CS 251: Bitcoin and Cryptocurrencies Fall 2016

Homework 2

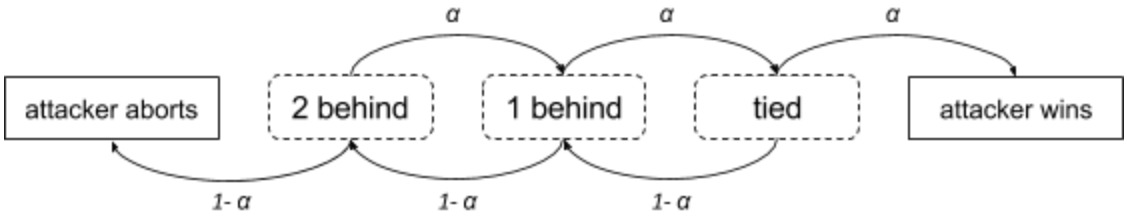
due: 2016-11-02 23:59 via Gradescope (entry code `M4YJ69`)

1. **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 a 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 script must list more than five public keys. Write a script to achieve the same thing where each party is only assigned one key (only five keys total are listed in the redeem script)
    Hint: you'll need to use logical operations (e.g. OP_IF, OP_OR) in a non-standard script

2. **Network propagation**: 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. Assume that blocks are found in Poisson process with a mean time to find a block of $\beta=600$ seconds ($\lambda=1/600$). A *stale block* (likely to become an *orphan block*) occurs if some miner finds a block before news of the most recent block reaches them.
    a. Given the design of the Bitcoin P2P network, explain why an exponential propagation model (i.e. $\alpha(t) \propto b^t$ for some b) is a plausible model.
    Hint: Recall that the derivative of an exponential is another exponential function.
    b. Suppose $\alpha(t)=2^{t/30}-1$, that is, an exponentially increasing proportion of the mining power hears about a new block up until $t_{max}=30$ seconds, at which point all have heard. What is the probability of at least one stale block being found?
    Note: You do not need to solve the integral analytically. You may use a computer.
    c. If we lowered β 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 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 in practice. In particular, explain why our model for $\alpha(t)$ from part(b) is incomplete.

3. **Power consumption:** In this exercise we'll look at two ways to estimate the power consumption of the Bitcoin network. Assume in your answers that the current difficulty of finding a bitcoin block is $d=10^{21}$ (close to the actual value of $2^{69.9}$), the current exchange rate is 1BTC= US$600 and that there are no transaction fees (only the fixed block reward of 12.5 BTC). Recall that energy is measured in joules (J) and power is in watts (W), where 1 W = 1 J/s.
    a. First, estimate the energy consumption of the network assuming all mining rewards are spent on electricity. Assume all electricity is purchased at US industrial rates, which are about US$0.05/kWH (and recall that 1 kWh = 3.6 MJ).
    b. Why might your estimate in part (a) be too high? Why might it be too low?
    c. Next, estimate the energy consumption of the entire network assuming all mining is being done with recent 16nm scale ASICs. Assume these devices can perform about 10^{10} SHA-256[2] computations per 1 J of electricity (about 0.1 J per GH).
    d. Why might your estimate in part (c) be too low? Why might it be too high?

4. **Feather forking:** In class we learned about *feather forking*: a coalition of miners controlling a fraction α of the total mining power attempts to censor transactions by announcing: "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 shown in a probabilistic state machine:



    a. What is the probability that the censorship 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 long (expressed as a number of blocks found) 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 $\alpha \to 1$.

5. **Mining pool sabotage:** Recall that mining pools enable individual miners to share risk and reward, lowering the variance of their earnings while keeping the same expected value. Participants repeatedly submit *shares* (blocks that are valid at a lower difficulty) to prove how much work they are doing. Whenever the pool finds a block, the coinbase from that block is split among the participants in proportion to the number of shares submitted. One risk is *sabotage*, in which a participant submits shares, but withholds full solutions if they are found.
    a. Consider two pools, P_1 and P_2 with mining power α_1 and α_2, respectively. What will P_1's expected share of the total earnings be if it dedicates $\beta<\alpha_1$ power towards sabotaging P_2? Note that when P_1 finds a block it gets the entire coinbase. When P_2 finds a block, P_1 receives a fraction of the coinbase proportional to the number of shares P_1 generated while mining for P_2.
    Hint: P_2's total mining power is now $\alpha_2+\beta$, but only α_2 is used for finding a new block. Because β power is no longer used to find blocks, P_2's useful mining power, as a fraction of the entire network, is now $\alpha_2/(1-\beta)$. The same reasoning also applies to P_1. You may assume that the block discovery rate is always 10 minutes.
    b. Provide concrete values for α_1, α_2, β in which this attack is advantageous for P_1.
    c. Suppose P_2 wants to protect itself by kicking out participants observed to be reporting a suspiciously low rate of valid blocks compared to how many shares they report. Explain why this might inadvertently punish honest participants.
    d. Suppose two pools, each with power α, sabotage each other with power $\beta<\alpha$. For what range of β will the two pools lose revenue by attacking each other? How much will they lose? What classic game from game theory is this situation an instance of?

6. **Alternate proof-of-work**: For a hash function $H: \{0,1\}^* \rightarrow \{1,\ldots,2^{256}\}$, consider the following puzzle: given a challenge c and a difficulty d, find nonces $n_1 \neq n_2$ such that:
$$H(c,n_1) = H(c,n_2) \pmod d$$
That is, the miner must find *two* nonces that collide under H modulo d. Clearly, puzzle solutions are easy to verify and the difficulty can be adjusted granularly.
    a. A simple algorithm is to repeatedly choose a random nonce n and add $(n, H(c,n))$ to a set L stored to allow efficient search by n (such as a hash map). The algorithm terminates when the last hash value added to L collides with some hash value already in L. For given values of d, approximately how many invocations of H are required in expectation to generate a solution n_1,n_2? How much memory does this use?
    Hint: it will be helpful to familiarize yourself with the birthday paradox.
    b. Consider an algorithm with limited memory that chooses one random nonce n_1 and then repeatedly chooses a random nonce n_2 until it finds an n_2 that collides with n_1. How many invocations of H will this algorithm use in expectation? We note that there is a clever algorithm that finds a solution in the same asymptotic time as part (A), but using only *constant* memory.
    c. Recall that a proof-of-work is *progress-free* if for all h,k (where $h \cdot k < B$ for some large bound B) the probability of finding a solution after generating $h \cdot k$ hashes is k times higher than the probability of finding a solution after just h hashes. Is this puzzle progress-free? Explain.