CS 251: Bitcoin and Crypto Currencies Fall 2015

Homework 2

due: October 28, 11:59pm via email to cs251.stanford.fall.2015@gmail.com

Question 1: (block propagation time) Let's assume a simple model for how quickly Bitcoin blocks propagate through the network: after t seconds, a group of miners controlling a proportion $\alpha(t)$ of the mining power has heard about the transaction, up to some point t_{max} after which all miners will have heard about it. That is, $\alpha(t)=1$ for all $t \geq t_{max}$. Further assume that $\alpha(0)=0$ and $\alpha(t)$ is monotonically increasing in t.

A. Assuming that a block is found every b seconds, on average, with a Poisson distribution as we have assumed, provide a general formula for the probability that at least one stale block will be found. That is, what is the probability that after a valid block B is found, at least one other block B' will be found by a miner who hasn't heard about B yet?

B. As a simple example, consider $\alpha(t)=t^2/3600$, that is, a quadratically increasing proportion of the mining power hears about a new block up until 60 seconds, at which point all have heard. Assuming the normal block rate of b=600, what is the probability of at least one stale block being found?

C. If we lowered b to 60 seconds to make transactions post faster, how would this affect your answer from part (B)? What problems might this cause?

D. One could argue that the increased rate of stale blocks you identified in part (C) isn't really a problem as miners will still be paid at the same rate. Explain why this argument may not hold up in practice. In particular explain why our simple model of $\alpha(t)$ is not very realistic.

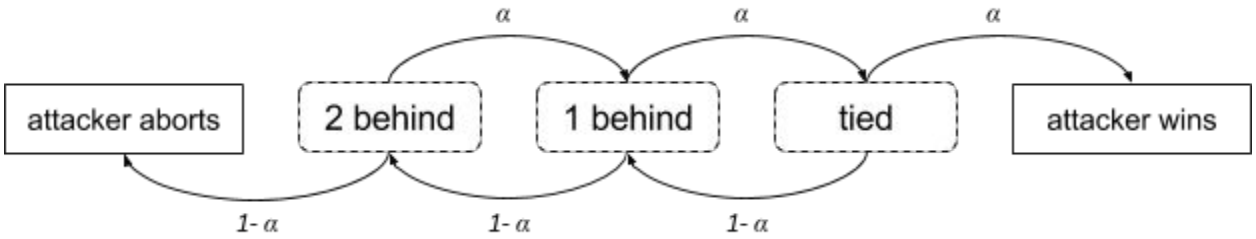
Question 2: (multi-judge escrow service) In class we saw how to use 2-out-of-3 multisig to build an escrow service where Alice can buy a product from Bob and, if all goes well, Alice gets the product and Bob gets paid. Otherwise a judge can adjudicate the dispute. One issue with that protocol is that the judge may demand a service fee and the participants, Alice and Bob, have no choice but to pay.

In this question your goal is to design an escrow system where, ahead of time, Alice and Bob agree on the set of three judges so that during adjudication they can choose any one of the three to adjudicate. We are assuming that the three judges are honest and consistent, that is, all three will always rule the same way.

A. Show how to implement this three-judge escrow system using a single standard multisig transaction to a multisig address that Alice and Bob agree on ahead of time. Your design must ensure that even if the three judges collude, they cannot steal the funds that Alice sends to Bob. Recall that if all goes well then the parties need not involve the judges. If something goes wrong, then any one of the three judges can adjudicate.
Hint: You will need to use more than 5 keys in the multisig transaction. Alice and Bob will use more than one key each.

B. Your solution from part (A) uses a standard multisig transaction, but the redeem script must list more than five public keys. Write a shorter (non-standard) script to do the same as in part (A) where each party is only assigned one key, so that only five keys are listed in the redeem script.
Hint: you will need to use logical operations (e.g. OP_IF, OP_OR etc.)

Question 3: (feather forking) In class we learned about *feather forking*, a strategy by which a minority coalition of miners can attempt to censor transactions from the blockchain. Suppose a group of miners which controls a fraction α of the total mining power announces "if we see a block containing a transaction from our blacklist B, we will attempt to fork until we are 3 blocks behind the main chain." This strategy can be seen in the following probabilistic state machine:

