

# Hydra Project

Hyperconnected layer 2 for Decentralized Aggregator

엄정용, 장제인, 한종빈

# Hydra Project

- Multichain supported Layer 2 Order Book DEX
- 체인 상의 코인 및 토큰을 Layer 2인 Hydra로 전송하여 오더북 기반으로 거래
- 이더리움 및 포크 체인 연결하여 안정적인 마켓 구성

# Background

# Multichain

- 다양한 체인이 각자의 목적과 역할에 따라 생태계를 형성
- 과거 타체인에 대한 배타적인 입장을 버리고 타체인을 인정하고 공존
- 상호운영성에 중점
- [Event] Mainnet 2021 I: 멀티체인 생태계는 다가오는가.

# Cross Chain

- 체인의 자산 등의 상태를 다른 체인으로 전송
- 일반적으로 브리지를 통하여 자산을 이동하여 다른 체인에서 전송, 거래 등에 사용
- Hydra는 멀티체인 자산을 하나의 Layer 2로 전송시켜 거래 후 원하는 체인으로 출금
- 단순히 Wrapped Token으로 전송하는 구조가 아닌 마켓을 형성하고 거래

# Layer 2

- 확장성 문제를 해결하기 위하여 도입
- Layer 1이 아닌 오프체인 솔루션을 통해 문제를 해결
- Layer 2는 Layer 1과 다른 트랜잭션 비용 구조를 가질 수 있음
- Layer 1에서 발생할 수 있는 트랜잭션 비용을 감소 시키거나 제거할 수 있음
- Plasma, Rollups (Optimistic Rollup, ZK Rollup), Side Chain

# DEX

- Decentralized Exchange
- 중앙화된 거래소를 탈피
- 온체인 상에서는 AMM 방식으로 구현된 것이 일반적
- 대표적인 덱스로 Uniswap, SushiSwap 등이 존재

# Order Book







- 주식 시장이나 암호화폐 시장 등에 존재하는 구매자와 판매자의 주문을 기록한 목록
- 구매자와 판매자를 연결하여 입찰 가격이 오더북 주문의 최저가보다 같거나 높으면 거래
- 모든 주문이 트랜잭션으로 처리되기 때문에 온체인에서 운영되기 어려움
- 낮은 TPS와 높은 거래 비용이 해결되어야 가능

호가창			거래		일반호가		누적호가		호가주문	
규모	BTC	가격	USD	채굴						
0.1145		56774		-		0.001	70,514,000	-0.04%	거래량 거래대금	11,671 BTC 822,350 백만원 (최근24시간)
0.5000		56773		-		0.060	70,510,000	-0.04%		
0.2159		56772		-		0.002	70,509,000	-0.04%		
6.8317		56769		-		0.064	70,503,000	-0.05%		
0.4424		56768		-		0.027	70,502,000	-0.05%		
0.7000		56766		-		1.432	70,501,000	-0.05%	52주 최고	82,700,000 (2021.11.09)
2.8574		56765		-		1.618	70,500,000	-0.06%	52주 최저	19,216,000 (2020.12.09)
0.3172		56763		-		1.733	70,499,000	-0.06%	전일종가	70,539,000
2.0141		56761		-					당일고가	70,991,000 +0.64%
0.4485		56754		-	체결강도	+92.84%	70,497,000	-0.06%	당일저가	70,032,000 -0.72%
0.1000		56750		-	체결가	체결량				
2.8574		56749		-	70,497,000	0.006	70,496,000	-0.06%		0.676
10.3370		56748		-	70,498,000	0.001				
10.0684		56747		-	70,498,000	0.132	70,495,000	-0.06%		0.052
35.5973		56744		-	70,498,000	0.149				
스프레드 (Spread)		1	0.01%		70,497,000	0.003	70,493,000	-0.07%		0.092
6.6646		56743		-	70,497,000	0.014				
9.0104		56742		-	70,497,000	0.831	70,449,000	-0.13%		0.033
3.1099		56741		-	70,498,000	0.005				
0.3537		56738		-	70,498,000	0.217	70,442,000	-0.14%		0.378
2.8574		56737		-	70,499,000	0.061				
9.4342		56736		-						
6.8318		56732		-						
2.8574		56731		-						
0.0141		56725		-						
0.3086		56724		-						
4.7851		56723		-						
0.1713		56721		-						
0.0300		56720		-						
2.7290		56719		-						
0.6654		56718		-						
0.5070		56717		-						



