CS251
Cryptocurrencies and Blockchain Technologies
Final Exam – Tuesday, December 10, 2019

OPEN BOOK, OPEN NOTES, OPEN LAPTOP, CLOSED WiFi

Your Name: _____

SUNet ID: _____ @stanford.edu

Check if you require expedited grading: ☐

In accordance with both the letter and the spirit of the Stanford Honor Code, I neither received nor provided any assistance on this exam.

Signature: _____

- The exam has 6 questions totaling 100 points.
- You have 150 minutes to complete them.
- Please only use the front side of every page to write your answers.
- Keep your answers concise.

1	/18
2	/16
3	/16
4	/18
5	/18
6	/14
Total	/100

1. [18 points]:

Questions from all over.

A) Why is the difficulty of the proof of work in Bitcoin set to ten minutes? What would go wrong if it was changed to ten seconds?

B) Does Nakamoto consensus assume a synchronous, partially synchronous, or asynchronous network model? Briefly explain why.

C) In class we discussed the Howey Test that defines what is a security. Are football game tickets a security? Consider that some people buy tickets for the purpose of scalping them later for profit.

D) What is the difference in the privacy models provided by Zcash vs. confidential transactions?

E) The Bitcoin blockchain contains many UTXOs that will never be spent, either because of an `OP_RETURN` script, or because the signing key needed to spend the UTXO has been lost. Nevertheless, they must be maintained as part of the active UTXO set. A frequent proposal is to modify Bitcoin so that UTXOs that are more than two years old will be automatically removed from the UTXO set. Is this a good idea? What would happen to transaction fees for a UTXO that is about to expire in one month?

F) All stablecoin systems maintain some 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 ether 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?

2. [16 points]:

Consensus protocols.

A) Suppose Alice buys a digital product (e.g., a digital book) from an online merchant and pays using Bitcoin. The merchant sends the product to Alice the minute it sees the payment transaction in a block in the Bitcoin network. Why does this strategy present a significant risk for the merchant?

B) A consensus protocol is said to be responsive if it offers instant finality on confirmed transactions. Your answer to the previous question shows that Nakamoto consensus is not responsive. Are there consensus protocols in the authenticated permissioned setting with a fixed set of n participants that are responsive? If not explain why not; if yes explain what assumptions are needed on the maximal number of faulty participants.

C) Proof of work. Recall that the mining difficulty in Bitcoin is updated roughly every two weeks so that the expected inter-block time remains at 10 minutes. The difficulty right after the genesis block is quite low, and it increases as more miners join the network. Suppose the blockchain currently contains n blocks. An attacker creates a blockchain of length $n + 100$, starting at the correct genesis block, but where all blocks have the same low difficulty as the genesis block. Because the difficulty is low in all blocks, creating such a long chain takes relatively little work. If miners follow the "mine the longeset chain" rule they would abandon the correct chain and mine this fake longer chain instead. Why is this not an attack on Bitcoin consensus?

3. [16 points]:

Solidity bugs. The following buggy Solidity contract implements a fund-raise for an independent film project by an unknown director. Anyone can send ether to the contract and receive tokens for their contribution, as implemented in the fallback function below. Once the film is finished, royalties from the film will be distributed to all participants based on token ownership. The project owner will fund the production of the film by withdrawing ether from the contract using the `withdraw` function below. The contract should ensure that every month the project owner can only withdraw 1 more ether than has ever been withdrawn.

```
contract MovieToken {
  address owner;                            // record owner address
  mapping (address => uint256) balances;  // investor balances
  uint256 totalSupply;                      // total supply in contract
  uint256 lastWithdrawDate;                 // last withdrawal time

  function MovieToken() {                   // Constructor
    owner = msg.sender;
    balances[owner] = msg.value;            // Owner may fund the movie
    totalSupply     = balances[owner];
  }

  function withdraw(uint256 amount) {     // note: function is not payable

    if (msg.sender != owner) { throw; }   // Only owner may withdraw funds
    if (amount == 0 || amount > this.balance) { throw; }
    if (now < lastWithdrawDate + (1 months))  { throw; } // Only once per month
    lastWithdrawDate = now;

    // Withdraw schedule: only withdraw 1 more ether than has ever
    // been withdrawn.
    uint256 maxAmount = (totalSupply - this.balance + (1 ether));
    if (amount > maxAmount) { throw; }

    if (!owner.send(amount)) { throw; }  // send the funds to owner
  }

  // Fallback function: record a fund-raise contribution
  function () payable {
    balances[msg.sender] += msg.value;   // transfer tokens to investor
    totalSupply += msg.value;            // record amount
  }
}
```

