Blockchain Foundations
Final
2026

1. The exam has 5 questions with a total of 134 points (with 20 bonus points that can take you up to 154 points). All subproblems of a problems with multiple parts are equally weighted. You have 3 hours to take the exam, and a 15 minutes grace period for submissions. 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. You may also use artificial intelligence assistance.

5. This exam is an assessment. You agree to abide by the Stanford Honor Code. You must work on this exam alone, with no collaboration with your fellow students whatsoever. You may not discuss the exam questions and answers with any other human while the exam is ongoing and until the deadline for exam submission has elapsed. Any collaboration with other people, inside or outside the class, is a violation of the honor code.

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. The blockchain and cryptographic protocols we analyzed in class (hash functions, signatures, bitcoin backbone, etc.) can be broken:

   (a) By no adversaries at all.

   (b) By a polynomial probabilistic-time adversary, with non-negligible probability.

   (c) By a polynomial probabilistic-time adversary, with overwhelming probability.

   (d) By an exponential adversary, with probability 1.

   An exponential adversary can iterate through all possible nonces or grind hashable values to find collisions.

2. Consider a second-preimage-resistant hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$. How many collisions $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$ *exist*?

   (a) None.

   (b) One.

   (c) A polynomial number of collisions.

   (d) An infinite number of collisions.

   An infinite number of collisions must exist because due to the infinite size of the input space and finite size of the output space of H.

3. Which of the following adversaries would be detrimental to the existential unforgeability of a signature scheme?

   (a) An adversary who, given a tuple (sk, m, σ) of a secret key, message, and a signature such that $\text{Ver}(pk, m, \sigma)$, where pk is the respective public key, finds a new message $m' \neq m$ and signature σ' such that $\text{Ver}(pk, m', \sigma')$.

   (b) An adversary who, given a tuple (pk, m, σ) of a public key, message, and a signature such that $\text{Ver}(pk, m, \sigma)$, finds a new message $m' \neq m$ and signature σ' such that $\text{Ver}(pk, m', \sigma')$.

   (c) An adversary who, given a tuple (pk, m, σ) of a public key, message, and a signature such that $\text{Ver}(pk, m, \sigma)$, finds a new signature $\sigma' \neq \sigma$ such that $\text{Ver}(pk, m, \sigma')$.

   (d) None of the above.

To break the existential unforgeability game, the adversary needs to generate a new message and signature given the public key which is distinct from the message given by the oracle. Only answer choice (b) meets these criteria.

4. For a verifier who knows the index and value of a leaf, what is the size of a proof-of-inclusion in a binary Merkle tree built using the $\kappa = 256$ bit hash function blake2s with 1024 leaves?

   (a) 320 bytes
   (b) 2560 bytes
   (c) 352 bytes
   (d) 32768 bits

The Merkle tree will have height $\log_2 1024 = 10$. Hence, 10 hashes will be required in the proof-of-inclusion for a leaf. This corresponds to $10 \cdot 256 = 2560$ bits or 320 bytes.

5. What is the relationship between safety and liveness of ledgers?

   (a) A safe ledger is always live.
   (b) A live ledger is always safe.
   (c) A ledger can be both unsafe and unlive.
   (d) A ledger is safe if and only if it is live.

For example, in the proof-of-work protocol, a powerful adversary can maintain two longest chains comprised of blocks containing only adversarial transactions resulting in both a safety and liveness violation amongst the ledgers of honest parties.

6. There exists *no* PPT adversary controlling 70% of the mining power in a proof-of-work blockchain who can eventually break:

   (a) All parametrizations of Chain Quality, no matter what constant $\mu > 0$ and $\ell \in \mathbb{N}$ we choose.
   (b) All parametrizations of Common Prefix, no matter what constant $k \in \mathbb{N}$ we choose.
   (c) All parametrizations of Chain Growth, no matter what constant $\tau > 0$ and $s \in \mathbb{N}$ we choose.
   (d) All parametrizations of Liveness, no matter what constant $u \in \mathbb{N}$ we choose.

Chain growth must be satisfied for some parameters τ and s as long as there is at least one honest party.

7. At the moment of a reorg, transactions in the abandoned chain are:

   (a) Discarded.

   (b) All placed in the mempool.

   (c) Placed in the mempool, as long as they can be applied on top of the new chain.

   (d) Placed within the blocks of the newly adopted chain so that they can become immediately confirmed.

Reorgs append the transactions of the abandoned chain to the front of the existing mempool and apply them on top of the new chaintip state to determine if they should remain.

8. Waiting k blocks before confirmation enables us:

   (a) To use the *Chernoff Bound* to bound the probability of failure to negligible.

   (b) To use the *Law of Large Numbers* to bound the probability of failure to zero.

   (c) To ensure that the honest parties win, in expectation.

   (d) To use the *Pigeonhole Principle* to ensure that the proof-of-work hash function is well-behaved.

Waiting for k blocks makes it difficult for adversaries to override a confirmed transaction with a new chain because they would have to keep up with the honest parties for an extended period of time.

9. No transactions are occurring on a blockchain network. A rational party will:

   (a) Stop mining blocks until more transaction traffic appears.

   (b) Keep mining empty blocks to reap the coinbase reward.

   (c) Re-include some previously confirmed transactions into new blocks it mines.

   (d) Create its own high-fee-paying transactions, and include them, to reward itself.

Since the mempool will be empty, the mined blocks will be empty besides a coinbase transaction.

10. A protocol designer proposes replacing the proof-of-work inequality $H(B) \leq T$ with the inequality $H(B) \geq 2^\kappa - T$ everywhere.

   (a) This is fine.

(b) This will work, but will make the blockchain insecure, as the probability of a successful query is no longer $\frac{T}{2^\kappa}$ in the Random Oracle model.

