CS251 Final Exam, Winter 2023
Tuesday, Dec. 12, 2023

Instructions:

- You may use your laptop during the exam as long as you do not access the Internet for anything other than overleaf. You may also use the Internet to download the exam and upload your answers to gradescope once you are done.
- You may not collaborate with others. Students are bound by the Stanford honor code.
- To submit your answers please either (i) use the provided LaTeX template, or (ii) write your answers on the printed exam, or (iii) write your answers on blank sheets of paper, but please make sure to start each question on a new page. When done, please upload your solutions to Gradescope (**7DVJKY**).
- The LaTeX template is at `https://cs251.stanford.edu/final-73pbntq39.zip` Please do not share the link with others.

Your Name: _____

SUNet ID: _____ `@stanford.edu`

- The exam has 5 questions totaling 100 points.
- You have two and a half hours to complete them.
- Please keep your answers concise.

1	/24
2	/18
3	/20
4	/18
5	/20
Total	/100

Problem 1. [24 points]: Questions from all over.

A) Suppose Alice sends to an Ethereum block builder a transaction that pays Bob 2 ETH from her externally owned account (EOA). The transaction is posted on chain in the next block. What prevents Bob for replaying the transaction and collecting two more ETH from Alice?

B) Alice has a secret key that lets her spend UTXO_0 on the Bitcoin network. She creates two transactions: Tx_1 and Tx_2. The input to Tx_1 is UTXO_0 and the output is UTXO_{bob} that Bob can spend. The input to Tx_2 is also UTXO_0 and the output is UTXO_{carol} that Carol can spend. Alice sends Tx_1 to one miner and sends Tx_2 to a different miner. Will both transactions eventually be posted on the Bitcoin chain? Justify your answer.

C) Suppose Alice owns some USDC on Ethereum. She wants to issue a transaction to some dApp, such as Uniswap, that is deployed on Ethereum. Can she directly use her USDC balance to pay for the gas needed to post the transaction?

D) Consider a transaction T posted in an Ethereum block where the To address is the address of some dApp. The transaction contains CALLDATA which encodes the arguments for the function F being called. Which of the following can read the CALLDATA contained in T?

(i) The code of the function F executed as a result of T.

(ii) The code of the function F executed as a result of a subsequent transaction to T.

(iii) A human looking at the chain after the block containing T is finalized.

E) In a privacy pool, such as Tornado, users can only deposit a fixed amount per transaction (say 1000 USDC), and later withdraw that exact amount. What would go wrong if users could deposit whatever amount they want and later withdraw the same amount they deposited?

F) Explain what is the purpose of the watchtower in a bi-directional payment channel. In your answer, you may focus on a simple Solidity-based payment channel.

Problem 2. [18 points]: Ethereum consensus.

A) In a PBFT-style consensus among n nodes, suppose we want to ensure safety assuming fewer than 2/5 of the nodes are malicious. What should the block finality rule be to ensure safety and the maximal possible liveness? Specifically, how many nodes must sign a block for it to become finalized?

B) Using your block finalization rule from part (A), how many nodes must be malicious in order to harm liveness?

C) Suppose Alice becomes an Ethereum validator by installing the Ethereum node software and transferring 32 ETH from her own account to the Ethereum staking deposit contract. Alice registers a validation signing key that she will use to sign blocks. Recall that once two-thirds of all validators sign a block, the block becomes finalized. [Ethereum validators actually sign epochs, which are a sequence of 32 blocks, but that detail is not relevant here.]

An attacker steals Alice's validation signing key. Explain what harm the attacker can cause to Alice.

D) Continuing with part (C), will the attacker's behavior that you described harm the safety of Ethereum consensus, assuming the attacker only has Alice's validation key?

E) Suppose 40% of the n Ethereum validators stop signing blocks. This will prevent future blocks from being finalized because signatures from the remaining 60% of validators are not enough to finalize a block. Explain how Ethereum consensus recovers from this. Recall that once a validator stops signing blocks, it will slowly lose its 32 ETH stake. Once its stake is exhausted, the validator can no longer be slashed and it will be ejected from the validator set. How many nodes (as a function of n) need to be ejected from the validator set so that blocks can be once again finalized?

Problem 3. [20 points]: Solidity Bugs – Wrapped ETH.

Accounts in Ethereum have a balance in ETH, but ETH cannot be used directly with DeFi services. These services only accept assets stored in ERC-20 contracts. To solve this mismatch, there is a simple Wrapped ETH (WETH) ERC-20 contract, located at a specific address, that enables anyone to convert (wrap) their ETH to WETH. The resulting WETH can be used in DeFi. Of course, anyone who wraps their ETH to WETH can, at any time, withdraw (unwrap) their WETH and get their ETH back. The following code is the implementation of the WETH contract, where we introduced three bugs.

```
contract WETH {
  mapping (address => uint)                       public  balanceOf;
  mapping (address => mapping (address => uint))  public  allowance;

  function deposit() public payable {
       balanceOf[msg.sender] += msg.value;
  }
  function withdraw(int wad) public {
       require(balanceOf[msg.sender] >= wad);
       balanceOf[msg.sender] -= wad;
       msg.sender.transfer(wad);
  }
  function totalSupply() public view returns (uint) {
       return this.balance;
  }
  function approve(address guy, uint wad) public {
       allowance[msg.sender][guy] = wad;
  }
  function transfer(address dst, uint wad) public payable {
       transferFrom(msg.sender, dst, wad);
  }
  function transferFrom(address src, address dst, uint wad) public {
       require(balanceOf[src] >= wad);
       if (src != msg.sender) {
           require(allowance[src][msg.sender] >= wad);
       }
       balanceOf[src] -= wad;
       balanceOf[dst] += wad;
  }
}
```

