EE 374 - Blockchain Foundations
Practice Final
March 15, 2023

1. The exam has 5 questions with a total of 124 points. You have 3 hours to take the exam. Questions have different numbers of points so please allocate your time to each question accordingly.

2. Please write the answer to each question on a separate page and upload a photo or scan to Gradescope.

3. **All answers should be justified, unless otherwise stated.**

4. The exam is open-book, open-notes, open-internet.

Good luck!

(60 points) Problem 1

For the following questions, choose the one most fitting answer among the four choices. No justifications are required for this problem. 2 points for a correct answer, 0 points for an incorrect answer. 0.5 point for leaving the answer blank. Knowing you don't know something has value.

1. You are a passive observer and you notice that the ledger has not been including any new transactions over the past week.

   (a) You can deduce that safety is lost.

   (b) You can deduce that liveness is lost.

   (c) You can deduce that both safety and liveness are lost.

   (d) You can make no such deductions.

   Solution: It may be that no transactions were issued during the past week. The only way to be sure is if you're monitoring the mempool or issuing your own transactions.

2. Consider a proof-of-work longest chain protocol with $t = n/3$. What is the minimum chain quality an adversary can cause over the long run?

   (a) $\mu = 0$

   (b) $\mu = 1/4$

   (c) $\mu = 1/3$

   (d) $\mu = 1/2$

   Solution: The minimum chain quality can be calculated as follows:

   $$\mu \geq 1 - \frac{\text{max. rate of adversarial blocks}}{\text{min. rate of chain growth}} \geq 1 - \frac{t}{n-t} = \frac{1}{2}.$$

   This chain quality can be achieved by a selfish mining adversary in the long run.

3. Consider a Merkle tree with 1024 leaves that uses SHA256. How long is the proof size for one inclusion in this tree, in bits?

   (a) 1024

   (b) 10

   (c) 2560

   (d) 262,144

   Solution: The proof consists of $\log_2 1024 = 10$ hashes, each of which is 256 bits long.

4. Which of the following assurances does an existentially unforgeable signature scheme give?

(a) Given an existing message and a correct signature on it, the adversary cannot create a new, different signature on the same message verifiable under the same public key.

(b) Given a message and a correct signature, the adversary cannot recover the secret key.

(c) Given just a correct signature and a public key, the adversary cannot recover the message.

(d) All of the above.

Solution: To see that (b) is correct, suppose that an adversary $\mathcal{A}$ obtains the correct signature for a message and then uses that to recover the secret key. Then we can create another adversary $\mathcal{A}'$ that runs $\mathcal{A}$, obtains the secret key, and then forges the signature for any new message, thus breaking existential unforgeability. This is a contradiction. Option (a) is incorrect because existential unforgeability only guarantees that the adversary cannot forge signatures on messages that it has not queried before (recall the definition of the existential unforgeability game). To see that option (c) is incorrect, consider an existentially unforgeable signature scheme $\mathcal{S}$ and construct another signature scheme $\mathcal{S}'$ which concatenates the message and the signature from $\mathcal{S}$ to produce the new signature. Clearly, the message can be recovered from the new signature, but $\mathcal{S}'$ can be existentially unforgeable.

5. Consider a $\kappa = 256$ proof-of-work puzzle with $T = 2^{192}$. How many queries do you expect to need before you get a solution?

(a) 256

(b) 64

(c) 2^{192}

(d) 2^{64}

Solution: The probability that each query is successful is $p = \frac{T}{2^\kappa} = 2^{-64}$. The success of each query is an independent Bernoulli random variable. Therefore, the expected number of queries before getting a solution is $\frac{1}{p} = 2^{64}$.

6. Consider a $\kappa = 256$ proof-of-stake longest chain puzzle with $T_p = 2^{236}$. There are $n = 1{,}000{,}000$ nodes. What is the expected number of nodes that will win in a given round?

(a) 0

(b) 1

(c) 10

(d) unbounded

Solution: The probability that each query is successful is $p = \frac{T}{2^\kappa} = 2^{-20}$. By linearity of expectation, the expected number of nodes that will win in a given round is $pn \approx 1$.

7. Consider a $\kappa = 256$ proof-of-stake longest chain puzzle with $T_p = 2^{236}$. There are $n = 1{,}000{,}000$ nodes. What is the expected number of valid blocks that can be proposed in a given round?

   (a) 0

   (b) 1

   (c) 10

   (d) unbounded

   Solution: If an adversarial node has a successful query, she can propose as many valid blocks as she wants.

8. In order to ensure ledger safety in an honest majority setting, the macroeconomic policy of a cryptocurrency must mandate that:

   (a) Total coin supply rate must be non-increasing over time.

   (b) Total coin supply rate must be strictly decreasing over time.

   (c) Total coin supply rate must be non-decreasing over time.

   (d) None of the above.

   Solution: Ledger safety is guaranteed as long as the majority is honest regardless of the macroeconomic policy of a cryptocurrency.

9. In a proof-of-stake longest chain protocol, which of the following adversarial bounds are sufficient to achieve ledger safety and liveness respectively?

   (a) $t < n/3$ for both safety and liveness.

   (b) $t < 2n/3$ for both safety and liveness.

   (c) $t < n/3$ for safety, but $t < 2n/3$ for liveness.

   (d) $t > n/3$ for safety, but $t > 2n/3$ for liveness.

   Solution: A proof-of-stake longest chain protocol is safe and live as long as $t < \frac{n}{2}$, so $t < \frac{n}{3}$ is sufficient, although not necessary.

10. If H is a random oracle, then $G(x) = H(H(x))$ is:

    (a) Collision resistant, but not preimage resistant.

    (b) Preimage resistant, but not second preimage resistant.

    (c) Behaving like a random oracle.

    (d) None of the above.

Solution: Recall that a random oracle returns a response chosen uniformly at random for every query and returns the same output if the query is repeated. For every x_1, x_2 such that $x_1 = x_2$, $H(x_1) = H(x_2)$ and, therefore, $H(H(x_1)) = H(H(x_2))$. For a fresh query x, $H(x)$ is chosen uniformly at random. The probability that $H(x)$ will appear in the previous cache of H as an input is negligible. Therefore, with overwhelming probability, the random oracle will pick yet another uniformly random number to issue as the output of $H(H(x))$.

