

DeFi

ERC-4337



The Lecture : ERC-4337

■ ERC 4337이란?

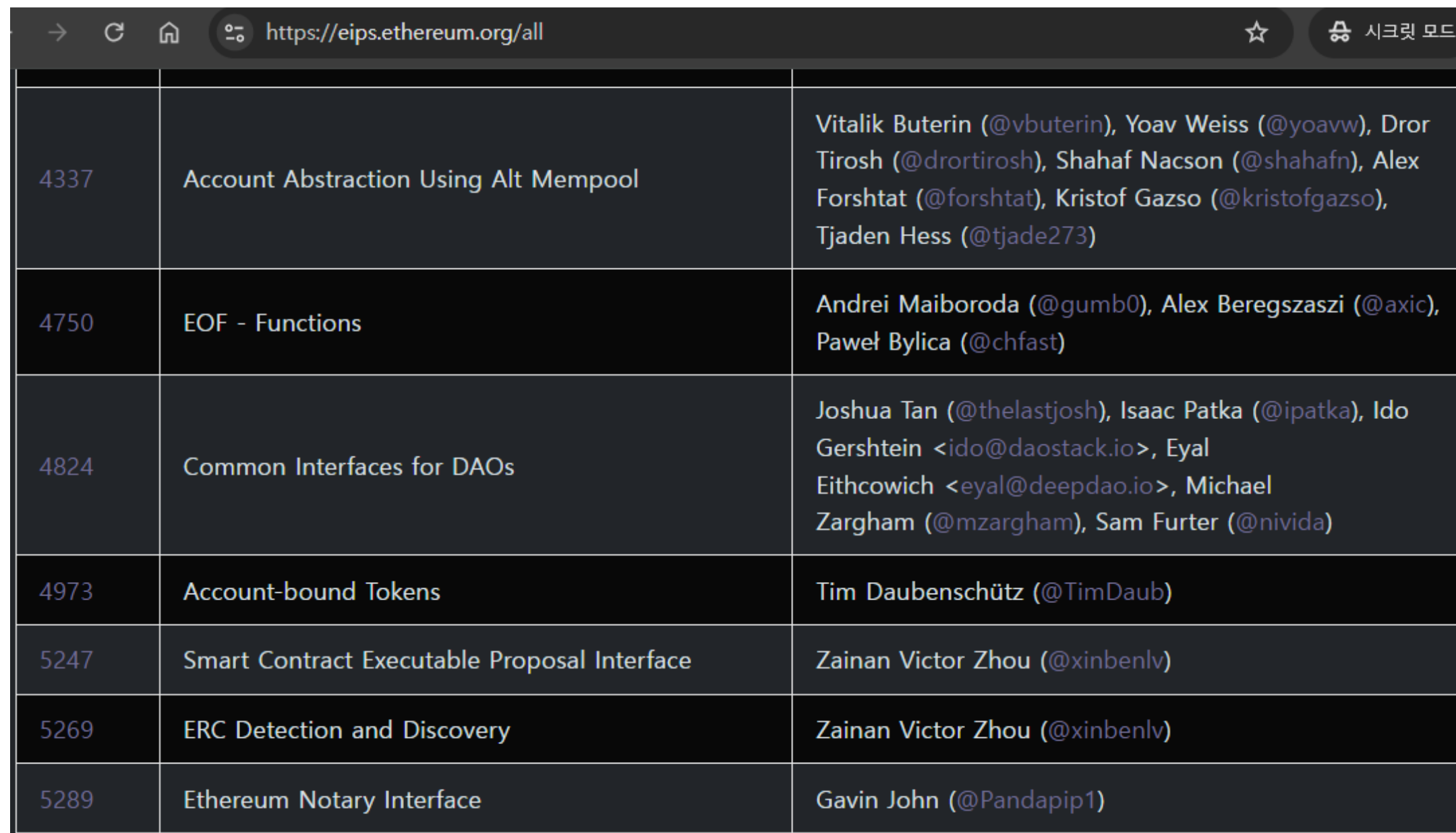
- ERC : Ethererum Request for Comment
 - 이더리움에 새로운 표준을 제안할 때 붙이는 형식
- 대표적인 ERC 종류
 - ERC-20 : "Token"에 대한 표준
 - ERC-721 : "NFT"에 대한 표준
 - ERC-4337 : "계정 추상화(account abstraction)"에 대한 표준 제안



The Lecture : ERC-4337

■ ERC 4337이란?

- ERC : Ethererum Request for Comment
 - 이더리움에 새로운 표준을 제안할 때 붙이는 형식

A screenshot of a web browser displaying the list of Ethereum Improvement Proposals (EIPs) on the etherscan.io website. The browser's address bar shows 'https://eips.ethereum.org/all'. The page contains a table with three columns: EIP number, title, and authors. The table lists several EIPs, including 4337, 4750, 4824, 4973, 5247, 5269, and 5289.

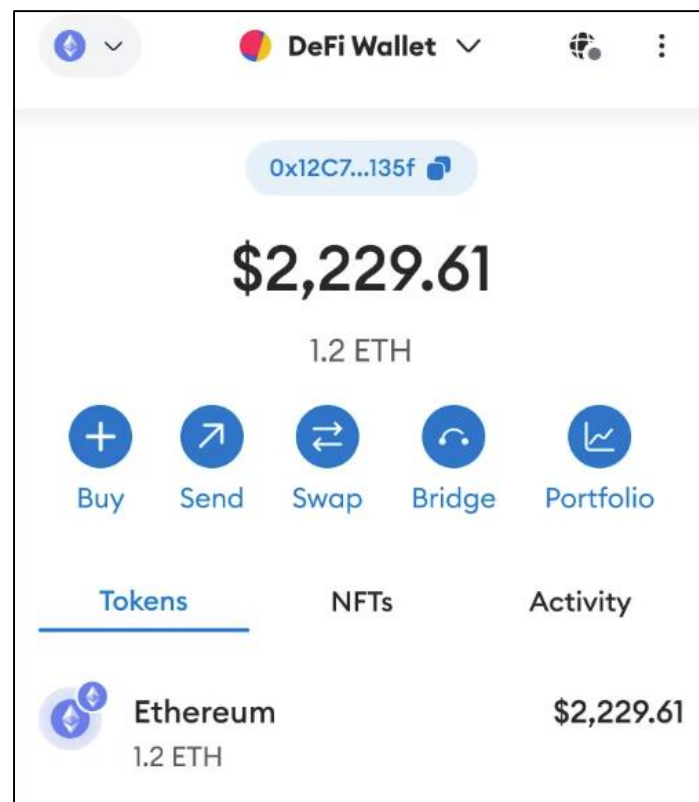
4337	Account Abstraction Using Alt Mempool	Vitalik Buterin (@vbuterin), Yoav Weiss (@yoavw), Dror Tirosh (@drortirosh), Shahaf Nacson (@shahafn), Alex Forshtat (@forshtat), Kristof Gazso (@kristofgazso), Tjaden Hess (@tjade273)
4750	EOF - Functions	Andrei Maiboroda (@gumb0), Alex Beregszaszi (@axic), Paweł Bylica (@chfast)
4824	Common Interfaces for DAOs	Joshua Tan (@thelastjosh), Isaac Patka (@ipatka), Ido Gershtein <ido@daostack.io>, Eyal Eithcowich <eyal@deepdao.io>, Michael Zargham (@mzargham), Sam Furter (@nivida)
4973	Account-bound Tokens	Tim Daubenschütz (@TimDaub)
5247	Smart Contract Executable Proposal Interface	Zainan Victor Zhou (@xinbenlv)
5269	ERC Detection and Discovery	Zainan Victor Zhou (@xinbenlv)
5289	Ethereum Notary Interface	Gavin John (@Pandapip1)

The Lecture : ERC-4337

■ Ethereum Account

■ EOA(Externally Owned Accounts)

- 정의 : 개인이 소유한 지갑 계정(Private Key로 동작)
- 특징
 - Private Key(개인키)만 있으면 지갑 운용 가능
 - 사용자가 직접 트랜잭션 발생 가능



메타마스크(Metamask), 하드웨어 월렛 등

■ CA(Contract Account)

- 정의 : 스마트 컨트랙트 코드가 담긴 계정
- 특징
 - 외부로부터 트랜잭션을 받아 코드를 실행
 - 미리 정의된 코드로만 동작 수행
 - 로직 기반 다양한 기능(Ex. 조건부 송금) 등

A screenshot of a web browser displaying a list of tokens on a platform like Etherscan. The URL is 'https://etherscan.io'. The page shows a table of tokens with columns for the token name, address, price, and balance. The tokens listed are: Ethereum (ETH), USDC (USDC), Wrapped Ether (WETH), Tether USD (USDT), MultiVAC (MTV), YFED.FINANCE (YFED), ERC-20: auroom.finance (AUROOM), and Crypto Interplanetary (CRI).