The current balance of this simple WETH contract is over three million ETH. That is, more than three million ETH have been wrapped as WETH.

A) Explain how Bob can abuse the implementation of the `withdraw` function to withdraw more than he deposited.

B) How would you fix the `withdraw` function? It is possible to fix the problem by adding only *one* character to the code of the function, but we will accept other fixes.

C) There is a serious bug in the `transferFrom` function. Suppose Alice authorizes Bob to transfer 1 ETH from her account. Explain how Bob can transfer much more than 1 ETH from Alice's account.

D) How would you fix the `transferFrom` function? There is a fix that requires adding only one line to the function.

E) Suppose both these bugs are fixed. Is it always true that the ETH balance of the WETH contract is equal to the sum of all the entries in `balanceOf[]`? That is, will the `totalSupply` function always return the correct total supply?

Problem 4. [18 points]: Rollups.

A) Briefly explain why a Rollup system writes the transaction data of all Rollup transactions, in the clear, to the Rollup L1 contract.

B) Every Ethereum transaction includes a 65-byte signature by the sender. In an optimistic Rollup, when L2 transaction data is sent to the L1 Rollup contract, do the transaction signatures need to be included in the transaction data? Please explain.

C) Same question as in part (B), but now applied to a proof-based Rollup (a.k.a a validity Rollup or zk-Rollup). Do the transaction signatures need to be included in the transaction data?

D) In a Rollup (either optimistic or validity) can a centralized coordinator prevent Alice from getting her funds out of her account on the L2 and back into her account on the L1?

E) In an optimistic Rollup, suppose that the coordinator pushes an invalid block of transactions to the L1 Rollup contract. Then there is a seven day period where anyone can contest that transition using a fault proof submitted to the L1 Rollup contract. If the fault proof is successfully submitted then the invalid block is ignored. To prevent this from happening, the coordinator could submit the invalid block during a week where there is a significant load on the L1 chain (for some unrelated reason) so that it is difficult to get transactions accepted to the L1. This will make it difficult for anyone to submit a successful fault proof within the seven day window. Can you propose a mechanism to mitigate this issue?

Problem 5. [20 points]: Stablecoins and MEV.

In class we explained how a custodial stablecoin, such as USDC, works. The custodian holds the treasury in a number of traditional banks, and issues tokens against the treasury. When all works well, every issued coin on chain is backed by a dollar held in a traditional bank.

A) Suppose the custodian holds 10% of the treasury in a traditional bank called Safe Value Bank (SVB). The remaining 90% are held in other banks. One day the Safe Value Bank fails and all the depositors of this bank lose their deposits. What will happen to the price of USDC in relation to USD? That is, what will the price of 1 USDC in USD converge to in the open market after this event? Please state any assumption you make about the behavior of the custodian following this event.

B) How can the custodian bring the exchange rate back to 1 USDC per USD?

C) Customers in Europe want to pay the custodian in Euro and receive back USDC. One (bad) option is for the custodian to hold part of its treasury in USD and part in Euro. That way, when new USDC coins are minted as a result of a Euro deposit, the deposit is simply added to the Euro pool held by the custodian.

Suppose that the custodian holds x USD and y Euro. Now, suppose that the price of the Euro drops relative to the USD. In particular, if previously 1 USD would buy z Euro, then now 1 USD buys $1.1 \cdot z$ Euro. What would happen to the price of USDC relative to USD? Please express the price of 1 USDC in USD in terms of x, y, z. You can assume that before the drop the price of 1 USDC was exactly 1 USD.

D) Clearly the custodian should not do what is described in part (C). How should the custodian support its European customers who want to deposit Euro in exchange for USDC?

E) Finally, a question about MEV. Say that Carol is the current block proposer. Recall that in MEV-boost, a relay sends a block header to Carol, and Carol signs the block header blindly, that is, without knowing what transactions are in the block. This is necessary to ensure that Carol cannot steal all the MEV for herself by creating a new signed block that pays all the MEV to Carol. Once Carol sends the signature on the block header to the relay, the relay posts the block — in the clear — to the Ethereum network. At this point Carol sees all the transactions in the clear. What is to prevent her from creating a new block where all the MEV is paid to Carol, sign that block, and post it to the Ethereum network? If enough other validators sign the block from Carol instead of the block from the relay, then Carol's block is the one that will be finalized and her MEV theft will be successful. What incentives encourage Carol not to do that?