11. Under a block size limit (in bytes), a rational miner prioritizes transactions by:

    (a) Their size, in bytes, in increasing order.

    (b) Their fees, in decreasing order.

    (c) The ratio fees / byte, in increasing order.

    (d) None of the above.

    Solution: The rational miner prioritizes transactions by ratio fees / byte, but in *decreasing* order.

12. Let G be a second-preimage resistant hash function. Consider the hash function $H(x)$ that prepends the fixed string "0110" to the output of $G(x)$. The function H might *fail* to be:

    (a) Collision resistant

    (b) Preimage resistant

    (c) Second-preimage resistant

    (d) None of the above

    Solution: $H(x)$ is second-preimage resistant, since $G(x)$ is second-preimage resistant. Recall that second-preimage resistance implies preimage resistance. However, it might not be collision resistant, if G is not collision resistant.

13. The data that needs to be downloaded by an SPV light wallet holding an address that has been used to send or receive money a constant number of times and is synchronizing for the first time is:

    (a) Linear in the chain length, but only logarithmic in the number of transactions per block.

    (b) Linear in the chain length and linear in the number of transactions per block.

    (c) Logarithmic in the chain length, but linear in the number of transactions per block.

    (d) Logarithmic in the chain length and logarithmic in the number of transactions per block.

Solution: An SPV light wallet has to download headers of all blocks to verify the chain structure and proof-of-work (linear size in the chain length) and Merkle proofs (logarithmic size in the number of transactions per block) for transactions related to the address.

14. When the hashrate of the network increases, the variable difficulty proof-of-work protocol:

    (a) Decreases T so that participants are incentivized to decrease q.

    (b) Decreases T so that f is decreased over the long run and convergence opportunities become denser and denser.

    (c) Decreases p in order to keep f constant over the long run.

    (d) Increases T in order to raise the difficulty.

    Solution: When the hashrate of the network increases, the difficulty increases too, so T decreases. Thus, the probability of a successful query p decreases too. The goal of the protocol is to keep f constant, since if it increases or drops we have problems (of safety and liveness respectively).

15. Which ledger virtues can a temporary majority adversarial miner break in a way that is unhealable (they are not regained after any point in time) in the proof-of-work protocol?

    (a) Safety.

    (b) Liveness.

    (c) Both.

    (d) Neither.

    Solution: When honest majority is regained, chain quality, thus, liveness is recovered. Similarly, common prefix and, therefore, safety is recovered.

16. Our Marabu protocol was attacked by a charming selfish mining adversary building a new chain from genesis. Many students are falsely claiming they are this Adversary. What is an appropriate way for the adversary to prove her identity?

    (a) Open source her mining code on GitHub.

    (b) Place her SUID in the "note" field of a new block on top of the adversarial chain.

    (c) Sign her SUID using the coinbase key of the first block in the attack.

    (d) Generate a new public/private key pair and use it to sign a message containing both the latest adversarial tip as well as her SUID concatenated together.

    Solution: Anyone can do parts (a), (b) and (d) but only the miner that produced the first block in the attack can sign the message with the corresponding private key. Part (a) is easy, since, once the Adversary open sources her GitHub, everyone else can fork it. Part (b) is easy if the attack has stopped. It doesn't take much effort to mine

a single block on top. Part (d) is easy, as anyone can generate a new key and sign anything with it.

17. Which of the following functions is negligible?

    (a) $1/n^2$

    (b) $e^{-n/3}$

    (c) $1/\log(\log(n))$

    (d) All of the above.

    Solution: $1/n^2$ is not negligible, as it is an inverse polynomial. Similarly, $1/\log(\log(n))$ is not negligible, because $\log \log n$ is even smaller than a polynomial. However, $e^{-n/3}$ is negligible, because it is a negative exponential.

18. Double spending transactions, in which some transaction believed to be confirmed is later reverted and becomes unconfirmed, are avoided:

    (a) In both the UTXO and the accounts model by a signature.

    (b) In both the UTXO and the accounts model by a nonce.

    (c) In the UTXO model by a signature, and in the accounts model by the nonce.

    (d) By the underlying blockchain.

    Solution: A safe blockchain guarantees that confirmed transaction remains confirmed later in time. Even if the signature and nonces are correct, a broken blockchain can cause safety issues and a confirmed transaction to be reverted. An adversary can create duplicate signatures and duplicate nonces, but only the blockchain can ensure that only one of them is accepted.

19. Why can't a coinbase transaction generate more money than the macroeconomic policy mandates?

    (a) It will invalidate the coinbase signature, and this will be checked by every node before propagating the block further.

    (b) It will invalidate the Law of Conservation, that input values must exceed output values.

    (c) Every node will do a hard-coded check that the output value is as required.

    (d) The coinbase transaction will be spending money that is not in the UTXO set.

    Solution: The amount of money generated in the coinbase transaction is specified by the protocol, which is hard-coded. Even a valid coinbase transaction does not have a signature, does not respect the Law of Conservation, and does not spend from the UTXO set.

20. During a chain reorg, transactions evicted from the chain are:

    (a) Placed back into the mempool if still valid.

(b) Evicted from the chain, but manually inserted into the ledger.

(c) Placed in the next block template, but not in the mempool.

(d) Discarded.

Solution: There is no guarantee that all transactions evicted from the chain are valid according to the new chain state. Thus, they should be verified first and, if valid, placed in the mempool.

21. Imagine a network where suddenly the non-eclipsing assumption no longer holds. What is the worst thing a non-mining adversary can do to a proof-of-work longest chain protocol?

(a) Transactions get confirmed, and they are never reverted in the future.

(b) Transactions get confirmed, but may be reverted in the future.

(c) Transactions are included in the chain, but may not be confirmed.

(d) No transactions ever make it to the chain.

Solution: If the non-eclipsing assumption no longer holds, the network is partitioned. The proof-of-work longest chain protocol is not safe, as different chains can be built in different parts of the network partition. If there exists a different longer chain in one of the network parts, the adversary can broadcast it to other parts and unconfirm the local chains. However, if at least one honest miner resides in one of the partitions, the chain will make progress due to chain growth, and transactions will keep getting k-confirmed, albeit at a slower rate.

