CS251: Cryptocurrencies and Blockchain Technologies **Fall 2023**

Assignment #3

Due: 11:59pm on Tuesday, **Nov. 7, 2023**
Submit via Gradescope (each answer on a separate page) code: **7DVJKY**

Problem 1. Liquidations. In Lecture 9 we discussed lending protocols. Our friend Bob uses a lending protocol to borrow ETH and AXS against his USDC collateral. He has the following debt position:

$$+1000 \text{ USDC}, \quad -0.2 \text{ ETH}, \quad -20 \text{ AXS}.$$

The collateral factor for these assets are: $0.9, 0.8., 0.7$ for USDC, ETH, and AXS, respectively.

a. Suppose the current exchange rates for USDC and AXS are 1500 USDC/ETH and 100 AXS/ETH. What is the health of Bob's debt position as defined in the lecture (slide 29)? Does Bob's collateral need to be liquidated?

Solution: *The value of the collateral is 1000 USDC * (1 ETH / 1500 USDC) * 0.9 = 0.6 ETH. The value of the total debt is 0.2 ETH + 20 AXS * 1 ETH / 100 AXS = 0.4 ETH. Thus, the health of the debt position is 1.5, meaning his position doesn't need to be liquidated as the health is above 1.*

b. Suppose the exchange rates for AXS changes to 48 AXS/ETH (making AXS more valuable than before) and the other exchange rates remain unchanged. What is the health of Bob's debt position now? Does Bob's collateral need to be liquidated?

Solution: *The value of the collateral stays the same, the debt value changes however to 0.2 ETH + 20 AXS * 1 ETH / 48 AXS = 37/60 ETH. The health of the position is now 36/37 which is smaller than 1. Hence, a part of the collateral has to be liquidated until the health increases up to 1 again.*

c. For parts (a) and (b), if you concluded that Bob's collateral needs to be liquidated, then consider a liquidiator who is willing to clear Bob's ETH debt in exchange for Bob's USDC collateral at a rate of 1520 USDC/ETH. How much USDC will Bob lose from his collateral so that his debt position becomes healthy again. Please round your answer to the smallest integer that makes Bob's debt position healthy.

Solution: *The collateral will be liquidated until the health of the position is back at 1. Denote the amount of liquidated USDC as x:*

$$health(x) := \frac{\frac{(1000-x)}{1500} \cdot 0.9}{0.2 - \frac{x}{1520} + \frac{20}{48}} = 1. \tag{1}$$

The smallest integer x for which the health increases above 1 is 288. Hence, 288 USDC of the collateral have to be liquidated to make the position healthy again.

Problem 2. Slippage. In Lecture 10 we discussed the constant product market maker $xy = k$ used by Uniswap. Suppose Alice wants to buy Δx type X tokens from Uniswap. We showed in the lecture that, assuming no fees ($\phi = 1$), she would have to send $\Delta y = y \cdot \Delta x/(x - \Delta x)$ type Y tokens to Uniswap to maintain the $xy = k$ invariant (see also this short writeup). Therefore, the exchange rate Alice is getting from Uniswap is

$$\frac{\Delta y}{\Delta x} = \frac{y}{x - \Delta x}.$$

In the open market, the exchange rate is some value p. Let us define the *slippage* s as

$$s = \frac{(\Delta y/\Delta x) - p}{p}.$$

This measures the difference in exchange rate between Uniswap and the open market (hence the name *slippage*). If $s = 0$ then the Uniswap exchange rate is the same as on the open market. If $s > 0$ then the Uniswap exchange rate is worse.

Show that the slippage s is always positive, and is approximately $s \approx \Delta x/x$, assuming x is much larger than Δx. Use the fact that we know that $p = y/x$ (by slide 29), and that for a small $\epsilon > 0$ we have $1/(1 - \epsilon) \approx 1 + \epsilon$. Your derivation shows that the exchange rate in Uniswap is always worse than on the open market, however, the larger the liquidity pool, the larger x is, and therefore the smaller the slippage $\Delta x/x$, for a fixed Δx.