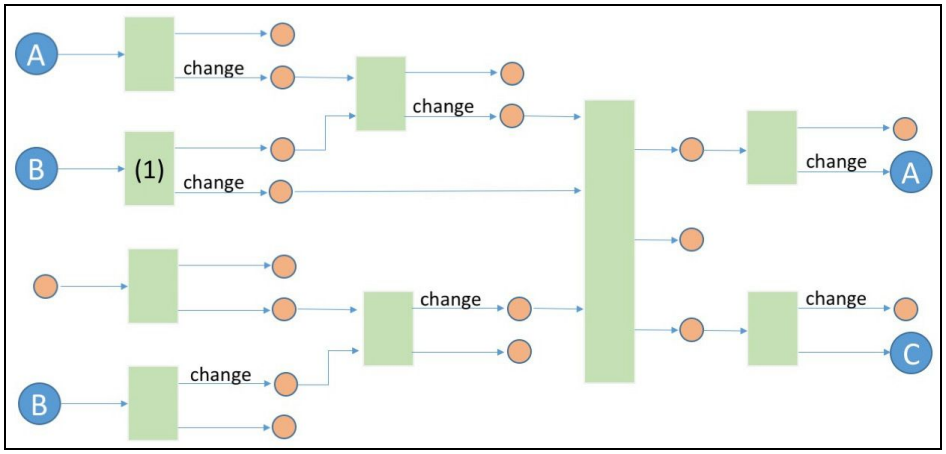


A. What is the probability that the censors' attack will succeed in terms of α?
   Hint: Express the probability of the fork succeeding from each active state in the state machine above as p_0, p_1, and p_2. You can now express each probability in terms of the others and solve a system of three equations for p_1, the probability of the attack succeeding from the start state, in terms of α.
B. On expectation, for how many blocks will the system be in a forked state after the attack is launched before it either succeeds or fails?
   Hint: You can solve this problem using a similar system of equations as above, noting that every time a transition is taken the fork lasts one additional block. Make sure that, as a sanity check, the attack is expected to resolve in 2 blocks for both $\alpha \to 0$ and $\alpha \to 1$.

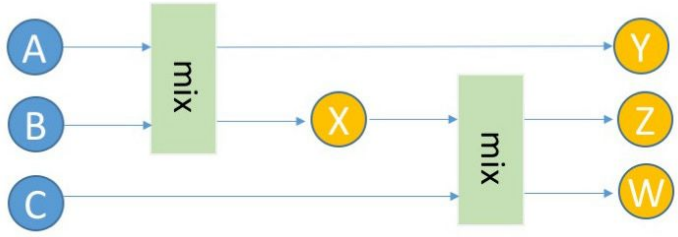
Question 4: (idioms of use) Consider the transaction graph in the figure below: rectangles represent transactions, empty circles represent fresh addresses, and filled in circles represent addresses controlled by the named entity (i.e., 'A' stands for Alice, 'B' stands for Bob, and 'C' stands for Carol). An edge labeled "change" means that the end node is the change address for that transaction, as identified by the heuristic discussed in class. Note that not every transaction has an identified change address.

A. Can an observer identify who was paid by Bob in the transaction marked (1) ?
B. Can an observer identity who paid Carol?

Explain how you identified the parties in question or why they could not be identified with certainty.



Question 5: (building a large mix from smaller ones) Suppose Bitcoin was changed such that transactions can have at most two inputs and two outputs. Alice, Bob, and Carol each have a single UTXO worth one 1 BTC. They want to mix their coins among the three of them (without a trusted server) so that three output UTXOs are worth 1 BTC each and they are perfectly shuffled: all six (3!) permutations are equally likely. They decide to do the following: Alice and Bob post a transaction with their two bitcoins as input and two fresh addresses, X and Y, as outputs, each holding 1 BTC. Alice controls one address and Bob controls the other, but an observer cannot tell which is which. Suppose X is listed as the first output in this transaction. Next, Carol and the owner of X post a similar transaction: Carol's coin and the coin X are the inputs, and two fresh addresses W and Z are the outputs, each holding 1 BTC. Carol controls one address and the owner of X controls the other. As before, an observer cannot tell which is which. They use Y,Z,W as the final shuffled coins. Pictorially, the mixing process looks as follows:



A. What are the anonymity sets for the addresses Y, Z, and W?
   Assuming that each mix independently assigns equal probability to both outcomes, what is the probability that Alice controls address W? What is the probability that Carol controls W? Because the probabilities are not ⅓, an observer has better than 1:3 odds in guessing who controls W.
B. Suppose at a later time it is revealed that Bob controls W. What else is revealed by this?
C. Your answers to the previous parts show that this mix is inadequate: the six possible permutations are not equally likely. By adding one transaction to the mix, can you help Alice, Bob, and Carol design a better mix so that all six (3!) permutations are equally likely to an outside observer?
   Hint: adding one more mixing transaction is sufficient. Show that the three mixes can function independently of one another, but one or more of the mixes must choose the ordering of outputs non-uniformly, namely with one-third probability one way and two-thirds the other way.

   Discussion: It can be shown that a mix of size two is sufficient to build a mix of size n, for any $n>2$, using only $O(n \log n)$ mixes of size 2. In other words, Coinjoin can be made to work even if the number of inputs per transaction is limited to two.

   However, note that if each mix is uniform (assigning an equal probability to both outcomes) and operates independently of the other mixes, then it is impossible to build a perfectly uniform mix for n users with any number of mixes. To see why, consider the number of possible outcomes for m uniform and independent mixes and compare with $n!$, the number of possible permutations of n users.