(c) This does not make syntactic sense, as the hash $H(B)$ is always out of the designated range.

(d) This is fine, but the T parameter must be adjusted accordingly to the value $T' = 2^\kappa - T$.

The probability of a successful query will remain the same at $p = \frac{T}{2^\kappa}$.

11. The Weak Conservation Law in the UTXO model ensures:

(a) That money is never double spent.

(b) That money remains scarce.

(c) That a party only spends money which rightfully belongs to it.

(d) That transactions cannot be reverted.

The Weak Conservation Law ensures that the sum of outputs is at most the sum of the inputs, ensuring the total amount of spendable money is not increased.

12. If you increase the mining difficulty $\frac{1}{T}$, then:

(a) You increase liveness, but reduce safety

(b) You increase safety, but reduce liveness

(c) You increase both safety and liveness

(d) You reduce both safety and liveness

By increasing the mining difficulty, fewer valid blocks are produced, reducing liveness. Additionally, safety increases because more successful queries become convergence opportunities.

13. The Streamlet protocol:

(a) requires a honest supermajority to be safe because the quorum q has to be set at $2n/3$.

(b) can relax the honest supermajority safety conditon but at the expense of decreasing the liveness of the protocol.

(c) can relax the honest supermajority safety condition without decreasing the liveness of the protocol.

(d) can relax the honest supermajority safety condition and increase the liveness of the protocol.

If the honest supermajority condition is relaxed, then there may not be enough honest parties to reach the quorum. As a result, liveness will not hold.

14. An SPV client needs to bootstrap from genesis, and is interested in retrieving *all* coinbase transactions in a chain with length $|\mathcal{C}|$, each block of which has α transactions. It will need communication complexity:

   (a) $\mathcal{O}(|C| + \alpha)$

   (b) $\mathcal{O}(\alpha|C|)$

   (c) $\mathcal{O}(|C| + \log \alpha)$

   (d) $\mathcal{O}(|C| \log \alpha)$

For each coinbase transaction, $\log \alpha$ hashes will be required as part of the proof-of-inclusion. Hence, the communication complexity is $O(|C| \log \alpha)$

15. A majority adversary in the proof-of-work bitcoin backbone model:

   (a) Can break the bounds of the *Patience Lemma*, because she can break the collision-resistance property of the hash function and mine arbitrarily quickly.

   (b) Can break the bounds of the *Patience Lemma*, because she can simulate mining in the future without actually performing any work.

   (c) Is bound by the confines of the *Patience Lemma*, as long as the execution is typical.

   (d) Is bound by the confines of the *Patience Lemma*, even if the execution fails to be typical.

The proof of the Patience Lemma only mandates typicality of the execution.

16. The accounts-based model, as compared to the UTXO model, has:

   (a) A larger k parameter for Common Prefix.

   (b) A smaller k parameter for Common Prefix.

   (c) The same k parameter for Common Prefix.

   (d) The Common Prefix parameter set to $k = 1$.

The models pertain to the execution layer. The consensus layer does not change.

17. During peer discovery in a peer-to-peer protocol, the initial set of peers is:

   (a) Hard-coded into the code of the node as an array of IP addresses.

   (b) Discovered by collecting IP addresses from the issuers of blockchain transactions.

(c) Derived from the public keys in the coinbase transaction issued by the miners.

(d) Requested from the user's ISP via an HTTPS exchange.

In PSET 1, you hard coded a list of our bootstrapping peers to connect to upon booting for the very first time.

18. Setting the bitcoin backbone parameter ϵ, representing the acceptable Chernoff error, to 0 would cause:

(a) Liveness to be lost, because the Chernoff interval λ would need to become infinite.

(b) Blocks to be produced very slowly, as the block production rate f must also be reduced accordingly.

(c) The Common Prefix parameter to become very small, allowing for instant confirmation.

(d) None of the above.

As λ would have to be infinite to ensure "perfect" typicality and $k \geq 2\lambda f$, k would also have to be infinite. This makes confirmation very difficult, implying liveness would be lost.

19. In the UTXO model, a signature on the transaction ensures:

(a) That the transaction is not a double spend.

(b) That the transaction was issued by an honest party.

(c) That the spending party is the rightful owner of the output spent.

(d) That the money has been produced correctly according to the macroeconomic rules of the chain.

Signatures verify that money is being rightfully spent.

20. Consider two independent worlds A and B in which the variable-difficulty proof-of-work protocol is executed. In world A the adversary plays honestly. In world B, the adversary is a selfish miner. The worlds are the same (in terms of the parameters n, t, q, Δ and so on) otherwise.

(a) World A will tend to have a higher mining target T than world B.

(b) World B will tend to have a higher mining target T than world A.

(c) The two worlds will tend to have the same mining target T, as the total mining power devoted to the network is the same.

(d) The two worlds will tend to have the same mining target T, because the selfish mining attack is undetectable.

World A will have a greater block generation rate than World B because the adversary in World B isn't broadcasting their blocks. Hence, World A will have a lower (and more difficult) mining target.

21. In a well-configured proof-of-work system with $n = 10$, $t = 5$, an adversary playing honestly will cause the long-term chain quality to be:

   (a) 0

   (b) $\frac{1}{2}$

   (c) $\frac{1}{3}$

   (d) 1

The adversary and honest parties are equally powerful (i.e., $t = n - t = 5$). Hence, in the long-term, half the blocks of the longest chain will be honest and the other half will be adversarial.

22. Imagine a network for which suddenly, but temporarily, Δ becomes very large because the underwater Internet cable connecting the Americas to Europe is cut. During that temporary outage:

   (a) The proof-of-work protocol remains safe, but temporarily loses liveness.

