CS251: Cryptocurrencies and Blockchain Technologies **Fall 2019**

Assignment #1

Due: 11:59pm on Mon., **Oct. 7, 2019**
Submit via Gradescope (each answer on a separate page) code: **MG7EP3**

Problem 1. Hash functions and proofs of work. In class we defined two security properties for a hash function, one called collision resistance and the other called proof-of-work security. Show that a collision-resistant hash function may not be proof-of-work secure.

Hint: let $H : X \times Y \to \{0, 1, \ldots, 2^n - 1\}$ be a collision-resistant hash function. Construct a new hash function $H' : X \times Y \to \{0, 1, \ldots, 2^m - 1\}$ (where m may be greater than n) that is also collision resistant, but for a fixed difficulty D (say, $D = 2^{32}$) is not proof-of-work secure with difficulty D. That is, for every puzzle $x \in X$ it should be trivial to find a solution $y \in Y$ such that $H'(x, y) < 2^m/D$. This is despite H' being collision resistant. Remember to explain why your H' is collision resistant, that is, explain why a collision on H' would yield a collision on H.

Problem 2. Beyond binary Merkle trees. Alice can use a binary Merkle tree to commit to a tuple of elements $S = (T_1, \ldots, T_n)$ so that later she can prove to Bob that some T_i is in S using a an inclusion proof containing at most $\lceil \log_2 n \rceil$ hash values. The binding commitment to S is a single hash value. In this question your goal is to explain how to do the same using a k-ary tree, that is, where every non-leaf node has up to k children. The hash value for every non-leaf node is computed as the hash of the concatenation of the values of all its children.

- **a.** Suppose $S = (T_1, \ldots, T_9)$. Explain how Alice computes a commitment to S using a ternary Merkle tree (i.e., $k = 3$). How does Alice later prove to Bob that T_4 is in S?
- **b.** Suppose S contains n elements. What is the length of the proof that proves that some T_i is in S, as a function of n and k?
- **c.** For large n, if we want to minimize the proof size, is it better to use a binary or a ternary tree? Why?

Problem 3. Bitcoin script. Alice is on a backpacking trip and is worried about her devices containing private keys getting stolen. She wants to store her bitcoins in such a way that they can be redeemed via knowledge of a password. Accordingly, she stores them in the following `ScriptPubKey` address:

```
OP_SHA256
<0xeb271cbcc2340d0b0e6212903e29f22e578ff69b>
OP_EQUAL
```

- **a.** Write a `ScriptSig` script that will successfully redeem this transaction given the password. **Hint:** it should only be one line long.
- **b.** Suppose Alice chooses an eight character password. Explain why her bitcoins can be stolen soon after her UTXOs are posted to the blockchain. You may assume that computing SHA256 of all eight character passwords can be done in reasonable time.

c. Suppose Alice chooses a strong 20 character passphrase. Is the `ScriptPubKey` above a secure way to protect her bitcoins? Why or why not?
Hint: reason through what happens when she tries to redeem her bitcoins.

Problem 4. BitcoinLotto: Suppose the nation of Bitcoinia decides to convert its national lottery to use Bitcoin. A trusted scratch-off ticket printing factory exists and will not keep records of any values printed. Bitcoinia proposes a simple design: a weekly run of tickets is printed with an address holding the jackpot on each ticket. This allows everybody to verify that the jackpot exists. The winning ticket contains the correct private key under the scratch material.

a. If the winner finds the ticket on Monday and immediately claims the jackpot, this will be bad for sales because players will all realize the lottery has been won. Modify your design to prevent this (of course, you cant prevent the winner from proving ownership of the correct private key outside of Bitcoin).

b. Some tickets inevitably get lost or destroyed. Modify the design to roll forward any unclaimed jackpot from Week n to the winner in Week $n + 1$. If the Week $n + 1$ jackpot is unclaimed, then the jackpot from both week n and $n+1$ should roll forward to the winner of Week $n+2$, and so on. Can you propose a design that works, without enabling the lottery administrators to embezzle funds?

Problem 5. Lightweight clients: Suppose Bob runs an ultra lightweight Bitcoin client which receives the current head of the block chain from a trusted source. This client has very limited memory and so it only permanently stores the most recent block chain header (deleting any previous headers).

a. If Alice wants to send a payment to Bob, what information should she include to prove that her payment to Bob has been included in the block chain?

b. Assume Alices payment was included in a block k blocks before the current head and there are n transactions per block. Estimate how many bytes this proof will require in terms of n and k and compute the proof size for $k = 6$ and $n = 1024$.

c. One proposal is to add an extra field in each block header that contains the hash of a certain previous block header. Specifically, block header number h contains the hash of block header number $h - 1$ and the hash of block header number $m < h$, where m is computed according to the following rule: Let 2^i be the largest power of 2 dividing h, then $m = h - 2^i$. Explain how this can be used to reduce the proof size from part (b). What is the worst case size of a proof in terms of n and k?