22. How are peers discovered in a peer-to-peer network such as a blockchain network?

(a) The client connects to a secure HTTPS server which gives us a list of peers.

(b) The peers are discovered by connecting to a decentralized database such as a shared instance of MySQL, Postgres, or Mongodb that contains a list of all known peers. The database is replicated to ensure reliability.

(c) The list of peers is fixed and includes at least one known trustworthy honest peer such as the IP of one of the developers. This list is never updated to avoid introducing adversarial peers.

(d) Some peers are hardcoded in the code so that we can connect to them initially. The rest are discovered by asking our peers for their peers.

Solution: An HTTPS server and a database such as MySQL are centralized solutions. So is relying on the developers. Recall that our Marabu protocol uses (d) as a peer discovery mechanism, as do other peer-to-peer systems (Bitcoin, Ethereum, BitTorrent).

23. What is the best way to avoid denial-of-service attacks of fake blocks?

(a) Check the proof-of-work first, then download the transactions, then download the parent.

(b) Download and validate the transactions first, then check the proof-of-work, then download the parent.

(c) Download and validate the transactions first, then download the parent, and finally check the proof-of-work.

(d) Make sure the parent is available first by recursively downloading it and validating it, then check the proof-of-work, and only afterwards download the transactions.

Solution: Generating valid transactions and a correct chain structure is computationally fast for an attacker but finding a block with correct proof-of-work is the most computationally expensive part. Thus, the best way is to verify the proof-of-work first. Think of how long it would take for an adversary to cause a node to process 1 TB of data: If proof of work is checked first, this will take a very long time.

24. A malicious full node, when asked by an SPV node, pretends a transaction is in the chain, while it is not. What will the SPV node do?

(a) Accept the transaction, as it cannot check it itself; it relies on the full node for security.

(b) Ask other full nodes if they have accepted the transaction and take a majority vote.

(c) Reject the transaction, as the Merkle proof does not check out.

(d) Reject the transaction, because the transaction signature does not validate.

Solution: As the SPV node has downloaded the stable header chain, the adversary cannot provide a valid Merkle tree inclusion proof that passes the check.

25. Why does the Chain Growth property require a minimum number of rounds parameter s?

(a) So that the adversary has enough time to run.

(b) So that the expectation $\mathbb{E}[X]$ attains the lower bound we require.

(c) So that the Chernoff bound can be applied to the number of successful rounds X.

(d) So that the adversarially successful queries Z concentrate to a value lower than the convergence opportunities Y.

Solution: The Chain Growth Lemma relies on X to ensure the chain grows. The concentration of the X variable near its mean is ensured by typicality, which is proven by the Chernoff bound. The running time of the adversary is irrelevant to chain growth, as the adversary may choose to withhold her blocks. The expectation of X does not change with time. Lastly, the relationship of Z with respect to Y is useful for proving Common Prefix, not Chain Growth.

26. When representing the UTXO model as a State Machine Replication problem, what is the type of the output of the transition function $\delta(st, tx)$?

(a) A UTXO transaction.

(b) The current balances of the whole system, together with the nonce of each account.

(c) A set of unspent outputs.

(d) True or false, depending on whether the transaction can be applied to the previous state.

Solution: The output of the transition function is the system state. In UTXO model, the state is the set of all unspent outputs.

27. In a proof-of-work longest chain protocol, who can predict which miner will win the next block?

(a) Anybody.

(b) The adversary.

(c) The miner who will win the next block.

(d) Nobody.

Solution: The proof-of-work lottery is random and unpredictable.

28. In a proof-of-stake longest chain protocol using a hash function for the puzzle, who can predict which node will win in the next round?

(a) Anybody who knows the public keys of the nodes.

(b) The adversary.

(c) The node who will win in the next round.

(d) Nobody.

Solution: In such a protocol, the puzzle includes the hash of Genesis, a round number and a public key of a protocol participant. Thus, anyone who knows the public keys of the nodes can predict the next round winner.

(8 points) Problem 2

Consider a proof-of-work longest chain protocol with a 1/3 adversary in a population of $n = 3$ nodes with a hash rate of $q = 3$. The security parameter is $\kappa = 256$.

1. (2 points) What is the honest advantage δ?

   Solution: The honest advantage δ:

   $$t < (1-\delta)(n-t) \Rightarrow \tag{1}$$
   $$\delta < 1 - \frac{t}{n-t} \tag{2}$$
   $$= \frac{1}{2} \tag{3}$$

2. (2 points) Choose numeric parameters ϵ and f to ensure safety and liveness.

   Solution: Balancing equation:

   $$3\epsilon + 3f \leq \delta$$

   We can choose $\epsilon = f = \frac{\delta}{6} < \frac{1}{12}$. On the borderline case, we can set $\epsilon = f = \frac{1}{12}$.

3. (2 points) Calculate the exact numeric probability p of a successful query.

   Solution: The probability of honest parties getting a block within the unit of time:

   $$f = 1 - (1-p)^{q(n-t)} \Rightarrow \tag{4}$$
   $$p = 1 - (1-f)^{\frac{1}{q(n-t)}} \tag{5}$$
   $$= 0.0144 \tag{6}$$

4. (2 points) Calculate a numeric value for the mining target T to match the above.

   Solution:

   $$p = \frac{T}{2^\kappa} \Rightarrow \tag{7}$$
   $$T = 2^\kappa p \tag{8}$$
   $$= 2^{256} 2^{-6.118} \tag{9}$$
   $$\approx 2^{250} \tag{10}$$

You can use a calculator such as Python and round your numbers to three significant digits.

(20 points) Problem 3

Consider the longest chain proof-of-stake protocol we studied in class. We will look at an instantiation where $\Delta = 1$ second, $n = 100$, $\kappa = 128$, $T_p = 2^{118}$. For answering the questions below, you are free to choose the parameter k in the confirmation rule. You can use a calculator such as Python and round your numbers to three significant digits.

1. First suppose all 100 nodes are honest.

   (a) (1 point) Compute the expected growth rate of the longest chain, in blocks per second.