   (b) The Streamlet protocol remains safe, but temporarily loses liveness.

   (c) The longest-chain proof-of-stake protocol remains safe, but temporarily loses liveness.

   (d) All of the above.

PoWLC and and PoSLC do not remain safe under large network delays since multiple blocks could be mined in the time other parties' blocks are received. However, Streamlet does remain safe, but may not have liveness due to the amount of time it would take to receive votes.

23. Imagine a network for which suddenly, but temporarily, half the parties went offline. The online parties are all honest. During this period:

   (a) The proof-of-work protocol remains safe, but temporarily loses liveness.

   (b) The Streamlet protocol remains safe, but temporarily loses liveness.

   (c) The longest-chain proof-of-stake protocol remains safe, but temporarily loses liveness.

   (d) All of the above.

PoWLC and PoSLC both have dynamic availability whereas Streamlet does not.

24. The size of the header of a block:

(a) Is a constant.
(b) Is proportional to the chain size $|\mathcal{C}|$.
(c) Is proportional to the number α of transactions in the block.
(d) Is proportional to the logarithm $\log \alpha$ of the number of transactions in the block.

Block headers consist of 2 hashes (previous header and Merkle tree root) and a nonce. Each of these have a constant length of κ bits.

25. A ξ-*superblock* is a block B that satisfies $H(B) \leq \frac{T}{2^\xi}$ for some $\xi \in \mathbb{N}$. What is the probability that a given block B is a ξ-superblock, given that it is already a good block?

(a) $\frac{1}{2^{\xi+\kappa}}$
(b) $\frac{1}{2^\xi}$
(c) $\frac{T}{2^\xi}$
(d) $1 - \frac{T}{2^{\xi+\kappa}}$

The desired probability is equivalent to the ratio of the probability of a ξ-superblock to the probability of a good block. Hence, our answer is

$$\frac{\frac{T}{2^{\xi+\kappa}}}{\frac{T}{2^\kappa}} = \frac{1}{2^\xi}$$

26. The proof-of-stake longest chain is not accountable because:

(a) Parties have no identities tied to the proof-of-stake puzzle.
(b) Safety is based on a synchrony assumption.
(c) Adversary parties cannot equivocate in the longest-chain protocol.
(d) All of the above.

If safety is violated, a violator can argue that they hadn't received a block due to delayed messages from synchrony not being upheld.

27. The proof-of-stake longest chain protocol:

(a) requires a honest supermajority to be safe because adversary can equivocate.
(b) requires a honest supermajority to be live because adversary can equivocate.
(c) None of the above statements are correct.
(d) Both of the above statements are correct.

PoSLC only requires honest majority for safety and liveness.

28. The backbone proof of the common prefix property of the proof-of-work longest chain protocol cannot be reused for the proof-of-stake longest chain protocol because:

    (a) the chain growth lemma doesn't hold anymore.

    (b) the expected number of honest blocks during an execution is not propertional to the honest stake.

    (c) the expected number of adversary blocks during an execution is not proportional to the adversary stake.

    (d) the Chernoff bound cannot be used to bound the number of honest blocks during an exection.

Due to the equivocation power of adversaries, they can broadcast multiple blocks from a single win of the lottery.

29. The confirmation latency of a longest chain protocol:

    (a) increases when the honest advantage increases.

    (b) increases when the security parameter increases.

    (c) increases when the network delay bound increases.

    (d) more than one of the above statements is correct.

Options (b) and (c) are both correct. Confirmation latency is linear in both the security parameter κ and network delay Δ.

30. The confirmation latency of Streamlet:

    (a) decreases when the honest advantage increases.

    (b) decreases when the security parameter increases.

    (c) decreases when the network delay bound increases.

    (d) more than one of the above statements is correct.

Having a higher honest advantage ensures that the finalization condition for Streamlet will be met more often. Moreover, the expected time until a honest leader gets elected is shortened.

(16 points) Problem 2

Consider a longest chain blockchain in the UTXO model. The protocol follows a confirmation rule with $k = 1$ (a transaction is considered *confirmed* if it is included in $\mathcal{C}[:-1]$). Consider the transaction graph in Figure 3. The coinbase reward is 50 units. The amount of each transaction's output is denoted as a number above it. Imaginary txids are denoted within each transaction circle. Note that transaction 7 has two outputs, transaction 10 has one input, transaction 10 has two outputs, transaction 11 has two inputs, and transaction 8 has one input.

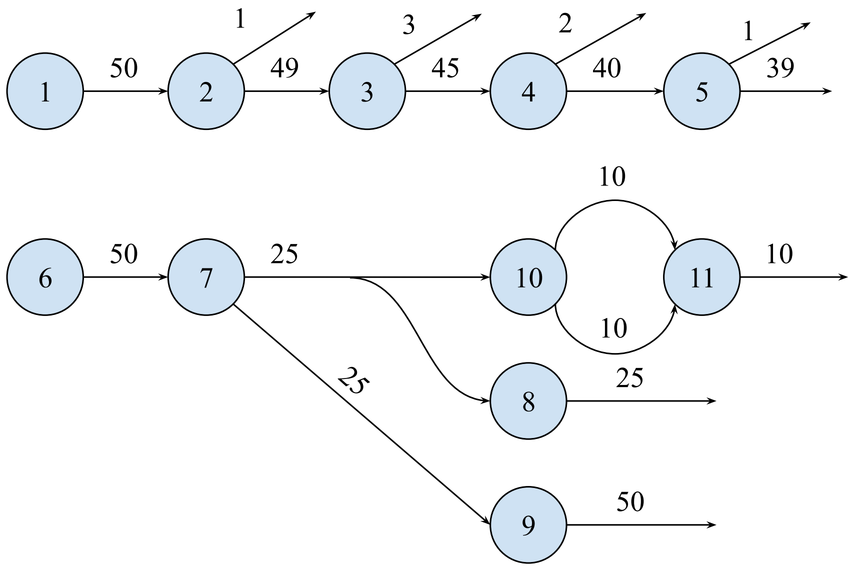

Figure 1: The transaction graph for Problem 2.

