대



Future

·Platform 지원 Token/Bridge contract / One Bridge Contract

▸ **현재**

→ 서비스체인 마다 각각 Bridge contract를 운용함.

→ Bridge / KCT contract는 KLVM (Solidity)을 사용하고 있음.

→ KLVM의 구조상 성능적인 한계가 있음.

▸ **향후**

→ Platform단에서 Token을 지원해주는 방식 w/o KLVM

→ Platform단에서 Bridge contract를 제공 w/o KLVM

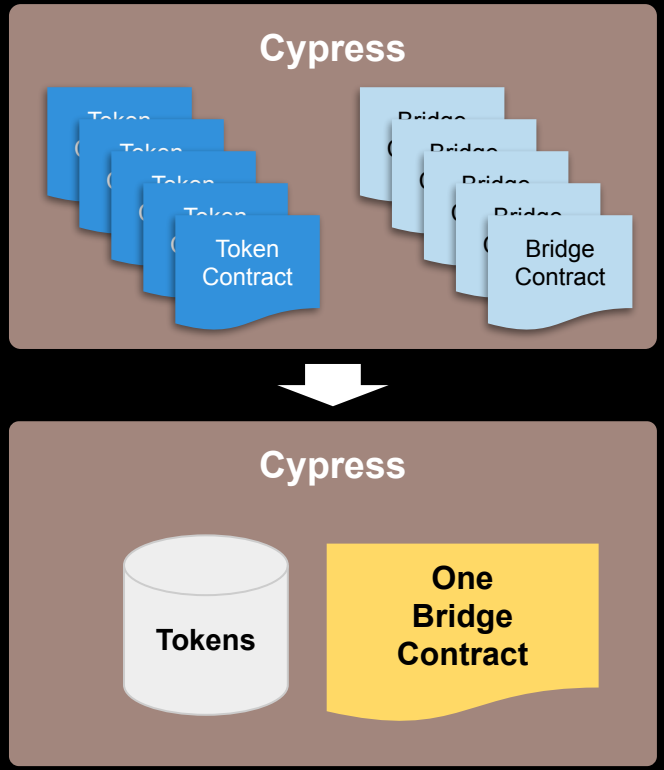
→ 하나의 Bridge contract 운용으로 모든 서비스체인의 정보를 통합

▸ **어려운 점**

→ 새로운 Token들을 KLVM (Solidity)에서는 어떻게 지원할 것인가?

= precompiled contract로 지원 가능

(특정 주소에 특정 기능을 native 언어로 구현하여 동작 시키는 방식, w/o KLVM)



THANK YOU

Ground X
27F, 521, Teheran-ro,
Gangnam-gu, Seoul, Republic of Korea