

# UX Improvement

For Better Blockchain Applications

# 목표


다음 사항들에 대해 알아본다:

- 블록체인의 문제와 그로 인한 제약사항
- User Experience (UX)
- 가스비 대납을 사용한 UX 향상
- 키 관리체계를 사용한 UX 향상

# 블록체인의 문제

	문제점
성능	<ul style="list-style-type: none"><li>• 한 블록이 처리할 수 있는 트랜잭션 개수의 제한</li><li>• 주기적인 블록 합의로 인해 발생하는 지연</li></ul>
비용	<ul style="list-style-type: none"><li>• 합의에 참여하기 위해 필요한 비용</li><li>• 블록을 유지하기 위해 필요한 비용</li></ul>
불편함	<ul style="list-style-type: none"><li>• 사용자가 가스비를 내야하는 불편함</li><li>• 사용자가 직접 키를 관리해야하는 불편함</li></ul>

# Klaytn의 해결법

	문제점	 해결법
성능	<ul style="list-style-type: none"><li>한 블록이 처리할 수 있는 트랜잭션 개수의 제한</li><li>주기적인 블록 합의로 인해 발생하는 지연</li></ul>	<ul style="list-style-type: none"><li>4000 TPS</li><li>1초 블록생성 주기</li></ul>
비용	<ul style="list-style-type: none"><li>합의에 참여하기 위해 필요한 비용</li><li>블록을 유지하기 위해 필요한 비용</li></ul>	<ul style="list-style-type: none"><li>대기업 참여</li><li>KAS (WIP)</li></ul>
불편함	<ul style="list-style-type: none"><li>사용자가 가스를 내야하는 불편함</li><li>사용자가 직접 키를 관리해야하는 불편함</li></ul>	<ul style="list-style-type: none"><li>가스비 대납 기능 구현</li><li>키 관리 체계 구현 (WIP)</li></ul>

# 사용자 경험 (User Experience, UX)

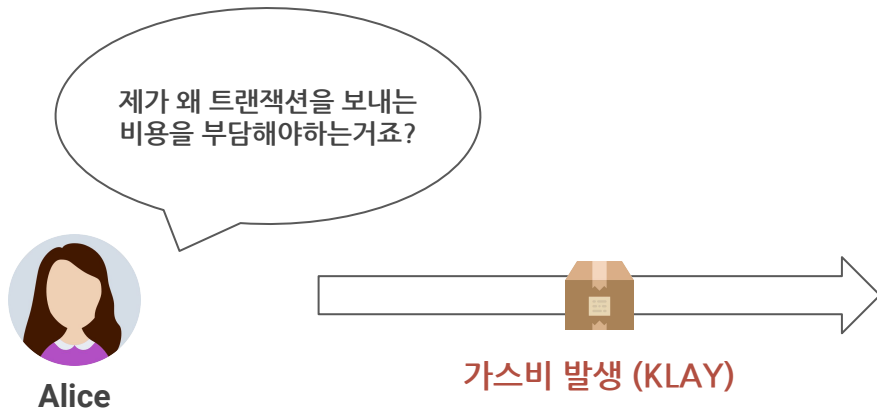
사용자 경험(使用者經驗, User Experience 유저 익스피리언스, 간단히 UX)은 사용자가 어떤 시스템, 제품, 서비스를 직, 간접적으로 이용하면서 느끼고 생각하게 되는 총체적 경험을 말한다. 단순히 기능이나 절차상의 만족뿐 아니라 전반적인 지각 가능한 모든 면에서 사용자가 참여, 사용, 관찰하고 상호 교감을 통해서 알 수 있는 가치있는 경험이다. 긍정적인 사용자 경험의 창출은 산업 디자인, 소프트웨어 공학, 마케팅, 및 경영학의 중요 과제이며 이는 사용자의 니즈의 만족, 브랜드의 충성도 향상, 시장에서의 성공을 가져다 줄 수 있는 주요 사항이다. 부정적인 사용자 경험은 사용자가 원하는 목적을 이루지 못할 때나 목적을 이루더라도 감정적, 이성적으로나 경제적으로 편리하지 못하거나 부정적인 반응을 불러일으키는 경험을 하게 되는 경우 발생할 수 있다.

**가스비가 UX에 미치는 영향**

# 가스비

- TX를 처리하는데 필요한 자원을 비용으로 전환한 것이 가스(gas)
  - 플랫폼 사용료
  - EVM 명령어마다 정해진 가스량이 존재
  - Sender는 (TX의 처리를 위해 필요한 가스의 총량) x (Gas Price)만큼의 KLAY를 제공
- 복잡한 연산을 수행할 수록 높은 가스비가 소모
  - Klaytn의 Gas Price는 25 ston/gas (0.000000025 KLAY)로 고정
  - KLAY 전송에 필요한 가스비는 0.000525 KLAY (21,000 gas)
  - Count 컨트랙트를 배포하는데 약 183,000 gas가 필요