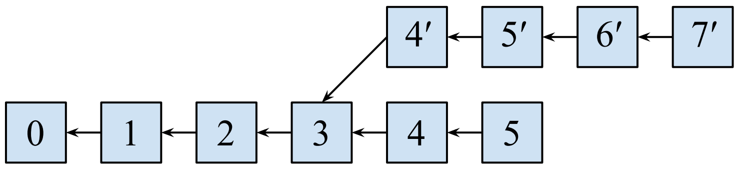

Figure 2: The block tree for Problem 2.

Consider the blocktree in Figure 2. In this system, blocks can be completely empty and coinbase transactions are not mandatory. The block contents are exactly the following:

- The genesis block 0 contains no transactions.
- Block 1 contains transaction 1.
- Block 2 contains transaction 6.

- Block 3 contains transactions $2, 3$.
- Block 4 contains transactions $4, 5$.
- Block 5 contains transactions $7, 8$.
- Block $4'$ contains transaction 7.
- Block $5'$ is empty.
- Block $6'$ contains transactions $10, 11$.
- Block $7'$ contains transactions $8, 5$.

Our timeline concerns an honest party P and is as follows:
Answer the following questions with regard to the timeline of a full node honest party P:

1. P has adopted the chain $[0, 1, 2]$ and has received transactions 1 and 6 by round 1. Subsequently, during round 1, he receives all transactions in the following order: $2, 3, 4, 5, 7, 8, 9, 10, 11$. What is his mempool at the end of round 1?

   The adopted chain starts us off with the coinbase transactions 1 and 6. To compute the mempool at the end of round 1, we must apply the received transactions in order to determine if they are valid. Transactions $2-5$ are valid and correspond to the upper half of the transaction graph. Transactions 7 and 8 are also valid. Transaction 9 violates the law of conservation. Transaction 10 is a double spend of the output of transaction 7. Transaction 11 is invalid due to transaction 10 being invalid. This leaves transactions $\boxed{2, 3, 4, 5, 7, 8}$ as the mempool.

2. What is his UTXO set (according to the mempool) at the end of round 1?

   Using the coinbase transactions and the mempool derived in previous part, we find that mempool UTXO is

   $$\boxed{\{(2,0),(3,0),(4,0),(5,0),(5,1),(7,1),(8,0)\}}$$

3. In round 2, he receives blocks 3, 4 and 5. What is his mempool at the end of round 2?

   The transactions in blocks 3, 4, and 5 are exactly the same as those in the mempool at the end of round 1. Hence, the mempool at the end of round 2 will be $\boxed{\text{empty}}$

4. What is his UTXO set (according to the mempool) at the end of round 2?

The mempool UTXO at the end of round 2 will be the same as the mempool UTXO at the end of round 1. This is because the mempool at the end of round 1 essentially got transferred to blocks 3, 4, and 5 on the longest chain during round 2, leaving the mempool UTXO unchanged as

$$\boxed{\{(2,0),(3,0),(4,0),(5,0),(5,1),(7,1),(8,0)\}}$$

5. In round 3 he receives blocks $4'$, $5'$, $6'$ and $7'$. What his chain at the end of round 3?

Blocks $4'$, $5'$, and $6'$ are successfully added on top of block 3. Block $7'$ is rejected because it contains transaction 8 which is a double spend of the outpoint $(7,0)$ already used by transaction 10 in block $6'$. Since the chain up to and including block $6'$ is longer than that of the chain up to and including 5, party P will adopt the chain consisting of blocks $\boxed{0,1,2,3,4',5',6'}$.

6. What is his mempool at the end of round 3?

A chain reorg is performed from the chain up to and including block 5 to the chain up to and including block $6'$ starting with an empty mempool. The latest common ancestor of blocks 5 and $6'$ is block 3. Hence, we must apply the transactions in blocks 4 and 5 (i.e., transactions $4,5,7,8$) on top of the state of the chain up to and including block $6'$. Transactions 4 and 5 will be valid. Transactions 7 and 8 will be rejected due to block $4'$ already containing transactions 7, and transaction 10 in block $6'$ already spending the input of transaction 8. Hence, party P's mempool at the end of round 3 will consist of transactions $\boxed{4,5}$

7. What is his UTXO set (according to the mempool) at the end of round 3?

Applying the mempool transactions 4 and 5 on top of blocks $0,1,2,3,4',5',6'$ will result in the UTXO set $\boxed{\{(2,0),(3,0),(4,0),(5,0),(5,1),(7,1),(11,0)\}}$

8. What is his ledger at the end of round 3?

The ledger will consist of the transactions in the blocks of the longest chain excluding the last block because $k=1$. In this case, those blocks are $0,1,2,3,4',5'$ consisting of the transactions $\boxed{1,6,2,3,7}$ in that order.

Use outpoints to designate the items in your UTXO set. Use a calculator such as Python and round your numbers to three significant digits.

(20 points) Problem 3

Consider the safety violation of Streamlet as shown in the figure below (the numbers within the blocks indicate their epochs). The upper fork has blocks notarized in three epochs $5, 6, 7$. The lower fork has a block from epoch 10 at the same height of the block from epoch 6.

1. Define what it means by the quorum of the Streamlet protocol.

   The quorum of the Streamlet protocol refers to the threshold of votes required to notarize a block.

2. Assuming the quorum of Streamlet is set to be $3n/4$. Argue that the safety violation cannot occur if the number of adversary parties is less than t, for the largest possible t you can provide an argument for.