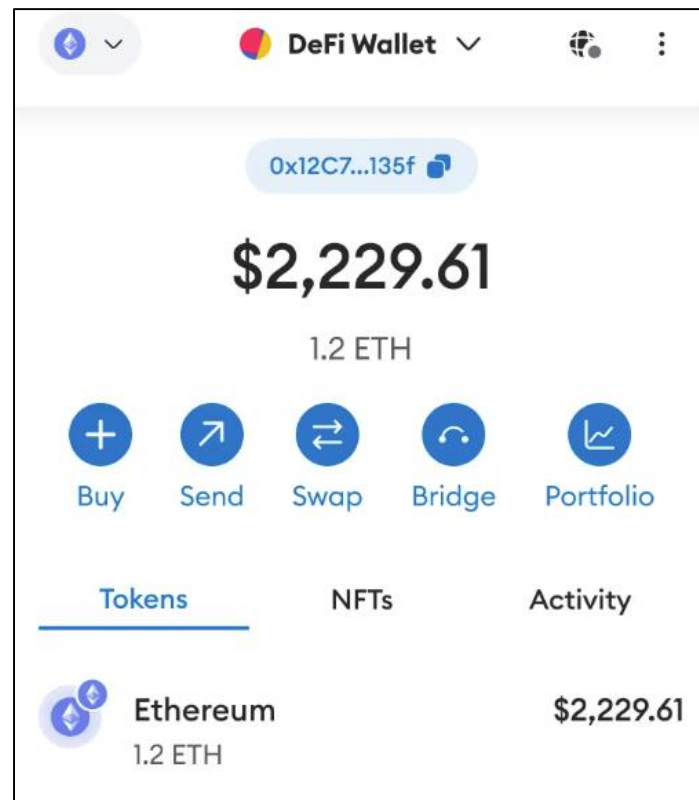
Token	Address	Price	Balance
Ethereum (ETH)	-	\$4,665.65 (+2.65%)	0
USDC (USDC)	0xA0b86991...E3606eB48	\$0.999811	72,242,633
Wrapped Ether (WETH)	0xC02aaA39...83C756Cc2	\$4,665.473451	4,909.2682
Tether USD (USDT)	0xdAC17F95...13D831ec7	\$1.001	3.837499
MultiVAC (MTV)	0x6226e00b...C14fAB77f	\$0.000478	1,000
YFED.FINANCE (YFED)	0x2DBd330b...dDDAcE2A	\$0.063287	0.000001
ERC-20: auroom.finance (AUROOM)	0xe747Ff1...eC2366Fb3	-	50.713719
Crypto Interplanetary (CRI)	0x12E95193...D1D463E4d	-	0.008

Uniswap V3 풀 컨트랙트

■ Ethereum Account

■ EOA(Externally Owned Accounts)

- 정의 : 개인이 소유한 지갑 계정(Private Key로 동작)
- 특징
 - Private Key(개인키)만 있으면 지갑 운용 가능
 - 사용자가 직접 트랜잭션 발생 가능



메타마스크(Metamask), 하드웨어 월렛 등



Private Key(개인키) 의존

- 키 분실 시, 자산 복구 불가능



확장성 부족

- Only 지갑으로 서, 단순 송/수신 기능만 가능



불편



가스비 문제

- 트랜잭션할 때마다 직접 가스비를 지불해야함



UX 불편

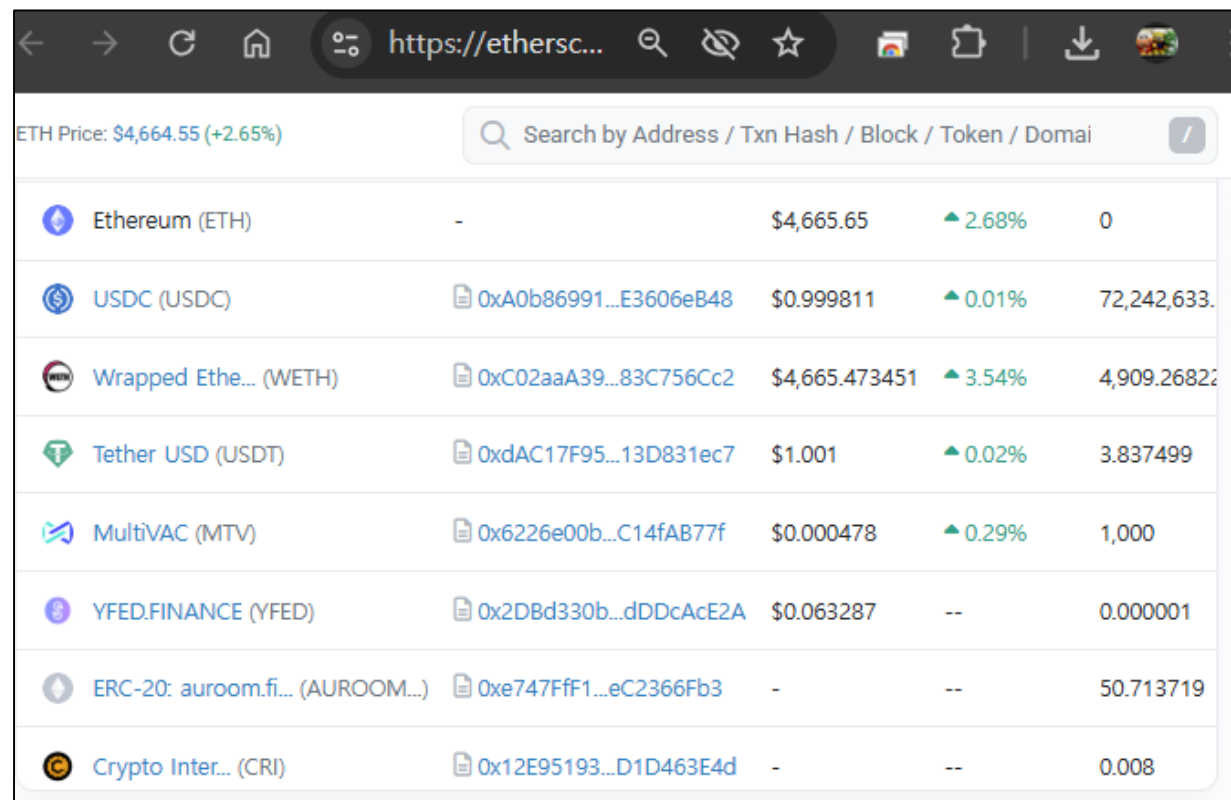
- 신규 유저는 키, 가스비 개념 및 관리가 불편

The Lecture : ERC-4337

■ Ethereum Account

■ CA(Contract Account)

- 정의 : 스마트 컨트랙트 코드가 담긴 계정
- 특징
 - 외부로부터 트랜잭션을 받아 코드를 실행
 - 미리 정의된 코드로만 동작 수행
 - 로직 기반 다양한 기능(Ex. 조건부 송금) 등



The screenshot shows the Etherscan website interface. At the top, it displays the ETH price as \$4,664.55 (+2.65%). Below this is a search bar and a table of tokens. The table has columns for token name, address, price, change, and balance. The tokens listed are Ethereum (ETH), USDC (USDC), Wrapped Ether (WETH), Tether USD (USDT), MultiVAC (MTV), YFED.FINANCE (YFED), ERC-20: auroom.fi (AUROOM...), and Crypto Inter... (CRI).

