EE374 Blockchain Infrastructure **Stanford, Spring 2025**

Midterm

Due: Tues, May-13-2025, 4:30pm by submitting to Gradescope (`R57ZN7`).

IMPORTANT – Insert your name and email:

Instructions

1. There are 7 questions in this exam, and the total number of points is 104. The total number of pages is 17.
2. For your convenience, a full description of the Tendermint protocol is included in p. 16 and 17 of the exam.
3. Midterm is due at 4:30pm Tues and Nusret will go over the solutions in the discussion section at that time.
4. The exam is open-book, but you may not get help or communicate with others on its content.
5. Please sign below to confirm that you agree to Stanford's Honor Code.

1. (30 points) True or False? 2 points for a correct answer. 0 point for an incorrect answer. 1 point for leaving blank.

   a) Nakamoto invented the first consensus protocol.

   b) Nakamoto invented the longest chain protocol.

   c) The attack that Nakamoto worried most about is that an attacker with more than 50% of the hash power can create bitcoins from thin air.

   d) The "no-forgery" property of a digital signature scheme guarantees that an adversary who does not know the private key cannot present a valid signature on a message if it has not seen a valid signature by the challenger on another message, but provides no guarantee if the adversary has already seen one such signature by the challenger.

   e) The execution state maintained by Bitcoin is set of all UTXOs.

   f) Alice has 2 bitcoins. In one transaction, she sends 1.5 bitcoins to Bob and 0.5 bitcoins back to herself. The size of the UTXO set increased by 2.

   g) Costless long range attacks on Proof-of-stake blockchains are not possible in Proof-of-work blockchains like Bitcoin.

   h) Compared to the Tendermint protocol, the Ethereum Proof-of-Stake protocol has two ledgers instead of one.

   i) A hash function that satisfies the collision resistant property also satisfies the random oracle property.

j) If a hash function H satisfies the collision resistant property, then the hash function G defined by $G(x) = H(H(x))$ for all x in the domain of H also satisfies the collision resistant property.

k) The largest fraction of blocks that an adversary can get into the longest chain in the Bitcoin protocol is equal to the fraction of hash power the adversary has, assuming the network delay is 0.

l) In order for a consensus protocol to be live, the chain quality should be greater than or equal to 50%.

m) If there is a safety attack on Ethereum Proof-of-Stake protocol, it is guaranteed that 1/3 of the total amount of ETH staked will be slashed.

n) Nakamoto's goal of inventing Bitcoin is to replace "trust me bro" security by "trust only math" security.

o) All Bitcoin transactions have at least one input and at least one output.

Answer:

a) F

b) T

c) F

d) F

e) T

f) F

g) T

h) T

i) F

j) T

k) F

l) F

m) F

n) F

o) F

---

2. (5 points) Two main consensus protocols discussed in the course are the Proof-of-Work longest chain protocol and the Proof-of-Stake Tendermint protocol. Identify the key differences in security properties and performance of these two consensus protocols.

Answer:

	PoW Longest Chain	**PoS Tendermint**
security resilience	1/2 of hash power	1/3 of stake
accountable safety	no	yes
long range attack security	yes	no
latency	slow (hours)	fast (seconds)
energy consumption	high	low
dynamic availability	yes	no
finality (safety under asynchrony)	no	yes

---

3. (14 points) In May 2021, China banned Bitcoin mining.

   a) (2 points) From publicly available data on the Internet, estimate the total Bitcoin hash rate on May 12, 2021, exactly 4 years ago. This was before the ban.

Answer:

It is 173.8595 EH/s.

---

   b) (2 points) From publicly available data on the Internet, estimate the total Bitcoin hash rate on June 12, 2021, 1 month later. This was after the ban.

Answer:

It is 126.4049 EH/s.

---

   c) (4 points) If the average Bitcoin block time was 10 minutes on May 12, 2021, what would be the average Bitcoin block time on June 12, 2021, assuming that the Proof-of-work difficulty threshold was not adjusted? What would the impact of this block time have on the security and performance of the Bitcoin protocol?