   We claim that $t = \boxed{\frac{n}{2}}$ is the largest possible value for t. Let P be any honest party. If party P votes for block 6, then block 5 must be notarized at some point during epoch 6 in their view. Hence, party P won't vote for block 9 when they receive it during epoch 9. This is because block 9 would not be extending the longest notarized chain since block 5 is already notarized. As a result, any party who votes for both blocks 6 and 9 must be adversarial. To make the safety violation possible, block 9 has to be notarized which requires at least $2q - n = \frac{n}{2}$ adversarial double-voters.

3. Assuming the quorum of Streamlet is set to be $3n/4$. Argue that if the safety violation occurs, then the number of adversary parties that can be provably found to violate the protocol has to be at least t, and find the largest possible t.

   As argued in the previous part, if there is a safety violation, then there must be at least $2q - n = \frac{n}{2}$ double-voters. Since double-voters can be easily identified, $t = \boxed{\frac{n}{2}}$.

4. Assuming the quorum of Streamlet is set to be $3n/4$. What is the largest number of adversary parties for which liveness of the protocol can be guaranteed? Explain.

   The largest number of adversarial parties for which liveness is guaranteed is $n - q = \boxed{\frac{n}{4}}$. If there are more than $\frac{n}{4}$ adversarial parties, then they can all abstain from voting preventing blocks from being notarized. On the other hand, if there are at least $\frac{3}{4}n$ honest parties, then they are large enough to meet the quorum which guarantees liveness.

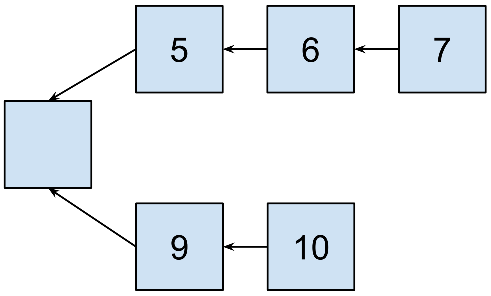

Figure 3: Safety violation. Upper fork has blocks from epochs $5, 6, 7$. Lower fork has block from epoch 10 at the same height as the block from epoch 6.

(18 points) Problem 4

In this class we studied three blockchain protocols: (i) proof-of-work longest chain, (ii) proof-of-stake longest chain, (iii) Streamlet. We have also introduced three protocol properties: (i) partition tolerance, (ii) dynamic availability, (iii) accountability.

1. Define the three protocol properties in terms of the basic properties of safety and liveness.

- **Partition tolerance:** Always safe but not necessarily live when the network is partitioned
- **Dynamic availability:** Liveness is guaranteed as long as the majority of online parties are honest
- **Accountability:** Whenever safety is violated, we can provably determine the violators of the protocol

2. Explain which of the three protocols satisfies which properties.

PoWLC:

- Doesn't satisfy partition tolerance since different partitions of the networks can grow their own independent chains.
- Satisfies dynamic availability due to lack of "voting thresholds."
- Fails to satisfy accountability since blocks cannot be traced to their miners in case of safety violation.

PoSLC:

- Doesn't satisfy partition tolerance for the same reason as PoWLC.
- Satisfies dynamic availability for the same reason as PoWLC.

- Fails to satisfy accountability since violating stakers can claim that synchronicity failed to hold causing them to build on top of the wrong block.

Streamlet:

- Satisifies partition tolerance due to a supermajority being required for notarization regardless of the number of partitions.
- Doesn't satisfy dynamic availability (e.g., if only $\frac{1}{2}n$ parties are online, then notarization can never be achieved)
- Satisfies accountability since double-voters can be caught.

3. Consider all the pairings of the three properties

   (a) partition tolerance and dynamic availability,

   (b) partition tolerance and accountability,

   (c) dynamic availability and accountability.

   For each pairing of the properties, argue either why it is impossible to have *any* protocol (not only the three we studied) that satisfies both properties, or give an example of a protocol that simultaneously achieves both properties.

(a) This pairing is impossible. Suppose the network is partitioned in half with each half having a honest majority. Assume for the sake of contradiction that partition tolerance and dynamic availability both hold. By dynamic availability, liveness will be guaranteed in each partition, so both partitions will grow their own possibly different chains. Eventually, this will result in a safety violation, implying partition tolerance cannot be satisfied.

(b) Streamlet satisfies both.

(c) This pairing is also impossible. Consider a similar setup as in part (a) with a partitioned network of only honest nodes. Assume for the sake of contradiction that dynamic availability and accountability both hold. Both partitions will grow their own chains until a safety violation occurs. However, nobody is accountable for the violation!

(20 points) Problem 5

Consider a bitcoin backbone execution with $n = 10$, $t = 2$, $q = 1000$, $f = 0.01$, and $\kappa = 256$.

1. Calculate the honest advantage δ for the given parametrization.

   Recall that
   $$t < (1-\delta)(n-t)$$
   must be satisfied by the honest advantage δ. In our case, this gives us
   $$2 < 8(1-\delta) \implies \delta < \boxed{\frac{3}{4}}$$

2. Calculate the probability p that a query is successful.

   Recall that
   $$f = 1 - (1-p)^{q(n-t)}$$
   gives the probability of a successful round. In our case, this gives us
   $$0.01 = 1 - (1-p)^{8000} \implies p = \boxed{1.26 \cdot 10^{-6}}$$

3. Calculate the target T.

   Recall that
   $$p = \frac{T}{2^\kappa}$$
   gives the probability of a successful query. In our case, this gives us
   $$1.26 \cdot 10^{-6} = \frac{T}{2^{256}}$$
   Solving for T results in a target belonging to the range $\boxed{\left[2^{236}, 2^{237}\right]}$

4. Use the balancing equation to calculate a reasonable ϵ.

   Recall that the balancing equation is
   $$3f + 3\epsilon < \delta$$
   In our case, this gives us
   $$3(0.01) + 3\epsilon < 0.75 \implies \epsilon < 0.24$$
   We can reasonably set $\epsilon = 0.2$

5. Suppose the probability of an execution being typical is $1-2^{-\left(\frac{\epsilon^2\lambda}{3}+\kappa-327\right)}$. Setting the acceptable probability of failure to 2^{-256}, calculate a λ that satisfies this.