# Ethereum Fork Chain

- 동일한 구조와 주소체계를 갖고 있어 Layer 2 연결 부담이 적으며 Web3 등 활용 가능
- EVM을 사용하고 있기 때문에 하나의 스마트 컨트랙트 개발로 여러 체인에 배포 가능
- 상품화된 Layer 2 솔루션 사례가 존재하여 활용 가능성 및 안정성 높음

#	이름	가격	24h %	7d %	시가총액 ⓘ	거래량 (24시간) ⓘ	유통 공급량 ⓘ	최근 7일
☆ 1	 Bitcoin BTC <a href="#">구매하기</a>	\$64,970.76	-2.16%	-3.87%	\$1,224,264,167,577	\$49,701,769,281 766,065 BTC	18,869,862 BTC	
☆ 2	 Ethereum ETH <a href="#">구매하기</a>	\$4,671.74	-0.85%	-2.33%	\$552,076,111,803	\$23,270,097,935 4,986,145 ETH	118,294,800 ETH	
☆ 3	 Binance Coin BNB <a href="#">구매하기</a>	\$626.21	-1.27%	-12.01%	\$104,329,401,232	\$3,844,113,798 6,145,943 BNB	166,801,148 BNB	

**Why need Hydra?**

# Why need Hydra?

- Multichain 시대가 도래함에 다양한 체인의 코인 및 토큰의 가치를 인정하고 거래
- 거래에 대한 필요는 커졌지만, 현재 중앙화된 거래소를 제외하고는 유니스왑 등의 AMM 거래소가 대부분
- 또한 다른 체인의 자산과 거래하기 위해서는 브리지를 통해 상대방 체인으로 자산을 이동시키고 유동성 풀을 구성하여 거래
- 충분한 유동성 풀이 구성되지 않은 경우, 슬리피지를 감당해야하며, 게으른 유동성 문제 등도 존재함

# Why need Hydra?

- 이더리움 및 이더리움 포크 체인을 연결하여 자유롭게 거래 가능한 마켓을 형성
- Layer 2가 허브가 되어 각 체인을 연결, 코인 및 토큰을 전송하여 거래
- 오더북 기반으로 자유롭게 거래하고 이를 원하는 체인의 코인, 토큰으로 출금
- Layer 2의 이점인 높은 처리량, 빠른 속도, 낮은 거래비용을 활용

# 체인 연결 가능성

- 이더리움을 Layer 1으로 하는 Layer 2 솔루션을 이더리움 및 포크 체인에 적용
- Layer 1 <-> Layer 2 자산이동 방식을 동일하게 적용 가능
- 동일 스마트 컨트랙트 코드를 재사용 가능하여 개발 범위도 상대적으로 적음
- Layer 1 자산을 Layer 2로 이동하여 거래 후 출금하는 일반적인 형태의 앱
- 상태를 두 개 이상의 체인에 제출, 검증하는 부분만 차이

Layer 2

# Layer 2 기술 검토

- Plasma
- Side Chain
- Optimistic Rollup
- ZK Rollup

# Plasma

- Joseph Poon, Vitalik Buterin
- 스마트 컨트랙트를 이용한 예치, 출금
- 머클 트리



# Plasma

## How does Plasma work

- 스마트 컨트랙트를 이용하여 Layer 1에서 플라즈마로 예치하여 자산 이동
- 플라즈마를 통해 오프체인 거래를 통해 상태 변환
- 플라즈마는 주기적으로 State Root를 Layer 1으로 제출
- 출금시 사기 증명을 거쳐야 하기 때문에 출금에 1~2주 정도 시간이 필요

