CS 251: Bitcoin and Cryptocurrencies Fall 2016

Homework 1

due: 2016-10-10 23:59 via Gradescope (entry code `M4YJ69`)

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*. 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 $H:P \times S \rightarrow \{0,1,...,2^n-1\}$ be a collision-resistant hash function. Construct a new hash function $H':P \times S \rightarrow \{0,1,...,2^{n'}-1\}$ (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, why a collision on H' would yield a collision on H.

2. **Beyond binary Merkle trees:** Alice can use a binary Merkle tree to commit to a set of elements $S = \{T_1, ..., T_n\}$ 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 *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 its children.
   a. Suppose $S = \{T_1, ..., T_9\}$. Explain how Alice computes a commitment to S using a ternary Merkle tree (i.e. k=3). How can 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*, what is the proof size overhead of a *k*-ary tree compared to a binary tree? Can you think of any advantage to using a k>2? (Hint: consider computation cost)

3. **Hiding vs. binding commitments**: Bob is launching a new secure messaging app, BobCrypt. When Alice installs the app, it creates an account for her on the BobCrypt server using a hash of her phone number. The app then queries the server by sending the hash of each phone number in Alice's address book to learn which of Alice's friends already have BobCrypt accounts. The goal is that users can discover their friends' accounts without the server learning the contents of every user's address books.
   a. Explain why this scheme does not achieve the intended security goal. How can Bob act maliciously to determine the phone numbers and contacts of all BobCrypt users?
   b. After you tell Bob that a simple hash is a binding commitment, but does not hide the committed value, he decides to use a common construction for hiding commitments in BobCrypt 2.0. The app now uploads H(phone number, *r*) where *r* is a random 128-bit nonce chosen by the app. Explain to Bob why it is not possible to provide the intended functionality using this approach.

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
<0xeb271cbcc2340d0b0e6212903e29f22e578ff69b>
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*=8, *n*=256.
   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.