We need the probability of success to be at least $1-2^{-256}$. Hence,

$$1-2^{-\left(\frac{\epsilon^2\lambda}{3}+\kappa-327\right)} \geq 1-2^{-256}$$

This is equivalent to

$$\frac{\epsilon^2\lambda}{3}+\kappa-327 \geq 256$$

Plugging in $\epsilon = 0.2$ and $\kappa = 256$ gives

$$\frac{(0.2)^2\lambda}{3}+256-327 \geq 256 \implies \lambda \geq 24525$$

6. What is the probability that round 5 is successful?

By definition of f, the probability that round 5 is successful is equal to $f = \boxed{0.01}$

7. What is the probability that round 5 is a convergence opportunity?

A convergence opportunity occurs when there is exactly one successful query amongst all $q(n-t)$ honest queries. Hence, by considering the relevant binomial distribution, our desired probability equals

$$\binom{q(n-t)}{1}p(1-p)^{q(n-t)-1} = 8000p(1-p)^{7999} = \boxed{0.00998}$$

8. What is the probability that round 5 has exactly two adversarial, but no honest, successful queries?

The probability of no honest successful queries is

$$(1-p)^{q(n-t)} = (1-p)^{8000} = 0.99$$

The probability of exactly two adversarial queries is, by considering the relevant binomial distribution with qt adversarial queries,

$$\binom{qt}{2}p^2(1-p)^{qt-2} = \binom{2000}{2}p^2(1-p)^{1998} = 3.17\cdot 10^{-6}$$

Hence, our final answer is

$$0.99(3.17\cdot 10^{-6}) = \boxed{3.14\cdot 10^{-6}}$$

9. What is a safe k for the confirmation rule?

We need for $k \geq 2\lambda f = 2(24525)(0.01) = 490.5$. Hence, we can reasonably set $k = 491$.

10. What is a parameter for the velocity τ that guarantees chain growth for intervals at least $s = \lambda$?

The minimum velocity τ is given by $(1 - \epsilon)f = 0.8(0.01) = \boxed{0.008}$ blocks per unit time.

Show the formula that you used for each calculation. Use a calculator such as Python and round your numbers to three significant digits.

(20 bonus points) Problem 6

Consider a *typical* Bitcoin Backbone execution correctly parameterized under honest majority in the static proof-of-work model.

1. An honest party P_1 broadcasts a new transaction tx_1 at round r_1 and another honest party P_2 broadcasts a new transaction tx_2 at round $r_2 \geq r_1$. Both transactions are included in the *same* block B, which eventually becomes stable and adopted by all honest parties. You correctly reason that, since the two transactions were both confirmed in the *same* block, they must have been broadcast closely together in time.

   Calculate an upper bound $d \in \mathbb{N}$ for the interval $r_2 - r_1$ such that we can rest assured that $r_2 - r_1 \leq d$. Your bound does not need to be tight, but needs to hold always for typical executions. Your bound can be a function of all the execution parameters $(\epsilon, \lambda, f, p, T, n, t, k, u, \mu, \ell, s, \tau)$. Prove that your bound holds.

   We claim that $d = \boxed{u}$ is a valid upper bound. For the sake of contradiction, suppose that $r_2 - r_1 > u$. By liveness, tx_1 will be mined and confirmed by the time tx_2 is broadcasted. Hence, tx_1 and tx_2 will not be included in the same block. This is a contradiction.

2. An honest party P_1 broadcasts a new transaction tx_1 at round r and an honest party P_2 broadcasts a new transaction tx_2 at the *same* round r. However, the transactions make it on different blocks B_1 with height h_1 and B_2 with height $h_2 > h_1$ of the same chain that eventually becomes stable and adopted by all honest parties. You correctly reason that, since the two transactions were both broadcast at the *same* time, they must be included close together on the chain.

   Calculate an upper bound $d \in \mathbb{N}$ for the distance $h_2 - h_1$ such that we can rest assured that $h_2 - h_1 \leq d$. Your bound does not need to be tight, but needs to hold always for typical executions. Your bound can be a function of all the execution parameters $(\epsilon, \lambda, f, p, T, n, t, k, u, \mu, \ell, s, \tau)$. Prove that your bound holds.

   We claim $d = \boxed{\ell}$. Suppose, towards a contradiction, that $h_2 - h_1 > l$. Then by Chain Quality, there must exist an honest block B within those ℓ blocks. The honest party who mined B must have seen both tx_1 and tx_2 at the time and therefore would have included the unconfirmed one.

   Alternative solution A. Suppose B_1 was mined at round r_1 and B_2 was mined at round r_2. It must hold $r_1 \geq r$ and $r_2 \geq r$, since tx_1 and tx_2 were not known prior to round r. Due to liveness, both transactions must make it to a stable block within u rounds of being broadcast, and Common Prefix guarantees that, once they make it into a stable block, this block will not be abandoned. Hence, $r \leq r_1 \leq r + u$ and $r \leq r_2 \leq r + u$, and hence $|r_1 - r_2| \leq u$. So the blocks connecting B_1 and B_2 must have been mined within fewer than u

rounds. We claim that $d = \boxed{\max(2fu+1, k)}$. Suppose, towards a contradiction, that $h_2 - h_1 > d$. Applying the Patience Lemma, the d blocks connecting B_1 and B_2 must have been computed in at least u rounds, a contradiction.