# 문제점 1



불필요한 비용으로 인식되어 사용자  
경험을 크게 저해하는 요소로 작용



## 문제점 2



사용하는 코인/토큰이 KLAY가 아닐 경우  
사용자에게 불필요한 추가 비용으로 인식

# 문제점 정리

1. 사용자는 플랫폼 사용료를 지불해야하는 것을 이해 못 함
  - 사용하는 서비스가 KCT (Klaytn Compatible Token)를 사용할 경우 더욱 그러함
  - 작은 금액이라도 민감하게 반응할 수 있음
2. 사용자가 KLAY를 취득하기 어려움
  - 법정화폐(fiat)로 가상자산을 취득하는 것이 어려움
  - 어플리케이션 최초 사용 시나리오에서 큰 장애물로 작용
  - 사용자에게 지원금을 지급할 수 있으나 남용(abusing)의 위험 있음

사용자가 가스비를 내지 않는 것이 사용자경험 개선에 큰 도움이 될 것

## 아이디어

서비스/어플리케이션 운영사가 사용자를  
대신하여 가스비를 **대납**할 수 있다면 어떨까?

# 대납의 필요성



Alice

Sender



Service

Payer

- ‘트랜잭션을 보내는 사람 (Sender)’과 ‘가스비를 부담하는 사람 (Payer)’을 구분
  - Sender를 일반 유저, Payer를 서비스/어플리케이션 운영사라고 정의
- Sender의 서비스 사용 → Payer의 매출 확대
  - Klaytn의 가스비는 매우 낮으나 KLAY 취득이 어려움
  - Payer는 Sender가 서비스를 seamless하게 사용할 수 있도록 가스비를 대신 내줄 유인이 충분

# Klaytn Fee Delegation



- 대납 트랜잭션은 두개의 서명을 가짐

- 트랜잭션은 Sender가 생성하고 서명한 뒤 Payer에게 전달 → Signature 1
- Payer는 Sender가 전달한 트랜잭션에 추가정보를 기입하고 서명 → Signature 2
- 각각의 서명이 sender 주소와 payer 주소에 부합할 경우 트랜잭션이 옳다라고 정의
- 가스비는 Payer의 밸런스에서 차감 (nonce 변경 없음)

# Fee Delegation in caver-js

// Sender-side

```
const { rawTransaction: senderRawTransaction } = await caver.klay.accounts.signTransaction({  
  type: 'FEE_DELEGATED_VALUE_TRANSFER',  
  from: sender.address,  
  to: '0x34ca11930cd5e0971d8bb9860d9b977d3bb9187b',  
  gas: '3000000',  
  value: 1,  
}, sender.privateKey);
```

// Payer-side

```
const { rawTransaction: finalTx } = await caver.klay.accounts.signTransaction({  
  senderRawTransaction: senderRawTransaction,  
  feePayer: payer.address  
}, payer.privateKey);
```

# Fee Delegation Analysis

## Advantages

- UX 향상
- 불필요한 토큰 거래 최소화
- 사용자층 확대 (개발자, 토큰 홀더 → 일반유저)
- 트랜잭션 규모 확대

## Disadvantages

- 플랫폼 성능 저하 (여러 서명 검증)
- 서비스 개발 복잡도 증가
- 서비스 관리비용 증가 (TOSS 기준 월 264 USD)