Token	Address	Price	Change	Balance
Ethereum (ETH)	-	\$4,665.65	▲ 2.68%	0
USDC (USDC)	0xA0b86991...E3606eB48	\$0.999811	▲ 0.01%	72,242,633
Wrapped Ether (WETH)	0xC02aaA39...83C756Cc2	\$4,665.473451	▲ 3.54%	4,909.26822
Tether USD (USDT)	0xdAC17F95...13D831ec7	\$1.001	▲ 0.02%	3.837499
MultiVAC (MTV)	0x6226e00b...C14fAB77f	\$0.000478	▲ 0.29%	1,000
YFED.FINANCE (YFED)	0x2DBd330b...dDDcAcE2A	\$0.063287	--	0.000001
ERC-20: auroom.fi (AUROOM...)	0xe747Ff1...eC2366Fb3	-	--	50.713719
Crypto Inter... (CRI)	0x12E95193...D1D463E4d	-	--	0.008

Uniswap V3 풀 컨트랙트



수동적 계정

- 스스로 트랜잭션 발생 불가 (항상 EOA 호출 필요)



가스비 지불 불가

- ETH 잔액이 있어도 스스로 지불 불가 (EOA 의존)



불편



코드 취약점 위험

- 버그/해킹 발생 시 대규모 피해(Ex. DAO 해킹)

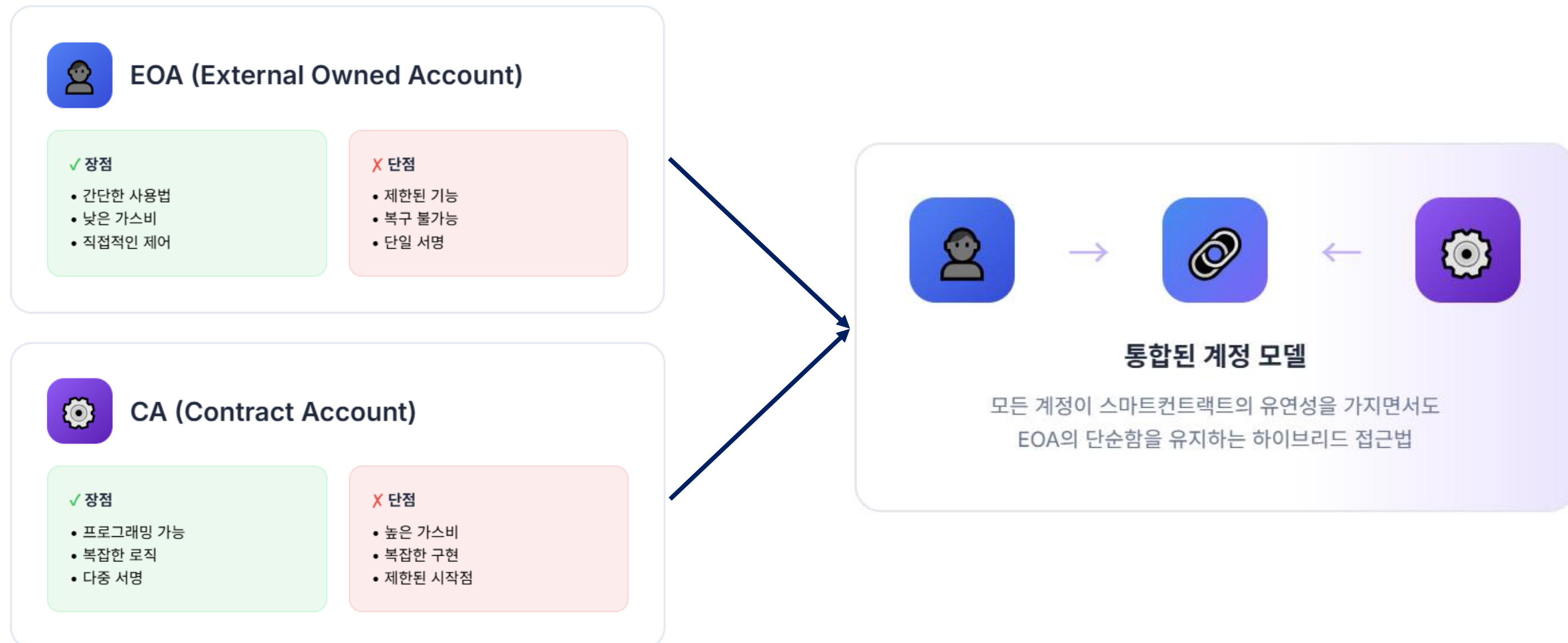


복잡한 개발/배포

- 단순 지갑 용도로 쓰기에는 진입 장벽 높음

Account Abstraction

- EOA와 CA의 장/단점을 활용/개선한 “Account Abstraction”가 논의되기 시작
- 즉, EOA와 CA의 구분을 없애고, 모든 Account가 스마트컨트랙트처럼 동작할 수 있도록 만드는 개념



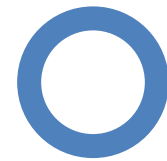
■ Account Abstraction

- 왜 Account가 스마트 컨트랙트처럼 동작되길 원할까?
= Account가 스마트 컨트랙트처럼 동작되면 어떠한 장점이 생길까?

전통 금융 은행의 계좌

기존 EOA

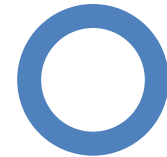
일일 한도 금지 제한
(Ex. 하루 1천만원 이하 이체)



정기 자동 이체
(Ex. 매일 25일 자동 송금)



계정 복구 기능
(Ex. 비밀번호 분실 시 자산 복구)



수수료 대납
(Ex. 특정 사용자에게 수수료 면제)



The Lecture : ERC-4337

■ Account Abstraction

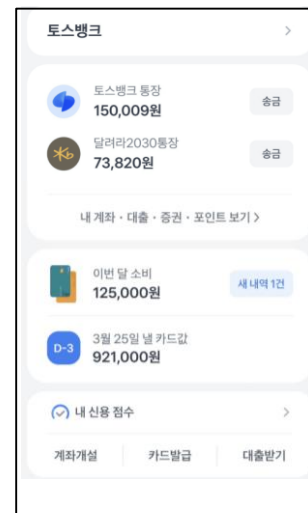
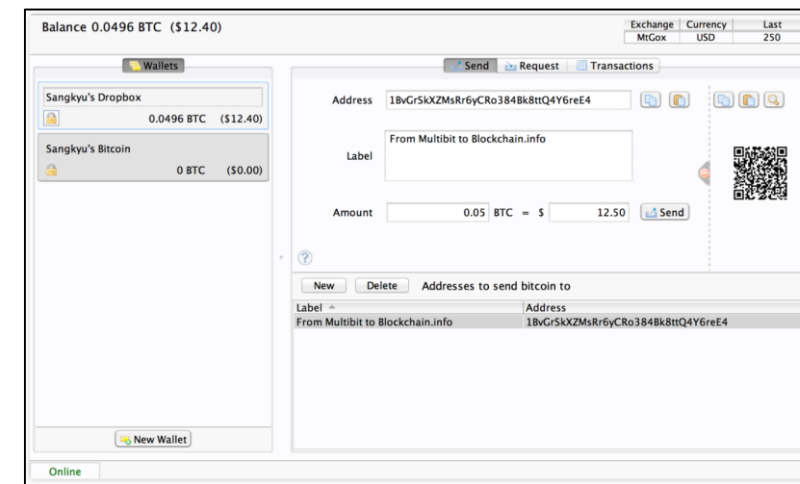
- 왜 Account가 스마트 컨트랙트처럼 동작되길 원할까?