# Plasma

## Problems

- 운영자의 악의적 행동 및 태만을 감시하기 위해 사용자 혹은 별도의 감시자가 체인을 지속적으로 감시해야함
- 예치된 금액을 출금하기 위해서는 1~2주의 시간이 필요하여 사용자 경험 측면에서 좋지 못함

# Side Chain

- Layer 1과 병렬로 운영되는 별도의 체인
- 높은 처리량 및 처리 속도를 개선하기 위하여 이더리움과 다른 합의 알고리즘 사용
- EVM 지원 가능

# Side Chain

## How do Sidechains work

- 아톰릭 스왑, 브리지 등을 통해 Layer 1의 자산을 이동
- 자체적으로 보안을 책임
- PoA, PoS 등의 Layer 1과 다른 컨센서스 알고리즘을 사용
- Layer 2의 거래가 완료되면 브리지를 통해 Layer 1으로 자산 출금

# Side Chain Problems

- 사이드 체인은 Layer 1과 병렬로 작동하는 별도의 체인이 존재하는 것이기 때문에, 보안을 자체적으로 유지해야함
- 충분한 보안 수준을 달성하지 못하는 경우 공격을 당할 수 있음
- 단일 앱 운영 수준에서 벗어나는 규모

# Optimistic Rollup

- Tx Batch, State Batch
- 전체 트랜잭션과 State Root를 Layer 1에 기록
- OVM (Optimistic Virtual Machine)
- Fraud Proof

# Optimistic Rollup

## How does Optimistic Rollup work

- Layer 1의 자산을 OR으로 전송하기 위해 예치
- OR에서 Layer 1을 관찰하여 예치된 자산만큼 자산을 Layer 2에 생성
- OR은 트랜잭션을 배치로 만들어 Layer 1으로 전송
- 거래 완료 후 Layer 1으로 자산 출금 요청
- 1~2주의 Challenge 기간을 거쳐 출금 완료

# Optimistic Rollup

## Problems

- OR에서 제출되는 상태를 지속적으로 검증해주는 검증인이 존재하면 정상적으로 처리된다는 가정에 의해 작동
- 출금 시에 1~2주의 Challenge 기간이 필요하기 때문에 사용자 경험이 좋지 않다는 단점이 존재



# ZK-STARKs

- ZK Rollup
- Off-chain state (Balances Tree, Orders Tree)
- starkKey
- Cairo
- SHARP (Shared Prover)

# ZK-STARKs

## How do ZK-STARKs work

- starkKey 발급 및 등록
- Layer 1의 Stark Contract를 통해 예치, Layer 2에서 자산 생성
- App으로 거래 요청, 트랜잭션을 StarkEx Service로 전달
- StarkEx Service에서 상태는 머클 트리로 구조화되며, 머클 루트는 Layer 1로 제출
- StarkEx Service는 트랜잭션 배치를 SHARP로 전달하여 증명을 생성
- 생성된 증명은 온체인 상에서 STARK Verifier 컨트랙트를 통해 검증

# ZK-STARKs

## Problems

- 이슈가 발생하는 경우, 탈출에 대한 전략은 있으나 서비스 중단 시 해결 방안 부재
- 최근 AWS 이슈로 dYdX 중단 사태
- 온체인에서 운영되는 다른 DApp에 비교하여 한계 존재

**ZK-STARKs**

# ZK-STARKs

- 상대적으로 우수한 확장성
- dYdX 등 오더북 기반 서비스로 검증된 Layer 2
- 증명 생성 및 검증을 통해 Optimistic Rollup과 같은 Challenge 기간 불필요