A) What is the maximum amount that can be withdrawn in each of the first four months, assuming the maximum amount is withdrawn every month?

month 1: ____ , month 2: ____ , month 3: ____ , month 4: ____ .

B) Describe an attack that lets the project owner withdraw all the funds sent to the project immediately on the first month.
Hint: if someone sends funds to a contract address *before* the contract is created, then upon creation of a contract at that address, the variable `this.balance` is equal to the amount of funds sent prior to creation. Note also that all variables in the `MovieToken` contract are *unsigned* integers, so that -1 is actually $2^{256} - 1$.

C) When a contract, say contract X, executes a `selfdestruct` with contract Y's address as an argument, this kills contract X and adds X's balance to Y, without triggering any code execution at Y. With this in mind, suppose we try to prevent your attack from part (B) by changing the second line in the constructor to:

```
balances[owner] = this.balance + msg.value;
```

Is there an attack that lets the owner withdraw all the funds on the first month?

D) How should the contract be fixed so that it correctly implements the intended withdrawal schedule, without changing the program logic?

4. [18 points]:

Recall that a Rollup server can be used to scale payment transactions in the Ethereum network. Rollup provides no privacy for the payer or payee: all transactions are recorded in the clear on the blockchain.

A) Suppose Rollup used a zkSNARK and did not record transactions on the blockchain. Would this provide the same level of privacy as Zcash? Justify your answer.

B) Optimistic Rollup is a variant of Rollup that eliminates the need for a SNARK. The Rollup server simply posts an updated Merkle root to the blockchain, without a SNARK. The update is signed by the Rollup server. Clearly this design is insecure. Please explain what could go wrong if it were deployed as is.

C) Optimistic Rollup prevents the attack from part (B) by giving anyone a week to contest any funds transfer. If a contest is submitted to the blockchain for a particular funds transfer, then the Rollup server has one day to present a valid transaction to justify the transfer. If it does, the blockchain verifies this one transaction, and if it is valid, the contest is dropped. Otherwise, the Rollup server is fined and the Merkle root is reverted to its state before the contested update. If no one contests a Merkle root update within one week, then the update is finalized, as are all payments that are part of this update. (this is a greatly over-simplified description of optimistic Rollup.) Questions on the next page.

- What are the implications of this design for an online merchant who wants to sell a digital product to a customer Alice? When can the merchant send the product to Alice? Explain your answer.

- What are the implications of this design for a user Alice who maintains her account balance in this Rollup system? What must she do to ensure the security of her funds?

D) Scaling by sharding. Consider a permissioned blockchain with a fixed set of n validator nodes. One proposal for scaling the blockchain is to split it into n independent blockchains, assigning one validator to each independent blockchain. Now, every DAPP randomly selects a blockchain to run on, thereby increasing throughput by a factor of n. What are two problems with this design?

5. [18 points]:

SNARKs. Let $\mathbb{F} = \{0, 1, 2, \ldots, p-1\}$ be a prime finite field.