   Solution: In the longest chain proof-of-stake protocol we studied, every staker has one coin which results in one query per round. The probability of a successful query:

   $$p = \frac{T_p}{2^\kappa} \tag{11}$$
   $$= 2^{-10} \tag{12}$$
   $$\approx 0.001 \tag{13}$$

   The probability of a successful round, i.e. the probability that at least one honest staker finds a block:

   $$f = 1 - (1-p)^n \tag{14}$$
   $$= 1 - (0.999)^{100} \tag{15}$$
   $$= 0.095 \tag{16}$$

   The chain grows by one block if the round is successful or remains of the same length otherwise. Thus, the expected chain growth rate is 0.095 blocks per second.

   (b) (1 point) Compute the expected growth rate of the k-confirmed chain, in blocks per second. (The k-confirmed chain is the portion $C[:-k]$.)

   Solution: Every staker is honest, so every block gets confirmed after becoming k-deep. Thus, the expected growth rate of the k-confirmed chain is 0.095 blocks per second.

   (c) (2 points) If we double T_p, does the expected growth rate of the longest chain double, more than double, or less than double? What about the growth rate of the k-confirmed chain?

   Solution: New target is $T_p = 2^{119}$. The probability of a successful query $p = 2^{-9} = 0.002$ and the probability of a successful round $f = 0.181$. The expected chain growth rate is 0.181 blocks per second which is less than double of the previous value of the expected chain growth rate. Intuitively, this is because when there are multiple successful honest queries in the same round, the chain can still only grow by one block.

2. Now suppose 20 of the 100 nodes are adversary, the other 80 are honest.

(a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain. (A "lower bound" means that it is a lower bound irrespective of the adversary's attack strategy; "tight" means that the lower bound is attainable for some adversary's attack strategy.)

Solution: The honest majority assumption holds for such parameters. However, adversary can choose not to participate in the protocol. Thus, the probability of a successful round, i.e. the probability that at least one honest staker finds a block:

$$f = 1 - (1-p)^n \tag{17}$$
$$= 1 - (0.999)^{80} \tag{18}$$
$$= 0.077 \tag{19}$$

Thus, a lower bound on the expected growth rate of the longest chain is 0.077 blocks per second. Due to Chain Growth, we know that the adversary can't reduce the growth rate any further.

(b) (1 point) Compute a tight lower bound on the expected growth rate of the k-confirmed chain.

Solution: Every block in the chain gets confirmed when it becomes k-deep. Thus, a lower bound on the expected growth rate of the k-confirmed chain is 0.077 blocks per second.

(c) (2 points) Is the protocol safe? Is the protocol live?

Solution: This protocol is both safe and live since $t = 20 < 80 = n - t$

3. Now suppose 20 nodes are still adversary, but instead of having the full 80 honest nodes online, 30 of them decide not to participate in the protocol and went on vacation to the Bermudas. However, the protocol designer does not know this and the protocol parameters are not adjusted.

(a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain.

Solution: There are 70 nodes in total: 20 adversarial and 50 honest. Again, the honest majority assumption holds for such parameters. However, the adversary can choose not to participate in the protocol. Thus, the probability of a successful round, i.e. the probability that at least one honest staker finds a block:

$$f = 1 - (1-p)^n \tag{20}$$
$$= 1 - (0.999)^{50} \tag{21}$$
$$= 0.049 \tag{22}$$

Thus, a lower bound on the expected growth rate of the longest chain is 0.049 blocks of transactions per second. Again, the adversary can't reduce growth rate further due to the Chain Growth virtue.

(b) (1 point) Compute a tight lower bound on the growth rate of the k-confirmed chain.

Solution: Every block gets confirmed when it becomes k-deep. Thus, a lower bound on the growth rate of the k-confirmed chain is 0.049 blocks of transactions per second.

(c) (2 points) Is the protocol safe? Is the protocol live?

This protocol is both safe and live since $t = 20 < 50 = n - t$

4. Now suppose a further 35 honest nodes went on vacation.

(a) (1 point) Compute a tight lower bound on the expected growth rate of the longest chain.

Solution: There are 35 nodes in total: 20 adversarial and 15 honest. Note that the honest majority assumption does not hold for such parameters. However, adversary can choose not to participate in the protocol. The probability of a successful round, i.e. the probability that at least one honest staker finds a block:

$$f = 1 - (1-p)^n \tag{23}$$
$$= 1 - (0.999)^{15} \tag{24}$$
$$= 0.015 \tag{25}$$

Thus, a lower bound on the expected growth rate of the longest chain is 0.015 blocks per second. Even though honest majority doesn't hold, Chain Growth still holds, so the adversary cannot reduce the growth rate further.

(b) (1 point) Compute a tight lower bound on the expected growth rate of the k-confirmed chain.

Solution: A lower bound on the expected growth rate of the k-confirmed chain is 0.015 blocks per second.

(c) (2 points) Is the protocol safe? Is the protocol live?

Solution: Such protocol is not safe and not live since the honest majority assumption does not hold.

5. (4 points) A protocol is said to be *available* if it is safe and live whenever the number of honest nodes online is greater than the number of adversary nodes. Is the longest chain PoS protocol available?

Solution: As examples in parts 3 and 4 illustrate, PoS longest chain protocol is available.

(18 points) Problem 4

Consider the Streamlet protocol we studied in class.

1. (2 points) Streamlet is said to be *partition tolerant* whenever the number of honest nodes exceeds $2n/3$, where n is the total number of nodes. Explain what that means.

   Solution: A protocol is partition tolerant if it is safe under network partition. In the last lecture we showed that under network partition Streamlet is safe but not live.

2. (2 points) Streamlet is said to be $1/3$-*accountable*. Explain what that means.

   Solution: This means that in the event of a safety violation, at least $\frac{1}{3}n$ nodes are provably identified as protocol violators. In the case of Streamlet, to produce a safety violation at least $\frac{1}{3}n$ nodes have to double vote, which can be provably identified via signatures.

3. Now suppose $\Delta = 1$ second and $n = 100$ nodes and the nodes are all honest.

   (a) (1 point) Compute the expected growth rate of the longest notarized chain (in blocks per second).

   Solution: We assume that the network is synchronous. If every node is honest, then every proposed block is valid and gets notarized. Thus, the chain will grow by one block every epoch, which is $2\Delta = 2\,\text{s}$. The growth rate is $\frac{1}{2}$ block per second.