Answer:

The average block time would be $10 \times 173.8595/126.4049 \approx 13.75$ minutes. This would imply a larger latency and less throughput. It might conditionally imply an improvement or degradation of security. It would imply an improvement of the security resilience, as less hash power (and the same difficulty and network delay Δ) would result in less forking. It might imply a degradation of security, if the drop in hash power affected only the honest miners.

---

d) (6 points) To avoid the block time from drifting away from the target 10 minutes, the Bitcoin protocol contains a difficulty adjustment mechanism: every epoch contains 2024 blocks, and the difficulty threshold is adjusted at the beginning of each epoch as follows. Let

$$G := \frac{2023 \cdot 10}{T_{\text{last}} - T_{\text{first}}},$$

where T_{first} and T_{last} are the timestamps (in minutes) of the first block and the last block of the previous epoch respectively. Then the difficulty threshold τ_{new} for mining the blocks in the new epoch is adjusted from the difficulty threshold τ_{old} for mining the blocks in the previous epoch by the formula:

$$\tau_{\text{new}} = \begin{cases} G\tau_{\text{old}} & \text{if } \frac{1}{4} < G < 4 \\ 4\tau_{\text{old}} & \text{if } G \geq 4 \\ \frac{1}{4}\tau_{\text{old}} & \text{if } G \leq \frac{1}{4} \end{cases}$$

Assuming a new epoch starts on June 12 and the total hash rate remains constant throughout this epoch, when will the next difficulty adjustment occurs, and what will the average block time after that adjustment? Justify your answer carefully. You can assume that the timestamps in the blocks are accurate.

Answer:

It will take on average $2023 \times 10 \times 173.8595/126.4049 \approx 19.3$ days (or 27824.7 minutes) for 2024 blocks to be mined. Therefore, for these blocks,

$$G = \frac{2023 \times 10}{T_{\text{last}} - T_{\text{first}}} \approx 0.727.$$

Then, the new difficulty will be $\tau_{\text{new}} = 0.727 \times \tau_{\text{old}}$ (as $1/4 < 0.727 < 4$), and the new average block time after the adjustment will become 10 minutes per block.

---

4. (14 points) In class, when we discuss the safety and liveness of consensus protocols, we assume that a client can download all the blocks that are made public by the honest validators/miners. In practice, a late-joining client has to download the blocks by querying other peers. For this problem, we will consider a scenario where a late-joining client queries k neighboring peers. Honest peers are assumed to send their chains and the corresponding certificates (if exist in the protocol), while dishonest peers can send anything or nothing at all. The client computes its ledger from the data it receives from all the peers. Assuming that the honest peers' ledgers are safe and live, we would like to explore the conditions under which the computed ledger of the late-joining client is safe and live.

   a) (6 points) First, consider Bitcoin's longest chain protocol. State how a late-joining client should compute its ledger based on the downloaded chains from its k peers. What is the minimal trust assumption on the k peers to guarantee the safety of the late-joining client's ledger? What is the minimal trust assumption on the k peers to guarantee the liveness of the late-joining client's ledger?

Answer:

A late-joining client should select the longest chain of blocks with the correct headers among the provided chains, remove the last k blocks, and output the sequence of transactions in the remaining chain as its ledger. At least one of the k peers must be honest for the safety of the late-joining client's ledger (otherwise, it can be fooled into accepting a short chain conflicting with the longest chain observed by the other clients). Similarly, at least one of the k peers must be honest for the liveness of the late-joining client's ledger.

---

   b) (6 points) Repeat part (a) for the Tendermint protocol.

Answer:

The late-joining client should select the longest chain of blocks (among the provided chains) such that there are $2f + 1$ (or over $2/3$) pre-commits by the validator set for each block. It should then output the sequence of transactions in this chain as its ledger. *None* of the k peers need to be honest for the safety of the late-joining client's ledger. However, at least one of the k peers must be honest for the liveness of the late-joining client's ledger.

---

c) (2 points) Compare (a) and (b), and provide an explanation for the difference(s) if there area any.

Answer:

The main difference is in the condition for safety, and the reason for this difference is that Bitcoin lacks cryptographic certificates to prove finality externally, whereas Tendermint explicitly uses certificates created from pre-commit votes of validators to establish finality. In the case of Bitcoin, at least one of the k peers must be honest for the safety of the late-joining client's ledger; otherwise, the client can be fooled into accepting a short chain conflicting with the longest chain observed by the other clients. In Tendermint, even if there is no honest node among the peers, the adversarial nodes cannot convince the client to output a block that was not confirmed, since no such block would have a valid certificate, i.e., $2f + 1$ pre-commits.