Alternative solution B. Let S be the set of rounds from round r to round $r + \max(\lambda, u)$. The height of the longest chain can grow by at most $X(S) + Z(S)$ during S. Typicality tells us that

$$\begin{aligned} X(S) + Z(S) &< \mathbb{E}[Z(S)] + \epsilon\mathbb{E}[X(S)] + (1+\epsilon)\mathbb{E}[X(S)] \\ &= \mathbb{E}[Z(S)] + (1+2\epsilon)\mathbb{E}[X(S)] \\ &= pqt|S| + (1+2\epsilon)f|S| \\ &= (pqt + f + 2\epsilon f)|S| \\ &= (pqt + f + 2\epsilon f)\max(\lambda, u) \end{aligned}$$

By liveness, we know that both transactions will be mined within u rounds and hence within S. Hence, $d = \boxed{(pqt + f + 2\epsilon f)\max(\lambda, u)}$ is a valid upper bound.

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: In proof-of-work, the hash rate of one party per round; in proof-of-stake, the quorum.
- 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 inequality: $3f + 3\epsilon \le \delta$.
- The proof-of-work inequality: $H(B) \le T$.
- The proof-of-stake inequality: $H(s_0 \,\|\, pk \,\|\, r) \le T_p$.

Security Definitions

Algorithm 1 The collision resistance game.

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

Definition 1 (Collision Resistant Hash Function)**.** *A hash function* $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ *is* collision resistant *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\mathit{collision}_{H,\mathcal{A}}(\kappa) = 1] < \mathrm{negl}(\kappa)$$

Algorithm 2 The preimage resistance game.

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

Definition 2 (Preimage Resistant Hash Function)**.** *A hash function* $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ *is* preimage resistant *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\mathit{preimage}_{H,\mathcal{A}}(\kappa) = 1] < \mathrm{negl}(\kappa)$$

Algorithm 3 The second preimage resistance game.

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

Definition 3 (Second Preimage Resistant Hash Function)**.** *A hash function* $H : \{0,1\}^* \longrightarrow \{0,1\}^\kappa$ *is* second preimage resistant *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\textit{2nd-preimage}_{H,\mathcal{A}}(\kappa) = 1] < \text{negl}(\kappa)$$

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

```
1: function existential-forgery-game$_{Gen,Sig,Ver,\mathcal{A}}(\kappa)$
2:     $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
3:     $M \leftarrow \emptyset$
4:     function $\mathcal{O}$(m)
5:         $M \leftarrow M \cup \{m\}$
6:         return $\mathsf{Sig}(sk, m)$
7:     end function
8:     $m, \sigma \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
9:     return $\mathsf{Ver}(pk, \sigma, m) \wedge m \notin M$
10: end function
```

Definition 4 (Existentially Unforgeable Signature Scheme)**.** *A signature scheme* Gen, Sig, Ver *is* existentially unforgeable *if for any PPT adversary* $\mathcal{A}$*:*

$$\Pr[\textit{existential-forgery-game}_{\text{Gen,Sig,Ver},\mathcal{A}}(\kappa) = 1] < \text{negl}(\kappa)$$

Algorithms

Algorithm 5 The Random Oracle

```
1: $r \leftarrow 0$
2: $\mathcal{T} \leftarrow \{\}$                              ▷ Initiate Cache
3: $Q \leftarrow 0$                 ▷ $q$ for honest parties, $qt$ for adversary
4: function $H_\kappa(x)$
5:     if $x \notin \mathcal{T}$ then                    ▷ First time being queried
6:         if $Q = 0$ then                            ▷ Out of Queries
7:             return $\perp$
8:         end if
9:         $Q \leftarrow Q - 1$
10:        $\mathcal{T}[x] \xleftarrow{\$} \{0,1\}^\kappa$
11:    end if
12:    return $\mathcal{T}[x]$                        ▷ Return value from Cache
13: end function
```

Algorithm 6 The environment.

```
1: $r \leftarrow 0$
2: function $\mathcal{Z}^{n,t}_{\Pi,\mathcal{A}}(1^\kappa)$
3:     $\mathcal{G} \xleftarrow{\$} \{0,1\}^\kappa$                                   ▷ Genesis block
4:     for $i \leftarrow 1$ to $n-t$ do                            ▷ Boot stateful honest parties
5:         $P_i \leftarrow \mathsf{new}\ \ \Pi(\mathcal{G})$
6:     end for
7:     $A \leftarrow \mathsf{new}\ \ \mathcal{A}(\mathcal{G},n,t)$                           ▷ Boot stateful adversary
8:     $\overline{M} \leftarrow []$                                    ▷ 2D array of messages
9:     for $i \leftarrow 1$ to $n-t$ do
10:        $\overline{M}[i] \leftarrow []$                  ▷ Each honest party has an array of messages
11:    end for
12:    while $r < \mathsf{poly}(\kappa)$ do                                   ▷ Number of rounds
13:        $r \leftarrow r+1$
14:        $M \leftarrow \emptyset$
15:        for $i \leftarrow 1$ to $n-t$ do                 ▷ Execute honest party $i$ for round $r$
16:            $Q \leftarrow q$      ▷ Maximum number of oracle queries per honest party (Section 2)
17:            $M \leftarrow M \cup \{P_i.\mathsf{execute}^H(\overline{M}[i])\}$          ▷ Adversary collects all messages
18:        end for
19:        $Q \leftarrow tq$                              ▷ Max number of Adversarial oracle queries
20:        $\overline{M} \leftarrow A.\mathsf{execute}^H(M)$                ▷ Execute rushing adversary for round $r$
21:        for $m \in M$ do                            ▷ Ensure all parties will receive message $m$
22:            for $i \leftarrow 1$ to $n-t$ do
23:                $\mathsf{assert}(m \in \overline{M}[i])$                        ▷ Non-eclipsing assumption
24:            end for
25:        end for
26:    end while
27: end function
```