제한된 기능

복잡한 관리

어려운 접근성



다양한 기능

간편한 관리

쉬운 접근성



사용자 경험(User Experience, UX)

The Lecture : ERC-4337

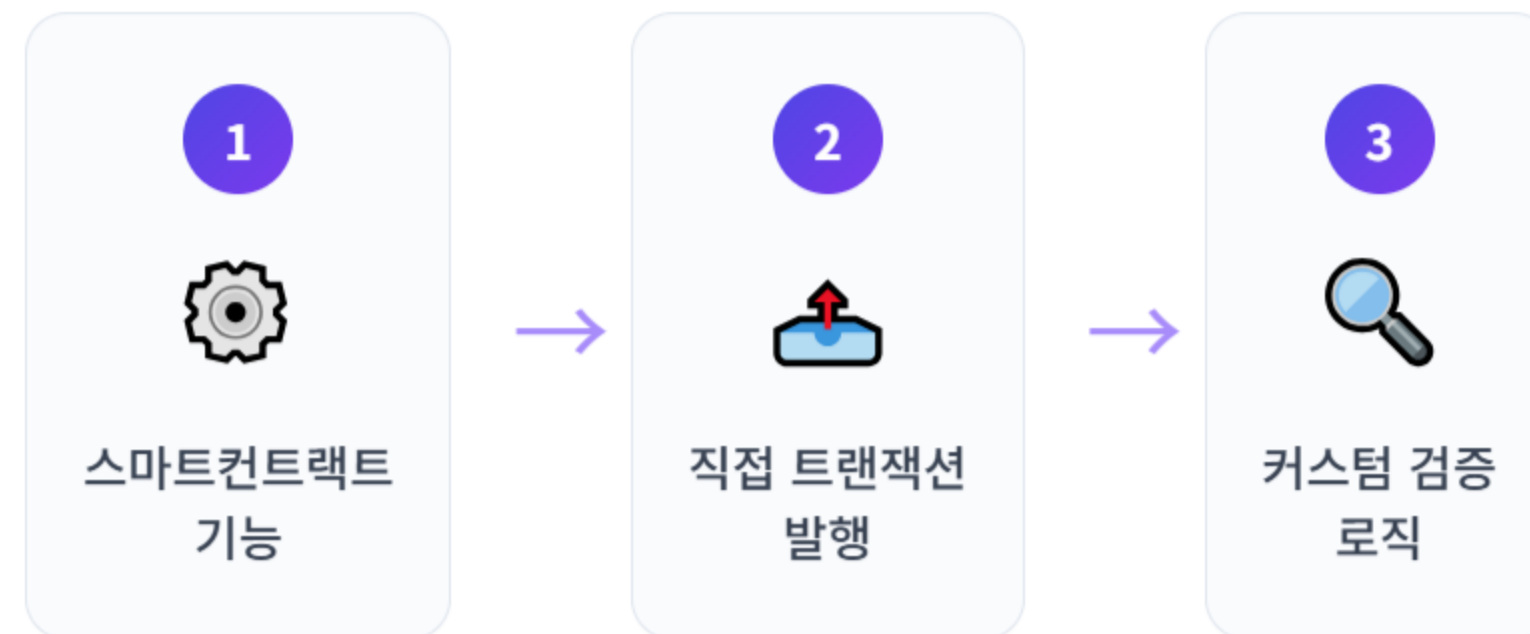
■ Account Abstraction의 제안들

- EIP-86 (2017년)
 - 초창기 Account Abstraction 논의
- ERC-2938 (2020년)
 - 새로운 트랜잭션 타입을 도입해 계정 추상화를 구현하려는 제안
- ERC4337 (2021년)
 - 하드포크 없이 Account Abstraction 구현 제안



■ Account Abstraction의 제안들

- EIP-86 (2017년)
 - 제안자 : Vitalik
 - 목표 : 사용자 계정을 **스마트컨트랙트처럼 동작하게 만들자!**(초기 Account Abstraction)
 - 구상
 - **지갑 계정에서 커스텀 로직 실행** 가능(Ex. 하루에 보낼 수 있는 금액 제한)
 - 한계
 - 네트워크가 기존의 EOA 전용(ECDSA) 검증 구조를 버리고, **코드 실행 기반 검증 구조로 변경** 필요
 - 프로토콜 레벨 큰 변경 필요..!(하드포크급 변화 필요)

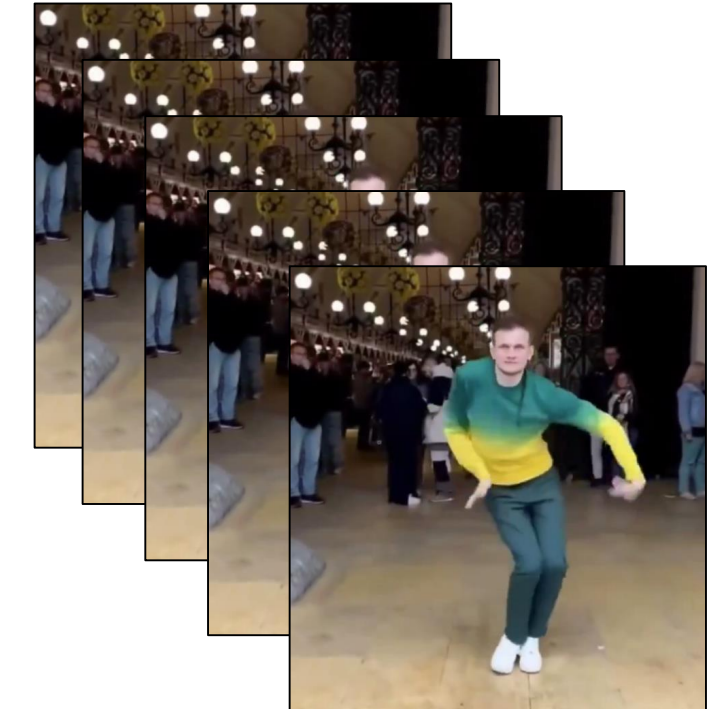


The Lecture : ERC-4337

■ Account Abstraction의 제안들

■ ERC-2938 (2020년)

- 제안자 : Vitalik 외 4명
- 목표 : 새로운 트랜잭션 타입 추가를 통한 계정 추상화 실현
- 구상
 - 기존 Tx에 새로운 필드(initCOde, validationGas, paymaster) 추가
 - 트랜잭션 처리 과정에 “**VALIDATE**” 단계를 추가하여 지갑이 직접 커스텀 로직 실행 가능
- 한계
 - 프로토콜 레벨 큰 변경 필요..!(하드포크급 변화 필요)



● Before (기존 EOA Tx)

nonce 트랜잭션 순서 번호	
gasPrice 가스 단가	
gasLimit 최대 가스 사용량	
to 수신자 주소	value 전송할 ETH 양
data 실행할 데이터	v, r, s 디지털 서명

새로운 필드
추가

● After (ERC-2938 AA Tx)

기존 필드:
nonce, gasPrice, gasLimit, to, value, data, v/r/s

새로운 필드:

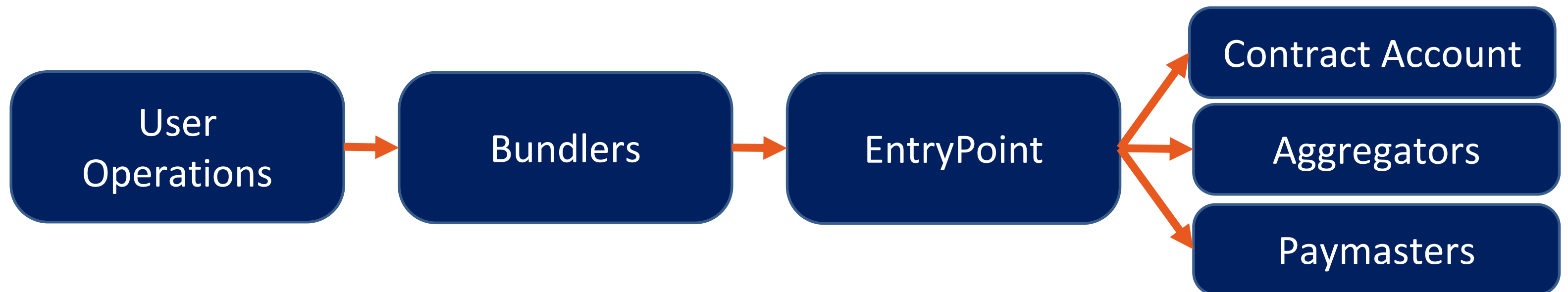
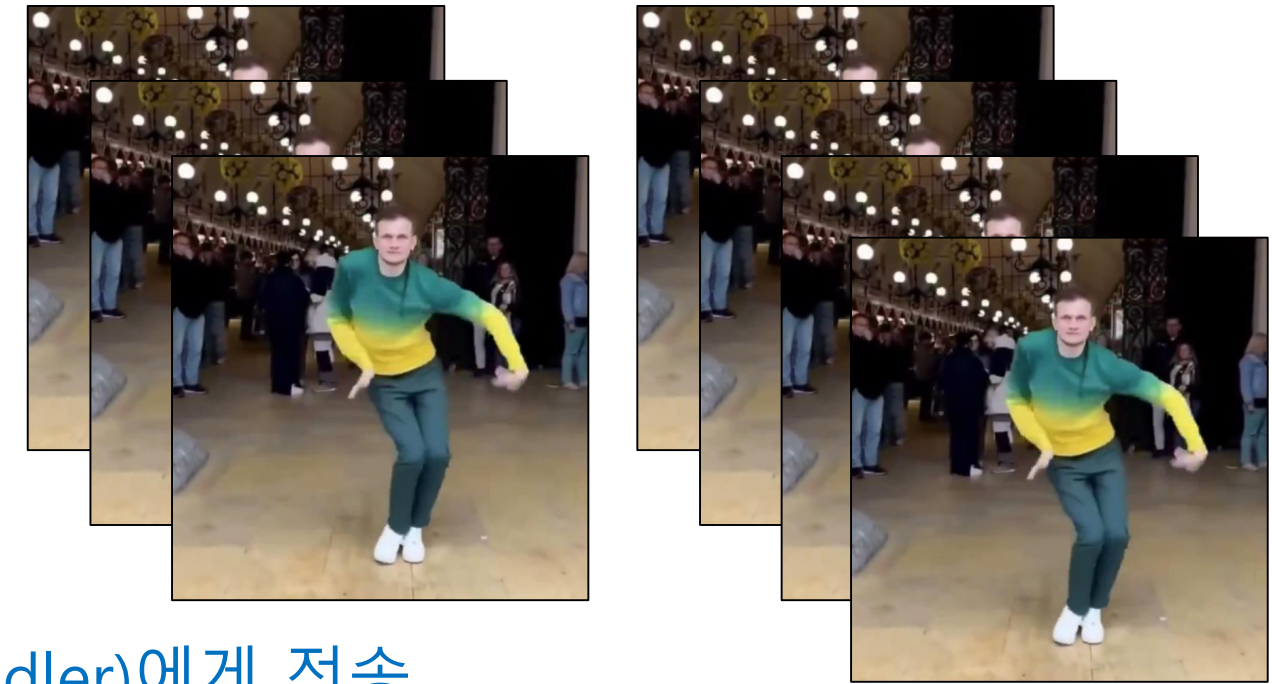
initCode 새로운 계정(스마트 월렛) 배포 로직 포함 → 계정 생성과 동시에 트랜잭션 실행
validationGas 검증 단계에서 소비되는 가스 분리 → 가스 사용량을 단계별로 관리
paymaster 가스비 대납자 지정 가능 → 사용자 대신 제3자가 가스비 지불

The Lecture : ERC-4337

■ Account Abstraction의 제안들

■ ERC-4337 [Account Abstraction Using Alt Mempool] (2021년)

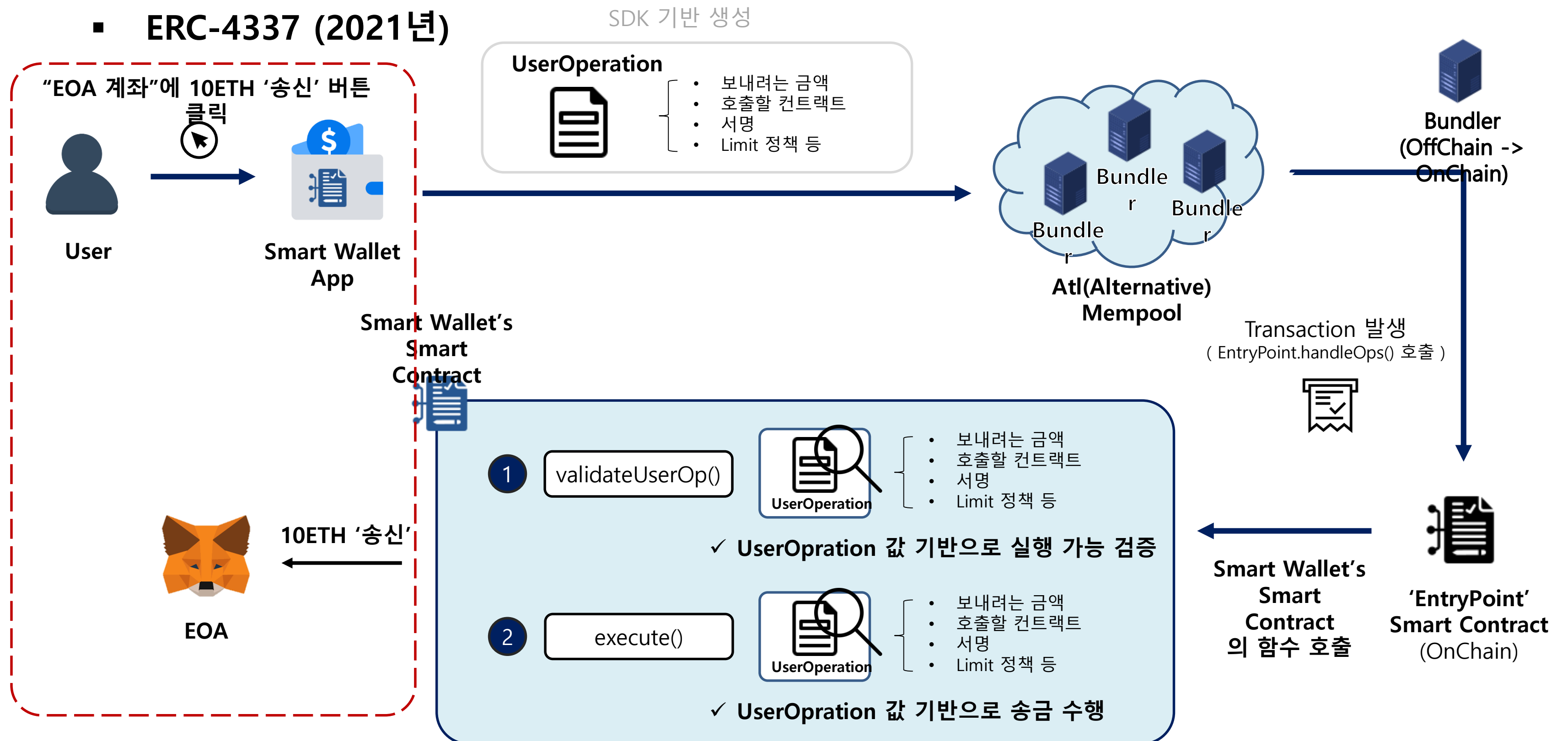
- 제안자 : Vitalik 외 6명
- 목표 : 프로토콜 변경 없이 계정 추상화
- 구상
 - 트랜잭션 대체 구조체(UserOperation)를 만들어 특수 노드(Bundler)에게 전송
 - 특수 노드(Bundler)는 ERC 4337을 지원하는 스마트 컨트랙트(EntryPoint)에게 전송
 - 수신받은 스마트 컨트랙트는 월렛의 VALIDATE 및 EXECUTE 를 호출
- 장점
 - 프로토콜 레벨 큰 변경 필요없이 Account Abstraction를 구현 가능



