CS 251: Bitcoin and Crypto Currencies Fall 2015

Homework 1

due: October 5, 11:59pm via email to cs251.stanford.fall.2015@gmail.com

1. **Ternary Merkle trees.** In lecture we saw how Alice can use a binary Merkle tree to commit to a set 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 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 *ternary* tree, that is, where every non-leaf node has up to three children. The hash value for every non-leaf node is computed as the hash of the concatenation of the values of its children.
    a. Suppose $S = \{T_1, \ldots, T_9\}$. Explain how Alice computes a commitment to S using a ternary Merkle tree. Explain how Alice later proves 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?
    c. For large n, is the proof using a ternary tree shorter or longer than the proof using a binary Merkle tree? Please explain.

2. **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*. In this question we explore the relation between these two properties. Show that a collision-resistant hash function may not be proof-of-work secure.
Hint: Let $H:P \times S \rightarrow \{0,1,\ldots,2^n-1\}$ be a collision-resistant hash function. Construct a new hash function $H':P \times S \rightarrow \{0,1,\ldots,2^{n'}-1\}$ (where n' 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 $p \in P$ it should be trivial to find a solution $s \in S$ such that $H'(p,s) < 2^{n'}/d$. This is despite H' being collision resistant. Remember to explain why H' is collision resistant, that is, explain why a collision on H' gives a collision on H.

3. **Mining rates**: Recall that the difficulty of finding a block continually adjusts so that 10 minutes is the mean time to find a block. Assuming that the total hash power of the Bitcoin network is constant:
    a. What is the probability that at least one block will be found in the next 10 minutes? Similarly, for k>0, what is the probability that at least one block will be found in the next *k* minutes?
    b. What is the probability that at least 6 blocks will be found in the next 60 minutes? Similarly, for k>0, what is the probability that at least *k* blocks will be found in the next $10 \cdot k$ minutes?
    c. Suppose a Bitcoin merchant wants to have a policy guaranteeing that orders will ship within *x* minutes after receipt of payment, but also wants to wait until a transaction is confirmed by at least 6 blocks to reduce the risk of forks. What value of *x* will ensure that with 99% confidence at least 6 blocks will be found within *x* minutes? For this *x*, what is the probability that 2 or fewer blocks will be found within *x* minutes?

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

```
OP_SHA1
<0x084a3501edef6845f2f1e4198ec3a2b81cf5c6bc>
OP_EQUAL
```

   a. Write a ScriptSig script that will successfully redeem this transaction. [Hint: it should only be one line long.]
   b. Explain why this is not a secure way to protect Bitcoins using a password.
   c. Would implementing this using Pay-to-script-hash (P2SH) fix the security issue(s) you identified? Why or why not?

5. **Lightweight clients**: Suppose Bob runs an ultra lightweight 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 Alice's 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 $k,n=6,2^{10}$.
   c. One proposal is to add an extra field in each block header pointing to the last block which has a *smaller* hash value than the current block. Explain how this can be used to reduce the proof size from part (b). What is the expected size of a proof (in bytes) now in terms of *n* and *k*? To simplify your analysis, you may use asymptotic (Big O) notation. What are the best-case and worst-case sizes?

6. **BitcoinLotto:** Suppose the nation of Bitcoinia has decided 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 can't prevent the winner from proving ownership of the correct private key outside of Bitcoin).
   b. Some tickets inevitably get lost or destroyed. So you'd like to modify the design to roll forward any unclaimed jackpot from Week *n* to the winner in Week *n+1*. Can you propose a design that works, without enabling the lottery administrators embezzle funds? Also, you want to make sure that the Week *n* winner can't wait until the beginning of Week *n+1* in an attempt to double their winnings.