Algorithm 7 The honest party

```
1:  $\mathcal{G} \leftarrow \epsilon$
2:  function CONSTRUCTOR($\mathcal{G}'$)
3:      $\mathcal{G} \leftarrow \mathcal{G}'$                                    ▷ Select Genesis Block
4:      $\mathcal{C} \leftarrow [\mathcal{G}]$                                  ▷ Add Genesis Block to start of chain
5:      round $\leftarrow 1$
6:  end function
7:  function EXECUTE($1^\kappa$)
8:      $\tilde{\mathcal{C}} \leftarrow \text{maxvalid}(\mathcal{C}, \bar{M}[i])$              ▷ Adopt Longest Chain in the network
9:      if $\tilde{\mathcal{C}} \neq \mathcal{C}$ then
10:         $\mathcal{C} \leftarrow \tilde{\mathcal{C}}$
11:         BROADCAST($\mathcal{C}$)                              ▷ Gossip Protocol
12:     end if
13:     $x \leftarrow$ INPUT()                                    ▷ Take all transactions in mempool
14:     $B \leftarrow$ POW($x, H(\mathcal{C}[-1])$)
15:     if $B \neq \perp$ then                                  ▷ Successful Mining
16:         $\mathcal{C} \leftarrow \mathcal{C}||B$                               ▷ Add block to current longest chain
17:         BROADCAST($\mathcal{C}$)                              ▷ Gossip protocol
18:     end if
19:     round $\leftarrow$ round$+1$
20: end function
21: function READ
22:     $x \leftarrow \epsilon$                                         ▷ Instantiate transactions
23:     for $B \in \mathcal{C}$ do
24:         $x \leftarrow x||B.x$                  ▷ Extract all transactions from each block in the chain
25:     end for
26:     return $x$
27: end function
```

Algorithm 8 Mining

```
1: function $\textsc{pow}_{H,T,q}(x, s)$
2:     $ctr \xleftarrow{\$} \{0,1\}^{\kappa}$                                        ▷ Randomly sample Nonce
3:     for $i \leftarrow 1$ to $q$ do                          ▷ Number of available queries per party
4:         $B \leftarrow s||x||ctr$                                                   ▷ Create block
5:         if $H(B) \leq T$ then                                              ▷ Successful Mining
6:             return $B$
7:         end if
8:         $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$
9:     end for
10:    return $\perp$                                                     ▷ Unsuccessful Mining
11: end function
```

Algorithm 9 The longest chain rule

```
1: function $\textsc{maxvalid}_{\mathcal{G},\delta(\cdot)}(\overline{C})$
2:     $C_{\mathsf{max}} \leftarrow [\mathcal{G}]$                               ▷ Start with current adopted chain
3:     for $C \in \overline{C}$ do              ▷ Iterate for every chain received through gossip network
4:         if $\mathsf{validate}_{\mathcal{G},\delta(\cdot)}(C) \wedge |C| > |C_{\mathsf{max}}|$ then               ▷ Longest Chain Rule
5:             $C_{\mathsf{max}} \leftarrow C$
6:         end if
7:     end for
8:     return $C_{\mathsf{max}}$
9: end function
```

Algorithm 10 Chain Validation

```
function VALIDATE_{$\mathcal{G},\delta(\cdot)$}($C$)
    if $C[0] \neq \mathcal{G}$ then                         ▷ Check that first block is Genesis
        return false
    end if
    st $\leftarrow$ st$_0$                                   ▷ Start at Genesis state
    $h \leftarrow H(C[0])$
    st $\leftarrow \delta^*(st, C[0].x)$
    for $B \in C[1{:}]$ do                                   ▷ Iterate for every block in the chain
        $(s, x, ctr) \leftarrow B$
        if $H(B) > T \vee s \neq h$ then                     ▷ PoW check and Ancestry check
            return false
        end if
        st $\leftarrow \delta^*(st, B.x)$     ▷ Application Layer: update UTXO & validate transactions
        if $st = \perp$ then
            return false                                     ▷ Invalid state transition
        end if
        $h \leftarrow H(B)$
    end for
    return true
end function
```

Chain Virtues

1. **Common Prefix ($k \in \mathbb{N}$).** $\forall$ honest parties P_1, P_2 adopting chains $\mathcal{C}_1, \mathcal{C}_2$ at any rounds $r_1 \leq r_2$ respectively, $\mathcal{C}_1[:-k] \preceq \mathcal{C}_2$ holds.

2. **Chain Quality ($\mu \in [0,1], \ell \in \mathbb{N}$).** $\forall$ honest party P with adopted chain $\mathcal{C}$, $\forall i$ any chunk $\mathcal{C}[i{:}i+\ell]$ of length $\ell > 0$ has a ratio of honest blocks μ.

3. **Chain Growth ($\tau \in \mathbb{R}^+, s \in \mathbb{N}$).** $\forall$ honest party P, $\forall r_1, r_2$ with adopted chain C_1 at round r_1 and adopted chain $\mathcal{C}_2$ at round $r_2 \geq r_1 + s$, it holds that $|\mathcal{C}_2| \geq |\mathcal{C}_1| + \tau s$.

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}.

Theorems

Lemma 5 (Patience Lemma)**.** *In typical executions, any $k \geq 2\lambda f$ blocks have been computed in at least $\frac{k}{2f}$ rounds.*