   (b) (1 point) Compute the expected growth rate of the finalized chain (in blocks per second).

   Solution: Since every node is honest, every notarized block from epoch $i, i > 1$ gets finalized in epoch $i+1$. The growth rate of the finalized chain is also $\frac{1}{2}$ block per second.

4. Now suppose 20 of the 100 nodes are adversary and the other 80 are honest.

   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest notarized chain. (A "lower bound" means that it is a lower bound irrespective of the adversary's attack strategy; "tight" means that the lower bound is attainable for some adversary's attack strategy.)

   Solution: The number of adversarial nodes is $20 < \frac{100}{3}$, therefore, they cannot cause a safety violation. However, adversary can choose to not participate in the protocol, which results in slowing down the chain growth. In such case, only 80 out of 100 nodes propose blocks. $80 \geq \frac{2}{3}n \approx 67$, so there are enough honest nodes get proposed blocks notarized. A lower bound on the expected growth rate of the longest chain $\frac{80}{100} \cdot \frac{1}{2} = \frac{2}{5}$ blocks per second.

   (b) (1 point) Compute a lower bound on the expected growth rate of the finalized chain. (Your lower bound does not need to be tight. However if the lower bound is not tight, it must be positive.)

Solution: Recall Streamlet Liveness theorem - if there are five consecutive epochs with honest leaders, then at least one more block is confirmed at the end of the five epochs compared to the beginning. The probability of five consecutive epochs being honest is $(\frac{80}{100})^5 = 0.328 \approx \frac{1}{3}$. So, an upper bound on the average time for a block to get finalized is 3 superepochs or 30 s. Hence, a lower bound on the growth rate of the finalized chain is $\frac{1}{30}$ blocks per second.

(c) (2 points) Is the protocol safe? Is it live?

Solution: The protocol is safe since $t = 20 < \frac{100}{3} = 33$. The network is synchronous, so the the protocol is also live.

5. Now suppose 20 nodes are still adversary, but instead of having the full 80 honest nodes online, 30 of them decide not to participate in the protocol and went on vacation to the Bermudas. However, the protocol designer does not know this and the protocol parameters are not adjusted.

   (a) (1 point) Compute a tight lower bound on the expected growth rate of the longest notarized chain.

   Solution: The number of adversarial nodes is $20 < \frac{100}{3}$, therefore, they cannot cause a safety violation. However, adversary can choose to not participate in the protocol. There are only $50 < \frac{2}{3}n \approx 67$ honest nodes. Blocks do not receive enough votes and do not get notarized. Thus, a lower bound on the expected growth rate of the longest notarized chain is 0.

   (b) (1 point) Compute a lower bound on the expected growth rate of the finalized chain. (Your lower bound does not need to be tight. However if the lower bound is not tight, it must be positive.)

   Solution: A lower bound on the expected growth rate of the finalized chain is also 0.

   (c) (2 points) Is the protocol safe? Is the protocol live?

   Solution: $t = 20 < \frac{100}{3} = 33$, so by the quorum intersection argument, adversarial nodes cannot produce two notarized blocks in the same epoch. However, if the adversary chooses not to participate, no block ever gets notarized. Thus, such protocol is still safe but not live.

6. (4 points) A BFT protocol is said to be *available* if it is safe *and* live whenever the number of honest nodes online is greater than twice the number of adversary nodes. Is Streamlet available?

   Solution: As the example in part 5 illustrates, Streamlet is not available. The number of honest nodes online is 50, which is greater than twice the number of adversarial nodes, 20. However, we showed that such protocol is still safe but not live, so it is not available.

(18 points) Problem 5

Answer the following questions in the Backbone model with static difficulty.

1. Consider executions with parameters $(n, q, t, f, \epsilon, T, \mu, \ell, s, \tau, u, k)$ of your choice, *without* honest majority, but with $n - t > 0$.

   (a) (4 points) Describe an execution where Safety is violated.
   Solution: See Figure 1. The parameters are $n = 12, t = 8, k = 3$.

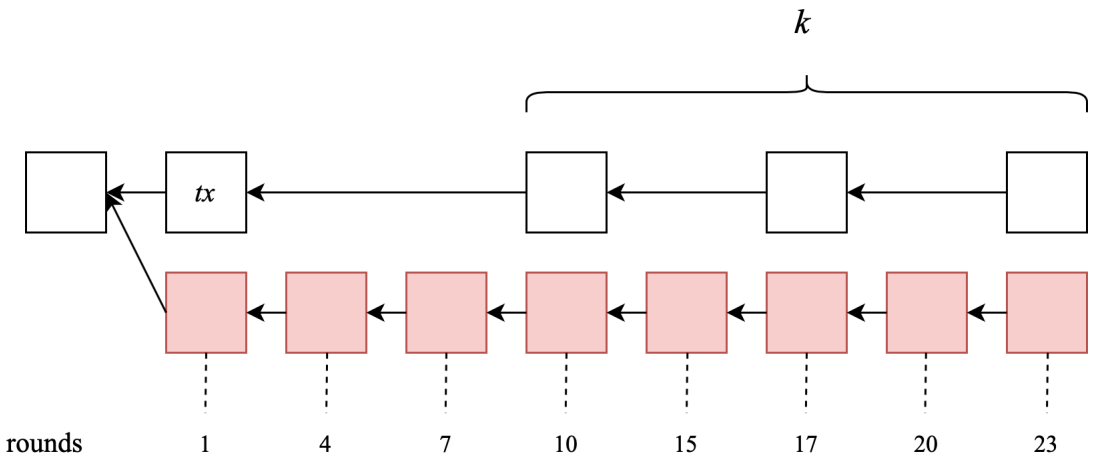

Figure 1: Private attack in dishonest majority execution. White blocks are honest and red blocks are adversarial. Adversary mines a chain in private and releases it after honest nodes have adopted the honest chain with 4 blocks, displacing the honest chain. The honest chain contains a transaction tx but the adversarial chain is longer and does not contain tx. After the honest block in round 23 is mined, tx gets confirmed because it is in a k-deep block. Adversary releases her chain after that, deconfirming tx and causing a safety violation.