---

5. (11 points) In this question, we analyze Nakamoto's private attack and the selfish mining attack under network delay. The adversary has a fraction $\beta < 1/2$ of the mining power, the honest and adversarial miners mine at rates λ_h and λ_a blocks per second respectively, and the network delay is assumed to be Δ (seconds). We assume that each single honest miner contributes only an infinitesimally small fraction of the honest hash power (but there are many honest miners in total).

   In the first part of this question, we focus on the private attack.

   a) (3 points) Express β in terms of λ_h and λ_a. When $\Delta = 0$, under what conditions does the private attack succeed (with non-negligible probability), or fail (except with negligible probability)? Write down the upper bound on β for the attack to fail.

Answer:

The adversary's mining rate is λ_a, and the honest mining rate is λ_h. β is the fraction of the total mining rate that the adversary has. Hence;

$$\beta = \frac{\lambda_a}{\lambda_a + \lambda_h}.$$

The private attack succeeds with non-negligible probability, if $\lambda_a \geq \lambda_h$. The private attack fails except with negligible probability, if $\lambda_a < \lambda_h$. The upper bound on β is $\beta < 1/2$.

---

   b) (4 points) Now, suppose $\Delta > 0$, and the adversary ensures that every new block mined by an honest miner reaches all the other honest miners exactly Δ time after it is mined. Then, what would be the expected time for the honest chain to grow by one level? Explain. (Here, the honest chain refers to the public chain mined by the honest miners without adversarial participation.)

   Hint: Recall that when $\Delta = 0$, the expected time for the honest chain to grow by one level was $1/\lambda_h$.

Answer:

In this case, when a new block B is mined at a new height h of the honest chain, that block is observed by the other honest miners after Δ time. During this Δ period, all other miners still try to extend the block at height $h-1$. Since each honest miner contributes only an infinitesimally small fraction of the total honest hash power, during this Δ period, the miner that created B would not be able to extend its own block and reach height $h+1$ either.

Now, once the Δ period elapses, all honest miners observe B and try to extend it. They will then mine a new block at height $h+1$ in expected $1/\lambda_h$ time. Therefore, the expected time for the honest chain to grow by one level becomes $\Delta + (1/\lambda_h)$.

---

c) (4 points) Given your answer above, when $\Delta > 0$, under what condition, does the private attack succeed (with non-negligible probability), or fail (except with negligibile probability)? Write down the upper bound on β for the attack to fail in terms of Δ. What happens to the upper bound as $\Delta \to 0$ and as $\Delta \to \infty$?

Answer:

By part (b), the growth rate of the honest chain becomes $\lambda_h/(1+\lambda_h\Delta)$. The private attack succeeds with non-negligible probability, if $\lambda_a \geq \lambda_h/(1+\lambda_h\Delta)$. The private attack fails except with negligible probability, if $\lambda_a < \lambda_h/(1+\lambda_h\Delta)$. Hence, the upper bound becomes $\frac{2+\lambda\Delta-\sqrt{4+(\lambda\Delta)^2}}{2\lambda\Delta}$. It approaches $1/2$ as $\Delta \to 0$ and approaches 0 as $\Delta \to \infty$.

---

6. (10 points) In class, we defined **resilience** of a consensus protocol as the maximum number of nodes controlled by the adversary such that the protocol satisfies safety and liveness (except with negligible probability). Recognizing that safety and liveness are of different importance, we refine this definition. Let **safety resilience** of a consensus protocol be the maximum number of nodes controlled by the adversary such that the protocol satisfies safety (except with negligible probability). Similarly, let **liveness resilience** of a consensus protocol be the maximum number of nodes controlled by the adversary such that the protocol satisfies liveness (except with negligible probability).

   a) (2 points) Let $n = 3f + 1$ denote the total number of nodes. Recall that we set the quorum size for Tendermint to $q = 2f + 1$ in the class. In this case, what are the safety and liveness resiliences?