The Lecture : ERC-4337

Account Abstraction의 제안들

ERC-4337 (2021년)

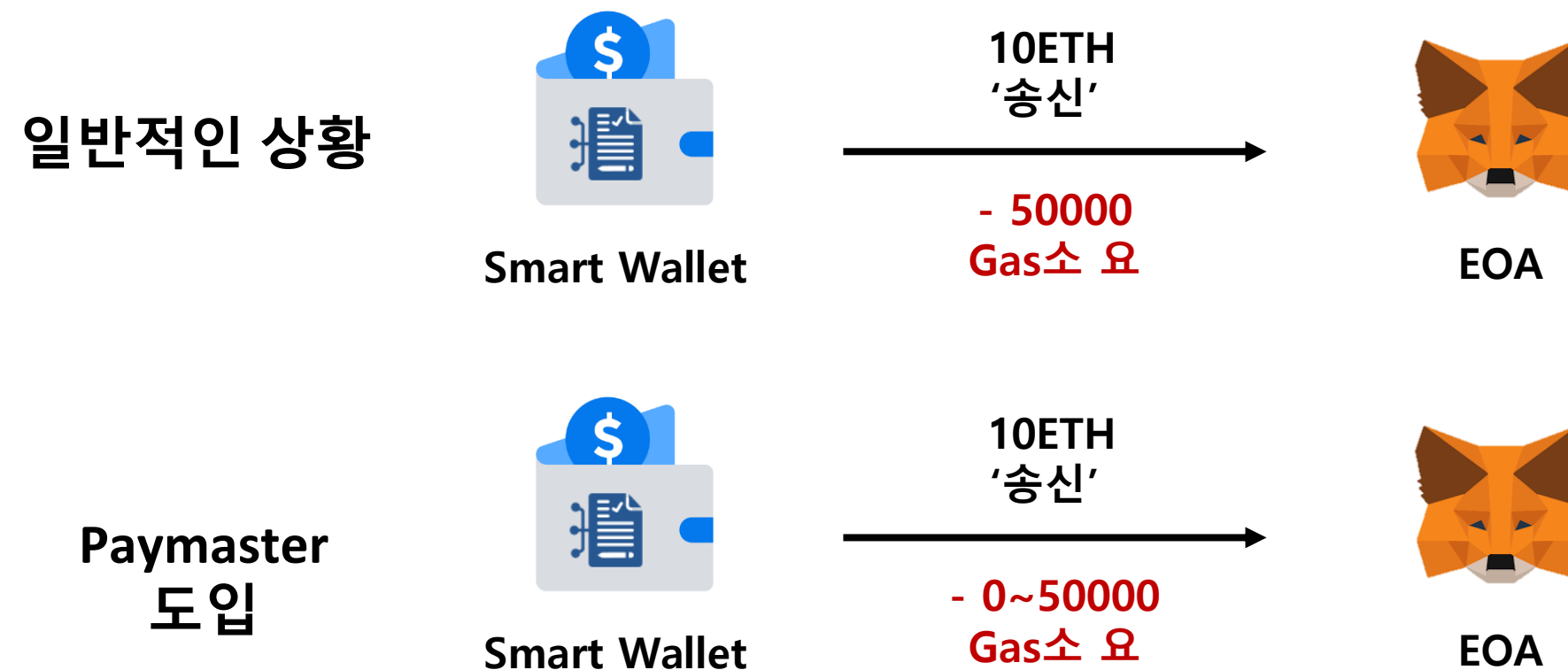


The Lecture : ERC-4337

■ Account Abstraction의 제안들

■ ERC-4337 (2021년) - Paymaster

- 목적 : 사용자 경험(User Experience, UX) 개선
- 구상
 - EntryPoint단에서 Paymaster를 개입시켜 **가스비 대신 지불**
- 장점
 - 수수료 대납 정책 구성 가능 (Ex. NFT 보유자만 대납 허용, 특정 기간 무료 제공)
 - 조건 기반 제어 (Ex. 사용자의 주소, IP, 메타데이터 기반 트랜잭션 허용 여부 결정)



The Lecture : ERC-4337

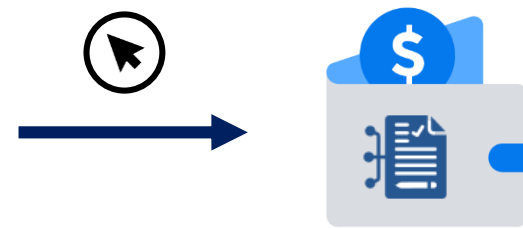
Account Abstraction의 제안들

ERC-4337 (2021년) - Paymaster

"EOA 계좌"에 10ETH '송신' 버튼
클릭



User

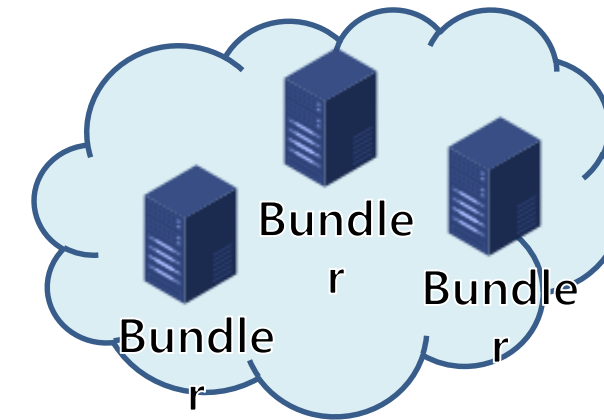


Smart Wallet
App

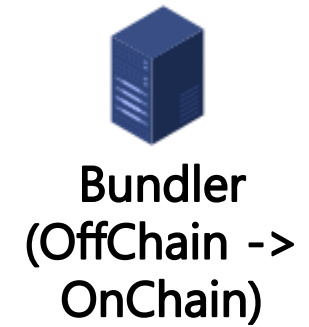
UserOperation



- 보내려는 금액
- 호출할 컨트랙트
- 서명
- **paymasterAndData**



Atl(Alternative)
Mempool



Bundler
(OffChain ->
OnChain)

Transaction 발생
(EntryPoint.handleOps() 호출)



Smart Wallet's
Smart
Contract



1

validateUserOp()



- 보내려는 금액
- 호출할 컨트랙트
- 서명
- Limit 정책 등

✓ UserOperation 값 기반으로 실행 가능 검증

3

execute()



- 보내려는 금액
- 호출할 컨트랙트
- 서명
- Limit 정책 등

✓ UserOperation 값 기반으로 송금 수행

Smart Wallet's
Smart
Contract
의 함수 호출

'EntryPoint'
Smart Contract
(OnChain)