   (b) (4 points) Describe an execution where Liveness is violated.
   Solution:
   See Figure 2. Choice of $n = 10$ and $t = 5$ results in the minimum of chain quality of $\mu \geq 1 - \frac{t}{n-t} = 0$. Thus, the adversary can cause a liveness violation. Such an execution can occur with high probability because due to adversarial majority, the adversary will get a block before the honest parties most of the time.

   (c) (1 point) Is it possible to have an execution with a Safety violation if chain quality is *not* violated?
   Solution: See Figure 3.

   (d) (1 point) Is it possible to have an execution with a Liveness violation if chain quality is *not* violated?
   Solution: No, chain quality and chain growth imply ledger liveness. Since chain quality is not violated and chain growth is not violated even in an execution with adversarial majority, liveness is also not violated.

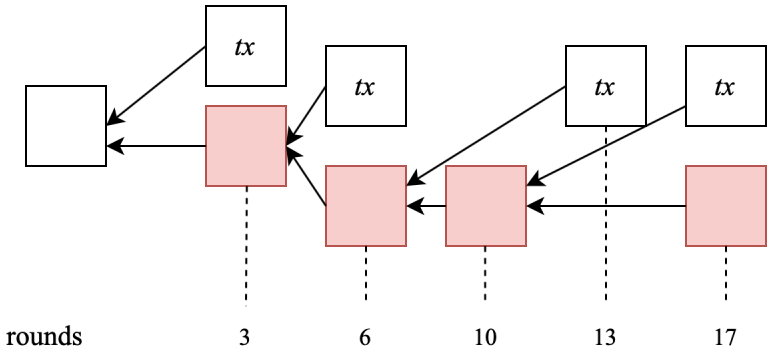

Figure 2: Selfish mining strategy. White blocks are honest and red blocks are adversarial. Recall that selfish mining adversary mines at the tip of the longest chain but keeps her blocks private. Every time an honest block is mined, she releases her block at the same level if she has one. Honest block mined in round 3 contains a transaction tx but gets displaced by an adversarial block. This adversarial block does not contain any transactions. This process gets repeated at every round an honest block is mined, so no transactions get included in the chain, causing a liveness violation. In this example, liveness is violated for $u = 13$ because honest parties attempted to include tx in rounds $3, ..., 16$ but the ledger at round 17 (with $k = 3$) does not contain tx.

For questions (a) and (b) above: What is the strategy of the adversary? What blocks and what transactions must the adversary produce to cause this ledger virtue violation? Draw the block tree and timeline of the execution illustrating which round each block was mined in, which transactions are included in which block, whether a block was computed by an honest or an adversarial party, and which honest parties have adopted which chain. You do not need to calculate values of parameters that are not necessary to support the respective statement.

2. Consider an execution with $n = 3$ parties of which $t = 1$ is adversarial, target $T = 2^{226}$, security parameter $\kappa = 256$ and hash rate $q = 1$, and $k = 6$.

   (a) (2 points) Calculate the numeric probability of a successful round.
   Solution: The probability of a successful query:

   $$p = \frac{T}{2^\kappa} \tag{26}$$
   $$= 2^{-30} \tag{27}$$

   The probability of a successful round:

   $$f = 1 - (1 - p)^{q(n-t)} \tag{28}$$
   $$= 1.86 * 10^{-9} \tag{29}$$

   (b) (2 points) Calculate the numeric probability of a convergence opportunity.
   Solution: Probability of a convergence opportunity is lower bounded by $q(n - t)p(1 - p)^{q(n-t-1)} = 1.86 * 10^{-9}$. Since the probability f is quite small, this bound is very close to the actual value.

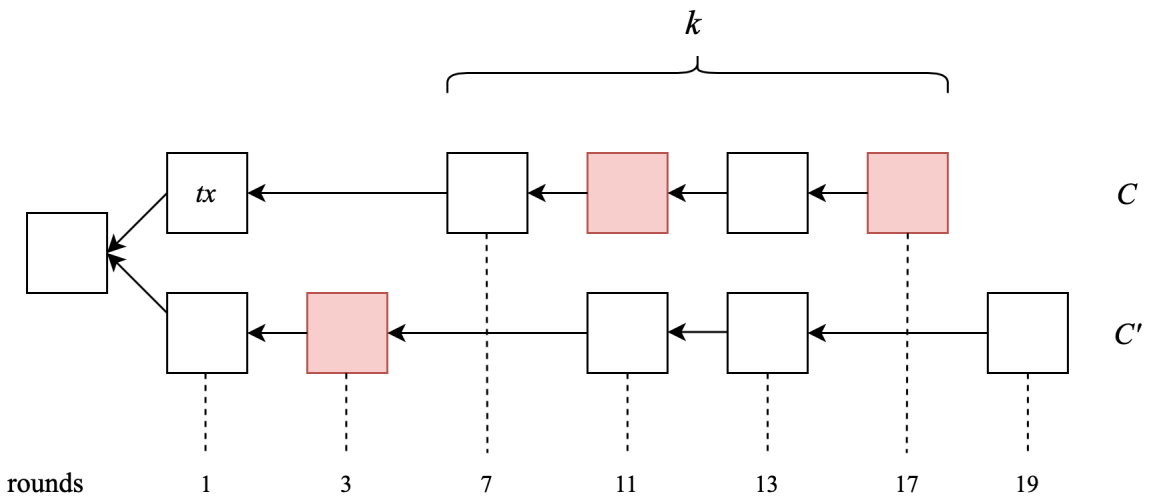

Figure 3: Safety violation scenario for $k = 4$. White blocks are honest and red blocks are adversarial. Rounds 7, 11 and 19 are convergence opportunities. Adversary matches honest blocks in these rounds with her block, forcing honest parties to work on different chains C and C'. To match the honest block in round 7, she releases her block from round 3. In round 11, she matches the honest block with her block mined in the same round. The honest block in round 19 is matched by adversarial block from round 17. C contains tx, which gets confirmed in round 19 while C' does not contain it. However, chain quality is not violated: $\mu = \frac{3}{5}$ and $\mu' = \frac{4}{5}$.

(c) (2 points) What is the numeric probability that the first 10 rounds are all successful?

