CS251: Cryptocurrencies and Blockchain Technologies **Fall 2021**

Assignment #1

2022年2月6日 周日晚23:59前发到群内指定邮箱提交Assignment

Problem 1. A broken proof of work hash function. In class we discussed using a hash function $H : X \times Y \to \{0, 1, \ldots, 2^n - 1\}$ for a proof of work scheme. Once an $x \in X$ and a difficulty level D are published, it should take an expected D evaluations of the hash function to find a $y \in Y$ such that $H(x, y) < 2^n/D$. Suppose that $X = Y = \{0, 1\}^m$ for some m (say $m = 512$), and consider the hash function

$$H : X \times Y \to \{0, 1, \ldots, 2^{256} - 1\} \qquad \text{defined as } H(x, y) := \text{SHA256}(x \oplus y).$$

Here $\oplus$ denotes a bit-wise xor. Show that this H is insecure as a proof of work hash. In particular, suppose D is fixed ahead of time. Show that a clever attacker can find a solution $y \in Y$ with minimal effort once $x \in X$ is published. Hint: the attacker will do most of the work before x is published.

Problem 2. Beyond binary Merkle trees. Alice can use a binary Merkle tree to commit to a list of elements $S = (T_1, \ldots, T_n)$ so that later she can prove to Bob that $S[i] = T_i$ using 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? What values are provided in the proof?

b. Suppose S contains n elements. What is the length of the proof that proves that $S[i] = T_i$, 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 P. Accordingly, she stores them in the following `ScriptPubKey` address:

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

a. Write a `ScriptSig` script that will successfully redeem this UTXO given the password P. **Hint:** it should only be one line long.

b. Suppose Alice chooses a six character password P. Explain why her bitcoins can be stolen soon after her UTXO is posted to the blockchain. You may assume that computing SHA256 of all six character passwords can be done in reasonable time.

c. Suppose Alice chooses a strong 30 character passphrase P. 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 public address is published that holds the jackpot. This allows everyone to verify that the jackpot exists, namely there is a UTXO bound to that address that holds the jackpot amount. Then a weekly run of tickets is printed so that the winning lottery 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 the design to use one of the locktime opcodes to ensure that the prize can only be claimed at the end of the week. (of course, you cannot prevent the winner from proving ownership of the correct private key outside of Bitcoin). The Bitcoin script opcodes are listed here: `https://en.bitcoin.it/wiki/Script`. Alternatively, explain how to use a 2-out-of-2 multisig for this.

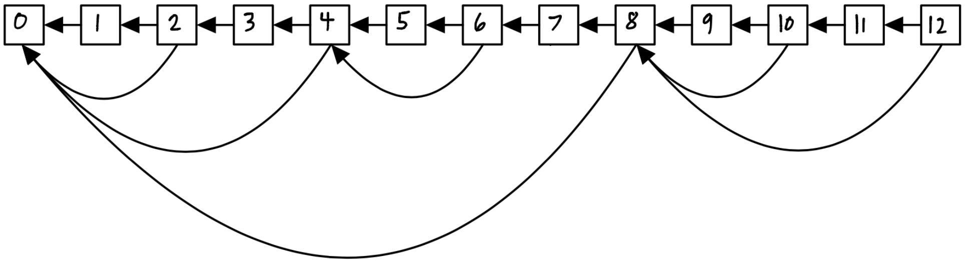
b. Some tickets inevitably get lost or destroyed. Modify the design to roll forward so that any unclaimed jackpot from Week n can be claimed by the winner in Week $n+1$. If the Week $n+1$ jackpot is unclaimed, then the jackpot from both weeks n and $n+1$ can be claimed by the winner of Week $n+2$, and so on. Can you propose a design that works without introducing new ways for the lottery administrators to embezzle funds beyond what is possible on the basic scheme described at the beginning of the question? You may assume that the lottery system runs for 1000 weeks, and then shuts down permanently. Hint: use multisig.

Problem 5. Lightweight clients: Suppose Bob runs an ultra lightweight Bitcoin client which receives the current head of the blockchain from a trusted source. This client has very limited memory and so it only stores the block header of the most recent block in the chain (the head of the chain), deleting any previous block 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 exactly n transactions per block. Estimate how many bytes this proof will require in terms of n and k. Compute the proof size for $k = 6$ and $n = 1024$.

c. One proposal is to add an extra field in every 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 $m = h - 2^i$. See the figure below for an example. 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?


0
1
2
3
4
5
6
7
8
9
10
11
12