A) Let $f(X) = c_0 + c_1 X + \ldots + c_4 X^4$ be a polynomial of degree 4 in $\mathbb{F}[X]$. Let $C_f(x)$ be a circuit that returns 0 if and only if $f(x) = 0$ (i.e. this circuit checks that x is a root of f). Show the R1CS program for this circuit by writing out the 4×5 matrices A, B, C that implement C_f. These matrices operate on a column vector $\boldsymbol{z} := (1, x, w_1, w_2, w_3) \in \mathbb{F}^5$. You must ensure that there exist $w_1, w_2, w_3 \in \mathbb{F}$ for which $(A\boldsymbol{z}) \circ (B\boldsymbol{z}) = C\boldsymbol{z}$ if and only if $f(x) = 0$.

B) Let $G = (V, E)$ be a graph on n-vertices. A 3-coloring of G is a vector $(w_1, \ldots, w_n) \in \{0, 1, 2\}^n$ so that for all edges $(u, v) \in E$ we have $w_u \neq w_v$ (i.e., no two adjacent vertices in the graph are assigned the same color). Finding a 3-coloring of a graph is an NP-hard problem.

Let f be the degree-4 polynomial that satisfies $f(2) = f(1) = f(-1) = f(-2) = 0$ and $f(0) = 1$. Show that the following set of algebraic constraints on $(w_1, \ldots, w_n)$ is satisfied if and only if $(w_1, \ldots, w_n) \in \mathbb{F}^n$ is a 3-coloring of G: (assume $|\mathbb{F}| \gg |E|$)

$$\begin{cases} \displaystyle\sum_{(u,v)\in E} f(w_u - w_v) = 0, \\ w_i(w_i - 1)(w_i - 2) = 0 \quad \text{for all } i \in \{1, \ldots, n\}. \end{cases} \tag{1}$$

C) How many constraints (rows) are there in the R1CS program that implements (1)? The program operates on the column vector $(1, w_1, \ldots, w_n, w_{n+1}, \ldots, w_{n+k})$ where the entries $w_{n+1}, \ldots, w_{n+k} \in \mathbb{F}$ are for you to define. Give your answer as a function of $n = |V|$ and $m = |E|$. You may use big-O notation. Justify your answer.

D) Suppose we use the R1CS program from part (C) to implement a trusted-setup zkSNARK to prove that a graph is 3-colorable in zero-knowledge. How big are the following quantities:

- The size of the proof: _____
- The running time of the verifier: _____
- The running time of the prover: _____
- The size of the prover parameters (called S_{P} in lecture): _____

Express your answers in terms of $n = |V|$ and $m = |E|$, and using big-O notation. Assume that the linear-only encodings are constant size, and that each linear-only encoding operation takes constant time.

6. [14 points]:

HD wallets. Bitcoin will soon transition from ECDSA signatures to Schnorr signatures. A schnorr secret key is a 256-bit integer α and the corresponding public key is $h := g^{\alpha}$, where g is some fixed public base. A signature on a message m is a pair of integers $\sigma := (c, z)$. A signature σ on m is valid under public key h if the following equality holds: $H\big(m, (g^z/h^c)\big) = c$, where H is a cryptographic hash function such as SHA256.

A) Recall that an HD wallet, as described in the lecture, maintains a spending key (k_1, k_2) and an address generation key (h, k_2), where $h := g^{k_1}$. Address number i is the hash of the public key h_i where $h_i := h \cdot g^{\text{HMAC}(k_2, i)}$. What is the purpose of having a spending key and a separate address generation key?

B) Let's see that Schnorr signatures can be insecure in this settings. Suppose an attacker steals the address generation key (h, k_2). Moreover, the attacker finds on the blockchain a valid Schnorr signature (c, z) on a message m under public key h_3, where h_3 is defined as in part (A) using $i = 3$. Show that the attacker can now derive a signature (c, z') on the same message m, but under the public key h_4. Explain how to construct z'. (this violates security because the address generation key (h, k_2) should not enable message signing.)

.

C) Can this be used to steal funds from an HD wallet? Say, if an attacker observes a transaction that spends a UTXO belonging to address h_3, can the attacker then spend a UTXO belonging to address h_4?
Hint: recall that a transaction signature in Bitcoin (typically) signs the transaction data, which includes the public key of the UTXO being spent.

.