Solution: The probability that the first 10 rounds are all successful is $f^{10} = 5.02 * 10^{-88}$.

You can use a calculator such as Python and round your numbers to three significant digits.

3. (2 points) In the above scenario, we give the adversary the fictitious ability to "snoop" all the queries to the Random Oracle and to choose whatever κ-bit answer she wishes to one (honest or adversarial) fresh query *per round*. The Random Oracle continues to use its cache to respond consistently. Can this adversary break common prefix? What strategy should she follow?

Solution: The adversary can snoop one of her own queries in every round and get one successful block per round. All she needs to do is make sure she gives (different) responses that are all below the target T. She can use this ability to mine an adversarial chain in private, starting at height h until she gets k blocks. She will likely be able to do that much prior to the honest parties mining a chain of height $h + k$, as she can get one valid block per round. When the honest parties mine their chain of height $h + k$, the adversary releases her privately mined $h + k$ chain, causing a common prefix violation on the chain. If she had also included conflicting transactions in the chains,

she could additionally cause a safety violation.

4. (3 bonus points) In the above scenario, we give the adversary the fictitious ability to "snoop" all the queries to the Random Oracle and to choose whatever κ-bit answer she wishes to one (honest or adversarial) fresh query *per execution*. The Random Oracle continues to use its cache to respond consistently. Can this adversary break common prefix? What strategy should she follow?

   Solution: The adversary starts mining a new, independent chain $\mathcal{C}'$, whose first block $\mathcal{C}'[0]$ uses as *previd* the value 0^κ (any fresh value below T will do). This chain does not start from the genesis block, for now. When she has mined such a chain of length k, she stops mining and waits. In the meantime, the honest parties have mined a larger chain $\mathcal{C}$, but no matter. The adversary waits for an honest party to make a query that mines on top of $\mathcal{C}$ and snoops on that query. She forces the random oracle to give value 0^κ as the output for that query. This is a valid block B, as $0 < T$, and B is extending $\mathcal{C}$ which extends genesis. However, now $\mathcal{C}'$ is a chain that extends B and is k blocks longer than $\mathcal{C}$, and also extends genesis. The adversary waits for the honest parties to mine a further k blocks on top of their own chain, and subsequently releases $\mathcal{C}'$, causing a common prefix violation. If she had also included conflicting transactions in the chains, she could additionally cause a safety violation.

Reference

Variables

- κ: The security parameter
- $\mathcal{A}$: The adversary
- Π: The honest protocol
- $\mathcal{G}$: The genesis block
- Δ: The network delay (in backbone, $\Delta = 1$)
- H: The hash function
- n: The total number of parties
- t: The adversarial number of parties
- q: Hash rate of one party per round
- T: The mining target
- p: Probability of a successful query
- δ: The honest advantage
- k: Common prefix parameter
- μ: Chain quality parameter (the honest ratio of blocks)
- ℓ: Chain quality chunk length (in blocks)
- τ: Chain growth rate (in blocks per round)
- s: Chain growth duration (in rounds)
- f: Probability of successful round
- ϵ: Chernoff bound error
- λ: Chernoff bound duration
- X: Successful round indicator
- Y: Convergence opportunity indicator
- Z: Adversarially successful query indicator

Formulae

- The honest majority assumption: $t < (1 - \delta)(n - t)$.
- The balancing equation: $3f + 3\epsilon \leq \delta$.
- The proof-of-work equation: $H(B) \leq T$.
- The proof-of-stake equation: $H(s_0 \,\|\, pk \,\|\, r) \leq T_p$.

Algorithms

Algorithm 1 The collision resistance game.

function $\textsc{Collision}_{H,\mathcal{A}}(\kappa)$
    $x_1, x_2 \leftarrow \mathcal{A}(1^\kappa)$
    return $x_1 \neq x_2 \wedge H_\kappa(x_1) = H_\kappa(x_2)$
end function

Algorithm 2 The preimage resistance game.

function $\textsc{Preimage}_{H,\mathcal{A}}(\kappa)$
    $x \xleftarrow{\$} \{0,1\}^{2\kappa}$
    $y \leftarrow H_\kappa(x)$
    $x^* \leftarrow \mathcal{A}(y)$
    return $H_\kappa(x^*) = H_\kappa(x)$
end function

Algorithm 3 The second preimage resistance game.

function $\textsc{2nd-Preimage}_{H,\mathcal{A}}(\kappa)$
    $x \xleftarrow{\$} \{0,1\}^{2\kappa}$
    $x' \leftarrow \mathcal{A}(x)$
    return $H_\kappa(x) = H_\kappa(x') \wedge x \neq x'$
end function

Algorithm 4 The existential forgery game for a signature scheme (Gen, Sig, Ver).

function existential-forgery-game$_{Gen,Sig,Ver,\mathcal{A}}(\kappa)$
    $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
    $M \leftarrow \emptyset$
    function $\mathcal{O}$(m)
        $M \leftarrow M \cup \{m\}$
        return $\mathsf{Sig}(sk, m)$
    end function
    $m, \sigma \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
    return $\mathsf{Ver}(pk, \sigma, m) \wedge m \notin M$
end function

Algorithm 5 The Random Oracle

$r \leftarrow 0$
$\mathcal{T} \leftarrow \{\}$ ▷ Initiate Cache
$Q \leftarrow 0$ ▷ q for honest parties, qt for adversary
function $H_\kappa(x)$
    if $x \notin \mathcal{T}$ **then** ▷ First time being queried
        if $Q = 0$ **then** ▷ Out of Queries
            return $\perp$
        end if
        $Q \leftarrow Q - 1$
        $\mathcal{T}[x] \xleftarrow{\$} \{0,1\}^\kappa$
    end if
    return $\mathcal{T}[x]$ ▷ Return value from Cache
end function

Algorithm 6 The environment.