Paymaster's
Smart
Contract

Paymaster's
Smart
Contract
함수 호출

2

Validate
Paymaster
UserOp



✓ UserOperation 값
기반으로 수수료 대납



EOA

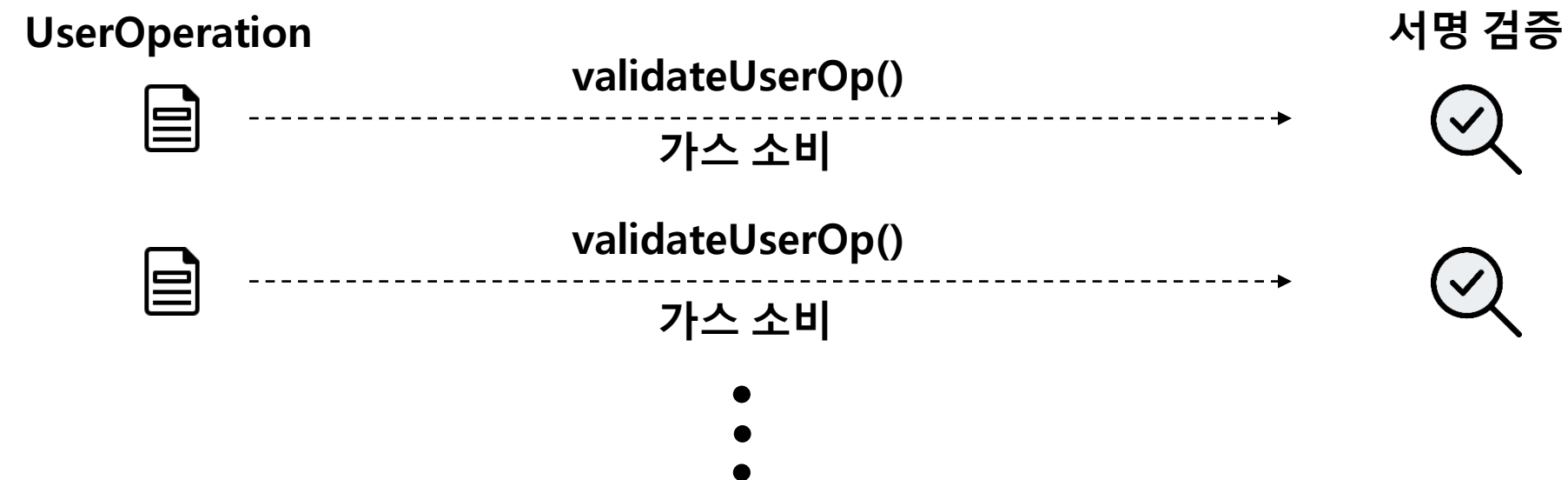
10ETH
'송신'

The Lecture : ERC-4337

■ Account Abstraction의 제안들

▪ ERC-4337 (2021년) – Aggregator

- 목적 : UserOperation에 대한 서명 검증 비용 절감 및 확장성 확보
- 구상
 - EntryPoint는 Aggregator 컨트랙트를 호출해 여러 UserOperation의 서명을 한꺼번에 검증
 - 개별 지갑(Account Contract)이 아니라, Aggregator가 “묶어서 유효하다”는 증명 제공
- 장점
 - 가스 절약: 동일한 서명 스킴(예: BLS, 다중 서명)을 묶어서 한 번에 검증
 - 확장성: 번들러가 수십~수백 개의 UserOperation을 하나의 블록 안에서 처리 가능
 - 유연성: 특정 서명 알고리즘(BLS, Schnorr 등)을 사용하는 지갑을 지원



Account Abstraction의 제안들

- ERC-4337 (2021년) – Aggregator
 - Aggregator에 적합한 서명 알고리즘 : BLS, Schnor
 - Aggregator에 적합한 서명 알고리즘 특징
 - 여러 서명을 하나의 "합성 서명"으로 결합 가능
 - 합성된 서명을 단 한 번 검증해서 여러 서명의 유효성을 한번에 확인 가능

개별 서명

유저 A : 메시지 $m_1 \rightarrow$ 서명 σ_1 ✓

유저 B : 메시지 $m_2 \rightarrow$ 서명 σ_2 ✓

유저 C : 메시지 $m_3 \rightarrow$ 서명 σ_3 ✓

유저 D : 메시지 $m_4 \rightarrow$ 서명 σ_4 ✓

유저 E : 메시지 $m_5 \rightarrow$ 서명 σ_5 ✓

합성 서명

$$\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4 + \sigma_5 \\ = \sigma_{agg} \quad \checkmark$$

The Lecture : ERC-4337

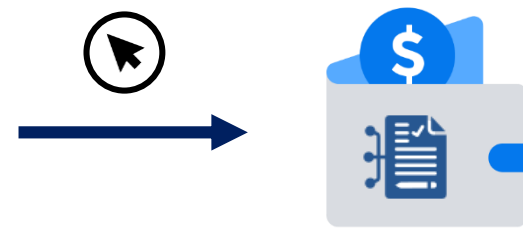
Account Abstraction의 제안들

ERC-4337 (2021년) – Aggregator

“EOA 계좌”에 10ETH ‘송신’ 버튼
클릭



User

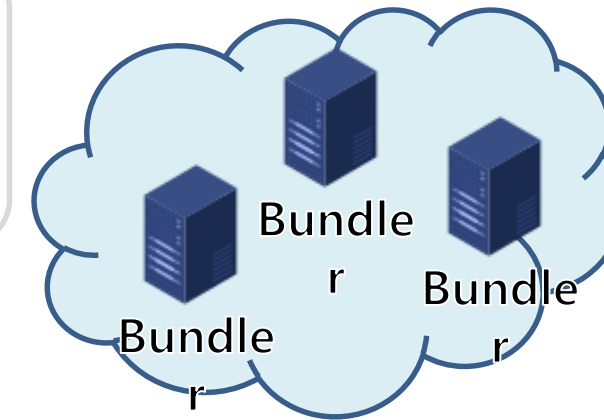


Smart Wallet
App

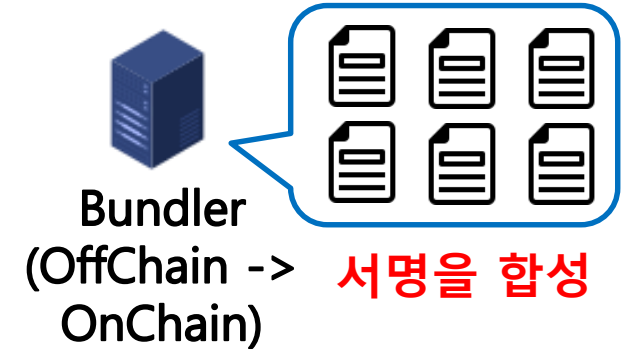
UserOperation



- 보내려는 금액
- 호출할 컨트랙트
- 서명
- **Aggregator가 활용할 수 있는 데이터**



At(Alternative)
Mempool



Bundler
(OffChain ->
OnChain)

서명을 합성

Transaction 발생
(EntryPoint.handleOps() 호출)



Smart Wallet's
Smart
Contract



2

execute()



- 보내려는 금액
- 호출할 컨트랙트
- 서명
- **Aggregator 활용 데이터**

✓ UserOperation 값 기반으로 송금 수행

Smart Wallet's
Smart
Contract
의 함수 호출

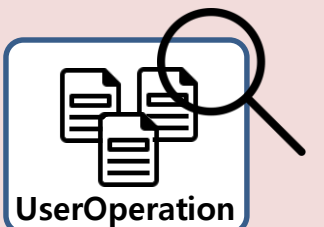
'EntryPoint'
Smart Contract
(OnChain)

Aggregator's
Smart
Contract
함수 호출

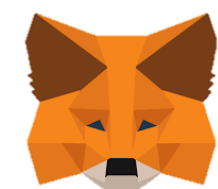
Aggregator's
Smart

1

Validate
Signatures



✓ UserOperation들의
합성 서명 값을
한번에 검증



EOA

10ETH
'송신'

The Lecture : ERC-4337

■ Account Abstraction의 제안들

- ERC-4337 (2021년)가 직면한 한계적인 상황
 - 한계점 : 기존 EOA 대비 2배 이상 가스 소모할 가능성 ▲
 - 원인
 - ERC-4337은 여러 컨트랙트(EntryPoint → SmartWallet → Paymaster)를 차례로 호출
 - 각 컨트랙트를 호출하다보니 가스 비용 증가
 - 이는 EOA 대비 가스량 소모가 증가

계약 종류	업무	가스 증가 이유
EntryPoint	UserOperation 수신 및 처리	컨트랙트 호출 오버헤드 + calldata 파싱 + 상태 검증
Smart Wallet (지갑 컨트랙트)	서명 검증, nonce 처리	사용자마다 커스터마이징된 검증 로직 실행 비용
Paymaster (선택적)	수수료 대납 및 검증	수수료 지불 조건 확인, 지불 승인 등 추가 로직

