CS251: Cryptocurrencies and Blockchain Technologies **Fall 2020**

Assignment #3

Due: 11:59pm on Wednesday, **Oct. 28, 2020**
Submit via Gradescope (each answer on a separate page) code: **92ZNG8**

Problem 1. Howey test. In lecture 11 we discussed the Howey test that defines what is a security using four conditions that must be met. Now, consider the following situation. Alice is a famous Hollywood producer. To finance her next blockbuster film she creates a new ERC-20 AliceToken and sells fifty million tokens to Hollywood studios at a rate of 1 USD per token. Every month after the film premiers, all proceeds from the film for that month are distributed equally among the token holders. For example, if in its first month the movie makes $100M, then every token holder gets $2 per token at the end of the first month. As an ERC-20 token, AliceTokens can be bought and sold on exchanges like Uniswap, and will have a fluctuating exchange rate relative to the USD, depending on how the movie does once it is released. Are AliceTokens a security according to the Howey test?

Problem 2. Stablecoins.

a. A collateralized stablecoin system maintains collateral so that when the price of the stablecoin drops, the collateral can be used to shrink the supply of coins and bring the price back up. Some projects maintain on-chain collateral (like MakerDao, which uses ETH and other asset types for collateral) while others maintain off-chain collateral (like USDT, which uses fiat currencies such as the US dollar for collateral). What is the primary difference between these systems in terms of the required level of trust by the public? Specifically, which design allows the public to verify that the collateral is properly managed and maintained?

b. In MakerDao, what is the purpose of the DAI Savings Rate (DSR)? Why is it that a high DSR can be used to bring up the price of DAI and a low DSR can be used to bring down the price of DAI?

Problem 3. Oracles. In class we discussed the MakerDAO system, where DAI is intended to be a stable currency governed by MKR token holders. A brief description of the MakerDAO system is available here, and a more in-depth description is available here.

Suppose that the MakerDAO pricing oracle (elected by MKR token holders) temporarily malfunctions and advertises that the price of ETH is $1,000, when in reality it is only $100.

a. How might an attacker exploit this situation to make money?

b. Assuming the error is corrected quickly enough not to destroy MakerDAO, who would bear the losses from such an attack? What would cause those losses?

Problem 4. Uniswap. Recall that Uniswap uses the elegant constant product formula, $xy = k$, to determine the exchange rate between two tokens. Assuming no fees ($\phi = 1$), we showed in the lecture that if the true exchange rate between two tokens A and B is M_p, then the market will drive the Uniswap contract to hold x tokens of type A and y tokens of type B, where $y/x = M_p$. Moreover, we showed that the liquidity providers make the most profit when the exchange rate M_p does not change over time (i.e., $r = 1$).

In some cases, it is beneficial to change the equilibrium point to some value other than $y/x = M_p$. To do so, suppose we change the product formula to $x^2y = k$.

a. The market will drive this modified Uniswap contract to hold x tokens of type A and y tokens of type B, where y/x is $c \cdot M_p$ for some constant c. What is c?

b. Under what conditions on M_p will liquidity providers make the most profit? That is, for what value of r is their profit maximized?

Problem 5. An insecure 3-party payment channel. Three parties, A, B, and C, are constantly making pairwise payments and thus design a 3-party payment channel based on the revocable hashed timelock contracts (HTLC) we saw in lecture 12. Once the channel is established, they can transact without ever touching the blockchain. The channel itself is a 3-out-of-3 multisig address that is bound to the public keys of A, B, and C. When B wants to pay A using the channel, the following happens without touching the blockchain:

- Party A receives from B a hashed timelocked transaction T_{A} that it can sign and post (the transaction is already signed by B and C). The transaction has three outputs:
  - one immediate output for B whose value is B's current balance in the channel,
  - one immediate output for C whose value is C's current balance in the channel, and
  - one output whose value is A's current balance in the channel, but with a hashed timelock spending rule: A can spend the output seven days after the transaction is mined, but either B or C can spend the output immediately if they have a hash preimage x initially known only to A.

  As in the two party payment channel, A can post this transaction to close the channel and collect her current balance after seven days (B and C can collect their balances immediately). However, if A wants to keep using the channel, then when she later pays B, she would first send the hash preimage x to B and C, thereby effectively invalidating the transaction T_{A}. Indeed, it would no longer make sense for A to post this transaction: if she did, then either B or C would immediately spend A's timelocked output, and A would lose her balance in the channel. This means that she can no longer close the channel in its old pre-payment state. A would be a given a new transaction T'_A that lets her close the channel in its new state, if she wants.

- Parties B and C each receive a similar transaction with three outputs representing the current balances in the channel. For example, party B's transaction has one output that is immediately available for A, one output that is immediately available for C, and one output that is hashed timelocked for B as above. C receives a similar transaction.

Let's show that while this general approach is secure for a two-party payment channel, it is completely insecure for three parties. In particular, two colluding parties can steal funds from the third. To see how, suppose the channel has a total of 100 BTC locked up. At some time in

the past 90 of the BTC belonged to A and 5 BTC belonged to B and C each. Currently 80 of the BTC belong to C and 10 BTC belong to A and B each. Show that A and B can collude to steal 75 BTC that currently belong to C, and split the loot between them.
Hint: think about what happens if A posts the transaction she had at the time when 90 of the BTC belonged to her.