$r \leftarrow 0$
function $\mathcal{Z}^{n,t}_{\Pi,\mathcal{A}}(1^\kappa)$
    $\mathcal{G} \xleftarrow{\$} \{0,1\}^\kappa$ ▷ Genesis block
    for $i \leftarrow 1$ to $n-t$ **do** ▷ Boot stateful honest parties
        $P_i \leftarrow$ new $\Pi(\mathcal{G})$
    end for
    $A \leftarrow$new $\mathcal{A}(\mathcal{G}, n, t)$ ▷ Boot stateful adversary
    $\overline{M} \leftarrow []$ ▷ 2D array of messages
    for $i \leftarrow 1$ to $n-t$ **do**
        $\overline{M}[i] \leftarrow []$ ▷ Each honest party has an array of messages
    end for
    while $r < \mathsf{poly}(\kappa)$ **do** ▷ Number of rounds
        $r \leftarrow r + 1$
        $M \leftarrow \emptyset$
        for $i \leftarrow 1$ to $n-t$ **do** ▷ Execute honest party i for round r
            $Q \leftarrow q$ ▷ Maximum number of oracle queries per honest party (Section 2)
            $M \leftarrow M \cup \{P_i.\mathsf{execute}^H(\overline{M}[i])\}$ ▷ Adversary collects all messages
        end for
        $Q \leftarrow tq$ ▷ Max number of Adversarial oracle queries
        $\overline{M} \leftarrow A.\mathsf{execute}^H(M)$ ▷ Execute rushing adversary for round r
        for $m \in M$ **do** ▷ Ensure all parties will receive message m
            for $i \leftarrow 1$ to $n-t$ **do**
                $\mathsf{assert}(m \in \overline{M}[i])$ ▷ Non-eclipsing assumption
            end for
        end for
    end while
end function

Algorithm 7 The honest party

$\mathcal{G} \leftarrow \epsilon$
function CONSTRUCTOR($\mathcal{G}'$)
    $\mathcal{G} \leftarrow \mathcal{G}'$ ▷ Select Genesis Block
    $\mathcal{C} \leftarrow [\mathcal{G}]$ ▷ Add Genesis Block to start of chain
    round $\leftarrow 1$
end function
function EXECUTE(1^κ)
    $\tilde{\mathcal{C}} \leftarrow \text{maxvalid}(\mathcal{C}, \bar{M}[i])$ ▷ Adopt Longest Chain in the network
    if $\tilde{\mathcal{C}} \neq \mathcal{C}$ **then**
        $\mathcal{C} \leftarrow \tilde{\mathcal{C}}$
        BROADCAST($\mathcal{C}$) ▷ Gossip Protocol
    end if
    $x \leftarrow$ INPUT() ▷ Take all transactions in mempool
    $B \leftarrow \text{PoW}(x, H(\mathcal{C}[-1]))$
    if $B \neq \perp$ **then** ▷ Successful Mining
        $\mathcal{C} \leftarrow \mathcal{C}||B$ ▷ Add block to current longest chain
        BROADCAST($\mathcal{C}$) ▷ Gossip protocol
    end if
    round $\leftarrow$ round$+1$
end function
function READ
    $x \leftarrow \epsilon$ ▷ Instantiate transactions
    for $B \in \mathcal{C}$ **do**
        $x \leftarrow x||B.x$ ▷ Extract all transactions from each block in the chain
    end for
    return x
end function

Algorithm 8 Mining

function $\text{POW}_{H,T,q}(x, s)$
  $ctr \xleftarrow{\$} \{0,1\}^{\kappa}$ ▷ Randomly sample Nonce
  for $i \leftarrow 1$ to q **do** ▷ Number of available queries per party
    $B \leftarrow s||x||ctr$ ▷ Create block
    if $H(B) \leq T$ **then** ▷ Successful Mining
      return B
    end if
    ctr $\leftarrow$ ctr $+1$
  end for
  return $\perp$ ▷ Unsuccessful Mining
end function

Algorithm 9 The longest chain rule

function $\text{MAXVALID}_{\mathcal{G},\delta(\cdot)}(\overline{C})$
  $C_{\mathsf{max}} \leftarrow [\mathcal{G}]$ ▷ Start with current adopted chain
  for $C \in \overline{C}$ **do** ▷ Iterate for every chain received through gossip network
    if $\mathsf{validate}_{\mathcal{G},\delta(\cdot)}(C) \wedge |C| > |C_{\mathsf{max}}|$ **then** ▷ Longest Chain Rule
      $C_{\mathsf{max}} \leftarrow C$
    end if
  end for
  return C_{max}
end function

Algorithm 10 Chain Validation

```
1: function VALIDATE_{G,δ(·)}(C)
2:     if C[0] ≠ G then                          ▷ Check that first block is Genesis
3:         return false
4:     end if
5:     st ← st_0                                  ▷ Start at Genesis state
6:     h ← H(C[0])
7:     st ← δ*(st, C[0].x)
8:     for B ∈ C[1:] do                           ▷ Iterate for every block in the chain
9:         (s, x, ctr) ← B
10:        if H(B) > T ∨ s ≠ h then               ▷ PoW check and Ancestry check
11:            return false
12:        end if
13:        st ← δ*(st, B.x)    ▷ Application Layer: update UTXO & validate transactions
14:        if st = ⊥ then
15:            return false                       ▷ Invalid state transition
16:        end if
17:        h ← H(B)
18:    end for
19:    return true
20: end function
```

Chain Virtues

1. **Common Prefix (k).** $\forall$ honest parties P_1, P_2 adopting chains C_1, C_2 at any rounds $r_1 \leq r_2$ respectively, $C_1[:-k] \preceq C_2$ holds.
2. **Chain Quality (μ, ℓ).** $\forall$ honest party P with adopted chain C, $\forall i$ any chunk $C[i : i+\ell]$ of length $\ell > 0$ has a ratio of honest blocks μ.
3. **Chain Growth (τ, s).** $\forall$ honest parties P and $\forall r_1, r_2$ with adopted chain C_1 at round r_1 and adopted chain C_2 at round $r_2 \geq r_1 + s$, it holds that $|C_2| \geq |C_1| + \tau s$.

Ledger Virtues

- **Safety:** For all honest parties P_1, P_2, and rounds r_1, r_2, $L_{r_1}^{P_1}$ is a prefix of $L_{r_2}^{P_2}$ or vice versa.
- **Liveness(u)**: If all honest parties attempt to inject a transaction tx at rounds $r, ..., r+u$, then for all honest parties P, tx will appear in L_{r+u}^{P}.