The Lecture : ERC-4337

■ Account Abstraction의 제안들

- ERC-4337 (2021년)가 직면한 한계적인 상황
 - 대응방안 : Rollup 기반 Layer 2 사용
 - 내용
 - Rollup은 수많은 트랜잭션을 압축하여 한 번에 L1에 업로드
 - L2에서는 gas 단가가 훨씬 저렴 (EntryPoint, SmartWallet, Paymaster 호출 비용도 부담 없음)
 - 실제 실행은 L2에서 수행하고, 결과만 L1에 기록하여 보안성 유지



The Lecture : ERC-4337

■ Account Abstraction(ERC-4337) 구현 시 핵심 포인트

- validateUserOp()는 Smartwallet에 구현
 - 이유
 - EntryPoint는 SmartWallet의 트랜잭션 실행을 직접 호출하는 주체임
 - 따라서 누구나 EntryPoint를 통해 SmartWallet을 우회 호출할 수 있는 구조
 - 그렇기에 SmartWallet쪽에서 직접 서명을 검증(Validate) 해야함

```
1  contract MySmartWallet is IAccount {
2      ...
3      // ✅ EntryPoint가 호출하는 핵심 검증 함수
4      function validateUserOp(
5          UserOperation calldata userOp,
6          bytes32 userOpHash,
7          uint256 missingFunds
8      ) external override returns (uint256) {
9          require(msg.sender == entryPoint, "Not from EntryPoint");
10         require(recover(userOpHash, userOp.signature) == owner, "Invalid sig");
11
12         if (missingFunds > 0) {
13             payable(entryPoint).transfer(missingFunds);
14         }
15         return 0;
16     }
17
18     function execute(...) external { ... }
19 }
```

The Lecture : ERC-4337

■ Account Abstraction(ERC-4337) 구현 시 핵심 포인트

- Smartwallet에서 nonce 검증 필수 구현
 - 이유
 - EntryPoint는 nonce를 추적하지 않음
 - ERC-4337에서는 SmartWallet이 직접 nonce를 관리해야 함
 - 만약 검증하지 않는다면 공격자가 유효한 UserOperation을 여러 번 재전송 가능

```
1 // 스마트월렛 내부 상태 변수로 nonce 선언
2 uint256 public nonce;
3
4 function validateUserOp(
5     ...
6 ) external override returns (uint256) {
7     require(msg.sender == entryPoint, "Only EntryPoint");
8
9     // ✅ Nonce 검증: 중복 요청(리플레이 공격) 방지
10    require(userOp.nonce == nonce, "Invalid nonce");
11
12    // ✅ 검증 통과 후 nonce 증가
13    nonce++;
14
15    ...
16    return 0;
17 }
```

The Lecture : ERC-4337

■ Account Abstraction(ERC-4337) 구현 시 핵심 포인트

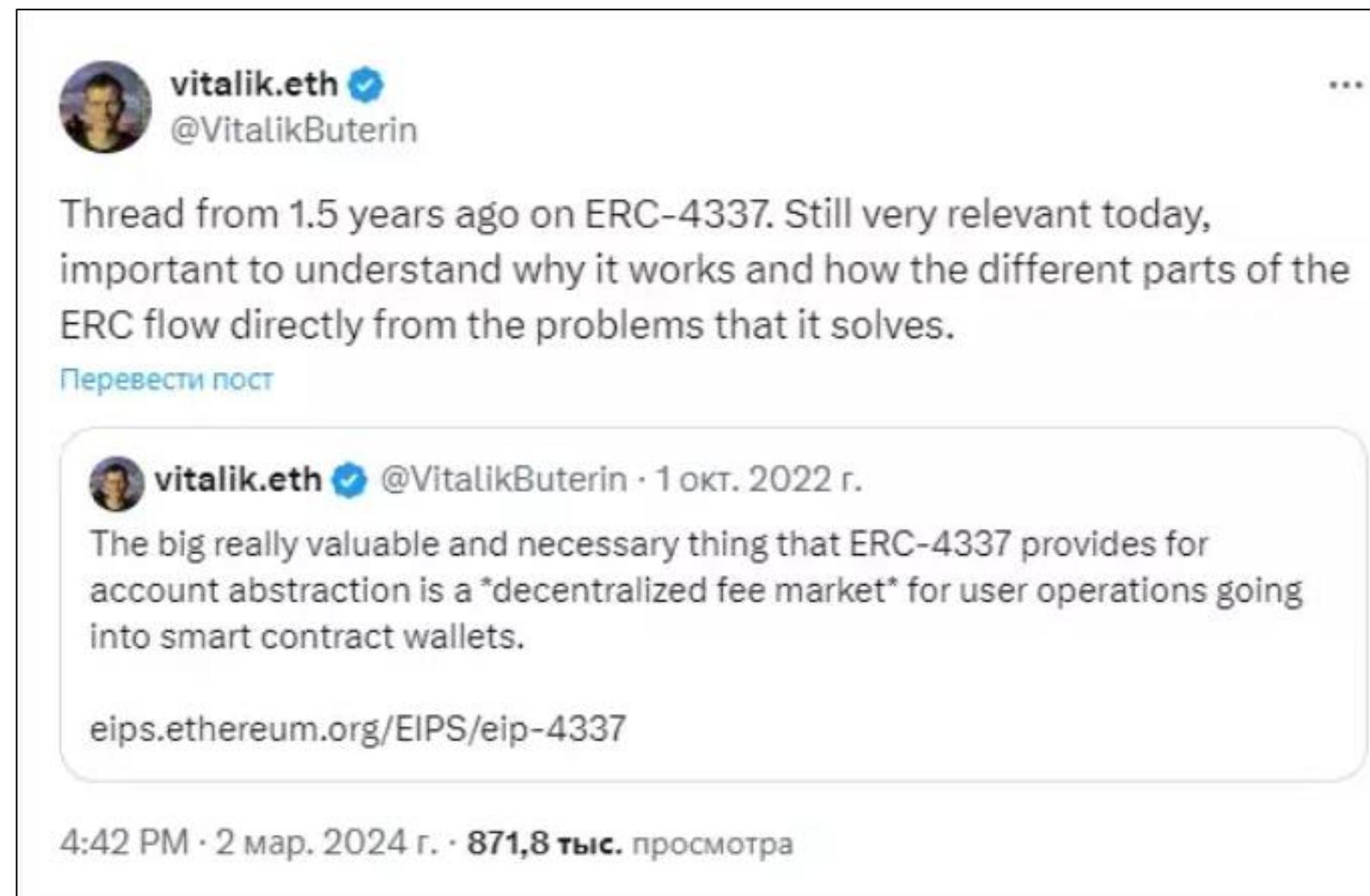
- EntryPoint 예치금 잔고 확인은 필수
 - 이유
 - Bundler는 UserOperation을 처리하며 선 가스비를 본인이 부담하는 구조
 - validateUserOp() 이후 SmartWallet의 예치금이 부족할 경우, 트랜잭션 중단
 - 트랜잭션 중단되면 Bundler만 가스비만 소요되고 종료됨

```
1  function validateUserOp(...) external returns (uint256) {
2      require(msg.sender == entryPoint, "only EntryPoint");
3
4      // !잔고 부족하면 revert
5      if (address(this).balance < requiredPrefund) {
6          revert("Insufficient prefund");
7      }
8      ...
9  }
```


The Lecture : ERC-4337

■ ERC-4337, Web3 지갑의 미래

- 비탈린은 “탈중앙화된 수수료 시장(decentralized fee market)”의 가치를 강조
 - 사용자는 ETH 없이도 (ERC-20 토큰, 제3자 대납) 쉽게 Web3에 진입 가능
 - 특정 검증자/채굴자에 의존하지 않고 누구나 참여 가능한 경쟁 시장 형성
 - 결과적으로 Web2처럼 누구나 편하게 쓸 수 있는 지갑 UX 완성



감사합니다.