# ZK-STARKs 문제점

- 온체인 상으로 Root만 제출되기 때문에 데이터 가용성 측면에서 한계 존재
- 서비스 중단 시 탈출에 대한 전략만 존재

# 개선 제안

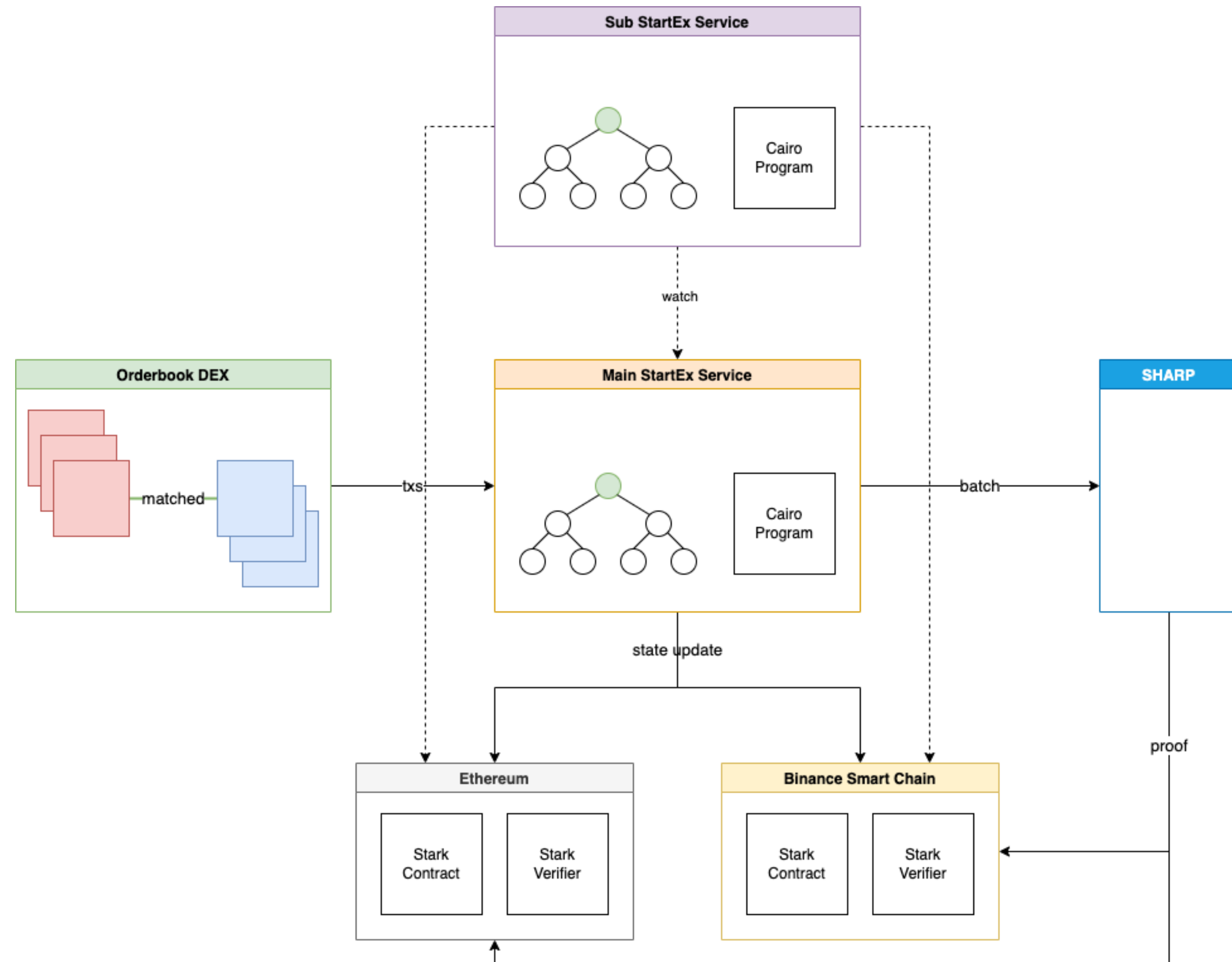
- 모든 트랜잭션을 동기화하여 데이터 가용성 확보하고 검증하는 서브 서비스
- Main-Sub 구조의 서비스 운영
  - 메인 서비스가 중단되는 경우, 서브 서비스가 메인 서비스를 대체하여 서비스 수행