**비밀키가 UX에 미치는 영향**



# 키관리의 어려움

잃어버릴 경우 복구 불가능

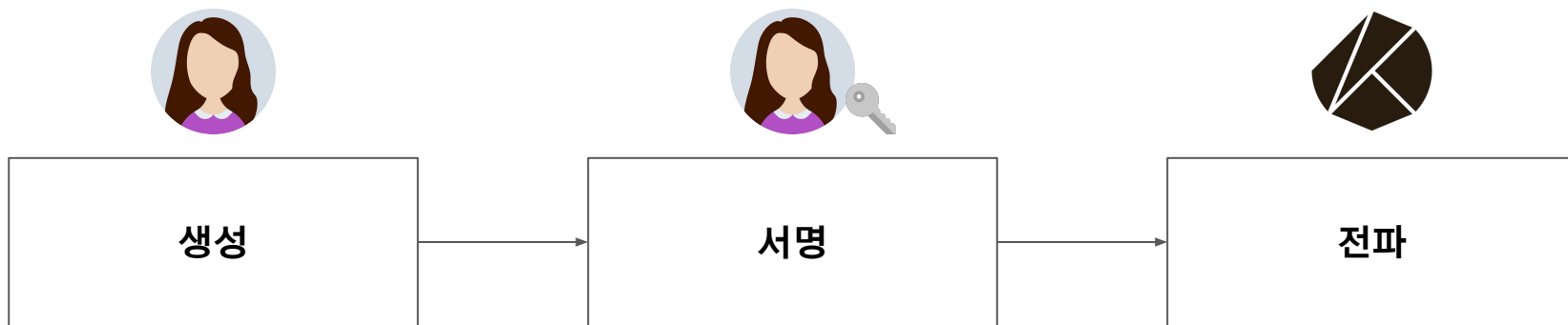
분실, 해킹의 위험

낮은 가독성 + 외우기 힘든 문자조합



개인이 관리하기 어려움

# BApp에 트랜잭션을 보낸다는 것은



# 키만 다른 사람에게 위임할 수 있을까?



생성

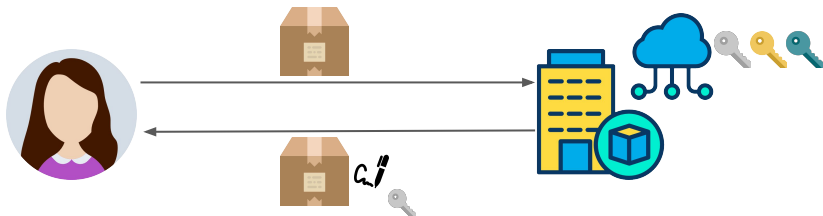


서명



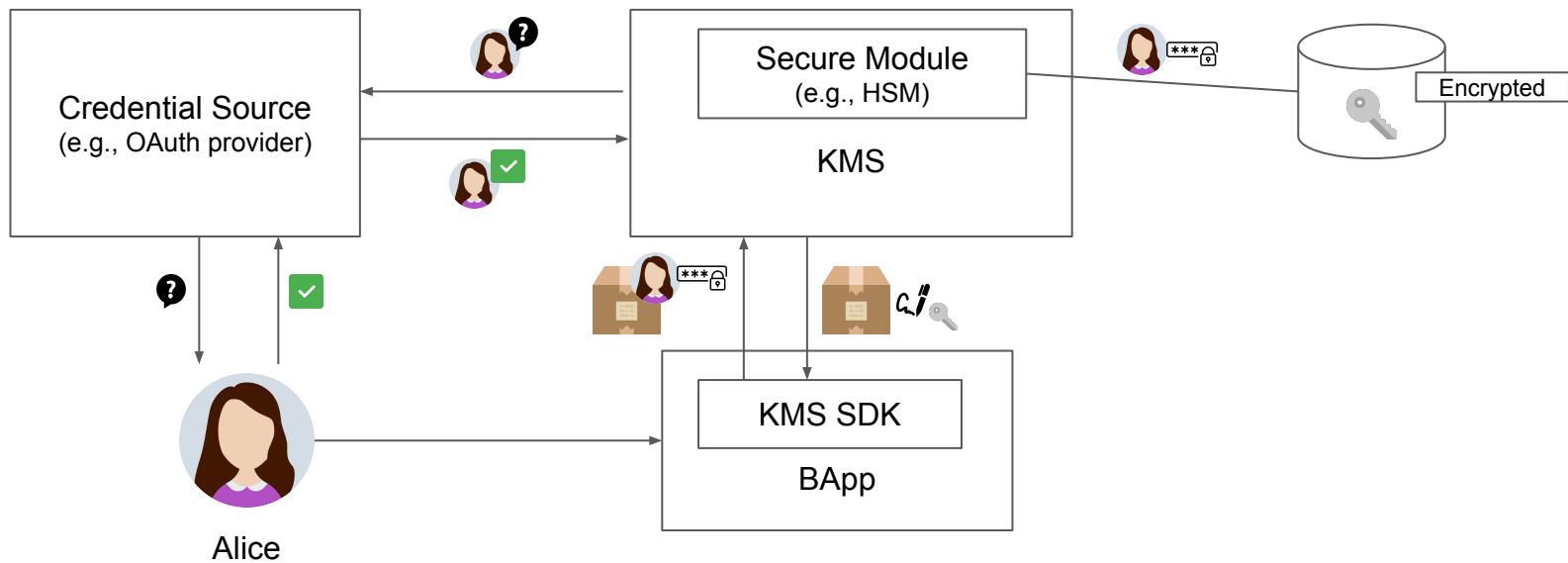
전파

# Key Management System (KMS)



- 키를 클라우드 서비스가 관리하는 아이디어
  - KMS는 키를 안전한 곳에 보관; 사용자는 KMS에 트랜잭션을 전송하여 서명을 요청
  - 사용자 인증이 성공할 경우 KMS는 사용자에게 **서명된 트랜잭션**을 전달
- 키를 분실할 위험을 방지
  - 키는 2중으로 암호화되어 보관 (분실, 재난 등에 대비)
  - 복호화된 키는 KMS도 열람할 수 없는 장치를 구현 (e.g., HSM)

# KMS 구현 예시



# KMS Analysis

## Advantages

- UX 향상
- 키 분실, 해킹 위험 최소화
- 사용자 인증을 통해 KYC 수행
- Seamless한 One Key ⇔ Multi BApp 사용

## Disadvantages

- 키 관리 주체가 중앙화 (SPoF)
- 키 관리 비용 발생 (하드웨어, 운영인력 등)
- 키 분실 또는 훼손 시 책임 발생

**End of Document**