Solution:

$$s = \frac{\frac{\Delta y}{\Delta x} - p}{p} = \frac{\frac{y}{x-\Delta x} - p}{p} = \frac{\frac{y}{x-\Delta x} - \frac{y}{x}}{\frac{y}{x}} > \frac{\frac{y}{x} - \frac{y}{x}}{\frac{y}{x}} = 0, \tag{2}$$

for $\Delta x > 0$. Moreover, with the approximation $\frac{1}{1-\epsilon} \approx 1 + \epsilon$ for small ϵ, we have:

$$s = \frac{\frac{\Delta y}{\Delta x} - p}{p} = \frac{\frac{y}{x-\Delta x} - p}{p} = \frac{\frac{y}{x-\Delta x} - \frac{y}{x}}{\frac{y}{x}} = \frac{1}{1 - \frac{\Delta x}{x}} - 1 \approx \frac{\Delta x}{x}. \tag{3}$$

Problem 3. Sandwitch attacks. Consider two assets X and Y on the Uniswap exchange. The X pool contains x tokens of type X and the Y pool contains y tokens of type Y. Recall that Uniswap v2 ensures that $x \cdot y = k$ for some constant k. Suppose that Uniswap charges no fees (i.e., $\phi = 1$).

Alice submits a transaction Tx that sends βx tokens of type X to Uniswap, for some $\beta \geq 0$ (here βx means β times x). When Tx executes, Uniswap will send back γy tokens of type Y to Alice, where $\gamma = \frac{\beta}{1+\beta} \in [0, 1)$. This ensures that the constant product is maintained.

Searcher Sam sees Alice's transaction in the mempool and decides to execute a sandwitch attack. Sam issues two transactions Tx_1 and Tx_2 and arranges with the current block proposer that Tx_1 will appear before Alice's transaction in the proposed block and Tx_2 will appear after.

a. Suppose Sam's Tx_1 sends ϵx tokens of type X to Uniswap, for some $\epsilon \geq 0$. Now, when Alice's Tx executes, she will receive back $\gamma' y$ tokens of type Y. What is γ' as a function of ϵ and β? Is she getting more or less tokens of type Y than before?

Solution: *Alice now receives $\gamma^{'} y$ tokens of type Y where $\gamma^{'} = \frac{\beta}{(1+\epsilon)(1+\epsilon+\beta}$. This fraction decreases with increasing ϵ, thus, Alice receives less y tokens than in the case without the previous transaction.*

b. Does Sam's profit increase or decrease with the amount he spends in Tx_1? In other words, is a larger ϵ better or worse for Sam?

Solution: *Sam's profit is given by the amount of X tokens he gets in transaction 2 for the $\frac{\epsilon}{1+\epsilon} y$ Y tokens he received after transaction 1 minus the ϵx X tokens he spent in transaction 1. Doing the calculations leads to a net profit of*

$$profit(x, \epsilon, \beta) = x\epsilon\beta \frac{\beta + \epsilon + 2}{1 + \epsilon(\beta + \epsilon + 2)} \tag{4}$$

X tokens. We can calculate the derivative of this expression with respect to ϵ and since the derivative is always positive, the profit always increases the larger the front-running transaction of the attacker is. Moreover, the profit converges in the limit of ϵ going to ∞ to βx which, thus, is the upper bound for the amount of X tokens the attacker can generate. In this case, Alice would lose all her tokens to Sam.

c. The validator Victor who is proposing the block sees both Sam's transactions. Victor realizes that he can issue both transactions itself and keep all the profit to itself. However, Victor has no X tokens, as needed for Tx_1. Can Victor mount the sandwitch attack without Sam?

Solution: *One idea might be to take a flash loan from a lending protocol such as Aave in order to get sufficient X tokens for a sandwich attack. The problem here for Victor though is that a flash loan would need to be repaid within the same transaction. By definition of a sandwich attack, however, Alice's transaction has to come in between the front-and back-running transaction of Victor's sandwich attack. Hence, the flash loan couldn't be repaid within the same transaction and is therefore not an option anymore. Victor could still get an overcollateralized loan from a lending protocol by posting a transaction depositing the collateral in a lending protocol and taking a loan in X tokens. With these X tokens he could perform a sandwich attack and repay his loan in a later transaction in the same block while keeping the profit of the attack minus the interest of the lending protocol.*

d. In Lecture 11 we discussed MEV-boost. Can Alice use MEV-boost to protect herself from these shenanigans by Sam and Victor?

Solution: *Alice can send her transaction directly to a block builder that she trusts (assume such a entity exists) instead of to the mempool. As the transaction is only visible for the block builder, searchers looking for opportunities of sandwich attacks can't see her transaction until it was included in a proposed block broadcasted by a relayer. However, theoretically a sandwich attack could still be possible if the trusted block builder chooses to betray Alice and send her transaction in the public mempool. This behaviour in the longterm would lead to a loss of trust on site of the users towards this particular block builder and might thus be detrimental.*