# 아키텍처

- Order Book DEX Application
- StartEx Service
  - Cairo Program for Order Book DEX
- SHARP (Shared Prover)
- Layer 1 Smart Contract
  - Stark Contract
  - Stark Verifier



# 아키텍처



# 풀어야하는 문제

- StarkEx Service 및 SHARP는 단일 체인에 대한 Layer 2이기 때문에, 이를 멀티 체인 Layer 2로 변경이 필요
- 앞에서 제안된 탈중앙성, 데이터 가용성, 서비스 중단에 대한 안정성 개선을 위한 서브 서비스 추가

# Smart Contract

- Perpetual Smart Contract
  - 입출금 담당 스마트 컨트랙트 (Deposit, Withdraw, Registration)
- FRI Statement
  - FRI 검증 컨트랙트
- Merkle Statement
  - 머클 증명 컨트랙트

# Smart Contract

```
function deposit(
    uint256 starkKey,
    uint256 assetType,
    uint256 vaultId,
    uint256 quantizedAmount
) public notFrozen {
    // The vaultId is not validated but should be in the allowed range supported by the
    // exchange. If not, it will be ignored by the exchange and the starkKey owner may reclaim
    // the funds by using depositCancel + depositReclaim.

    // No need to verify amount > 0, a deposit with amount = 0 can be used to undo cancellation.
    require(!isMintableAssetType(assetType), "MINTABLE_ASSET_TYPE");
    require(isFungibleAssetType(assetType), "NON_FUNGIBLE_ASSET_TYPE");
    uint256 assetId = assetType;

    // Update the balance.
    pendingDeposits[starkKey][assetId][vaultId] += quantizedAmount;
    require(pendingDeposits[starkKey][assetId][vaultId] >= quantizedAmount, "DEPOSIT_OVERFLOW");

    // Disable the cancellationRequest timeout when users deposit into their own account.
    if (
        isMsgSenderKeyOwner(starkKey) && cancellationRequests[starkKey][assetId][vaultId] != 0
    ) {
        delete cancellationRequests[starkKey][assetId][vaultId];
    }

    // Transfer the tokens to the Deposit contract.
    transferIn(assetType, quantizedAmount);

    // Log event.
    emit LogDeposit(
        msg.sender,
        starkKey,
        vaultId,
        assetType,
        fromQuantized(assetType, quantizedAmount),
        quantizedAmount
    );
}
```

# Cairo Program

```
1  %lang starknet
2  %builtins pedersen range_check
3
4  from starkware.cairo.common.cairo_builtins import HashBuiltin
5
6  struct Order:
7      member my_asset_id : felt
8      member my_asset_amount : felt
9      member target_asset_id : felt
10     member target_asset_amount : felt
11     member account_id : felt
12 end
13
14 struct BuyOrder:
15     member buy_order_id : felt
16     member state : felt
17     member order : Order
18 end
19
20 struct SellOrder:
21     member sell_order_id : felt
22     member state : felt
23     member order : Order
24 end
25
26 @storage_var
27 func buy_order(buy_order_id : felt) -> (buy_order : BuyOrder):
28 end
29
30 @storage_var
31 func sell_order(sell_order_id : felt) -> (sell_order : SellOrder):
32 end
33
34 @external
35 func add_buy_order{
36     syscall_ptr : felt*,
37     pedersen_ptr : HashBuiltin*,
38     range_check_ptr
39 } (
40     buy_order_id : felt,
41     my_asset_id : felt,
42     my_asset_amount : felt,
43     target_asset_id : felt,
44     target_asset_amount : felt,
45     account_id : felt
46 ):
47     let new_order = Order(my_asset_id, my_asset_amount, target_asset_id, target_asset_amount, account_id)
48     let new_buy_order = BuyOrder(buy_order_id, 0, new_order)
```

**End of Document**