Answer:

Both the safety and liveness resiliences are f.

---

   b) (4 points) Given a quorum size of $q \in [0, 3f + 1]$, what would the safety and liveness resiliences be as a function of q? Plot how safety (y-axis) and liveness (x-axis) resiliences vary against each other as q varies.

Answer:

Safety resilience would be $f_s = 2q - n - 1$, and the liveness resilience would be $f_\ell = n - q$. The plot is that of the $y = n - 2x - 1$.

---

   c) (4 points) Given a quorum size of $q \in [0, 3f + 1]$, what would be the *accountable safety* and liveness resiliences corresponding to q (suppose Tendermint does have accountable safety for this question)? Plot how accountable safety (y-axis) and liveness (x-axis) resiliences vary against each other as q varies.

Answer:

The accountable safety resilience would be $f_s = 2q - n$, and the liveness resilience would be $f_\ell = n - q$.

---

7. (20 points) In this question, we analyze the accountable safety of Tendermint with a quorum size of $q = 2f + 1$. In the questions below, we do not make any assumptions about the number of honest nodes, unless otherwise stated.

   a) (8 points) Suppose two different blocks B_0 and B_2 from rounds 0 and 2 are confirmed by clients c_0 and c_2 respectively. Block B_0 was proposed via the message $\langle \mathsf{Proposal}, r = 0, vr = -1, B_0 \rangle$, and block B_2 was proposed via the message $\langle \mathsf{Proposal}, r = 2, vr = 1, B_2 \rangle$. Then, can c_0 and c_2 come together and identify at least $f + 1$ nodes that violated the protocol rules? If so, write down the proof. If not, argue why.

   Hint: This question is very similar to question 1-c in homework 5; the only difference is that here, we consider the original confirmation rule of Tendermint.

Answer:

The answer is no. Note that c_0 must have observed $2f + 1$ round-0 pre-commits for B_0, and c_2 must have observed $2f + 1$ round-2 pre-commits for B_2. A node that sent round-2 pre-commits for B_2 is either adversarial, or must have observed $2f + 1$ round-2 pre-votes for B_2. Suppose the latter case is true. Then, since the $2f + 1$ nodes that sent the round-0 pre-commits for B_0 must have locked on B_0 at round 0, by quorum intersection, at least $f + 1$ nodes (denoted by the set S_2) must have

locked on B_0 at round 0, yet sent a round-2 pre-vote for $B_2 \neq B_0$. Now, either these $f+1$ nodes are adversarial, or they must have observed $2f+1$ round-1 pre-votes for block B_2. Therefore, again by quorum intersection, at least $f+1$ nodes (denoted by the set S_1) must have locked on B_0 at round 0, yet sent a round-1 pre-vote for $B_2 \neq B_0$, which is a protocol violation. In all of the cases above, at least $f+1$ nodes are adversarial. However, these adversarial nodes cannot be identified by c_0 and c_2; since c_0 and c_2 do not necessarily observe the round-1 or round-2 pre-votes for B_2 (they only observe the $2f+1$ round-0 and round-2 pre-commits for B_0 and B_1 respectively).

---

b) (8 points) Now, suppose we know *for a fact* that in any given set of $2f+1$ nodes, there will be at least one honest node. Then, in the question above, can c_0 and c_2 come together and identify at least $f+1$ nodes that violated the protocol rules? If so, write down the proof. If not, argue why.

Hint: If c_0 and c_1 ask an honest node what it knows about the pre-votes from the current or the past rounds, the honest node will provide whatever information it has to c_0 and c_1.

Answer:

The answer is yes. Note that c_0 must have observed $2f + 1$ round-0 pre-commits for B_0, and c_2 must have observed $2f + 1$ round-2 pre-commits for B_2. Out of the $2f + 1$ nodes that sent the round-2 pre-commits for B_2 (call this set S_{c2}), at least one is honest, and must have observed $2f + 1$ round-2 pre-votes for B_2. Similarly, out of the $2f + 1$ nodes that sent the round-2 pre-votes for B_2 (call this set S_{p2}), at least one is honest, and must have observed $2f + 1$ round-1 pre-votes for B_2. Then, since the $2f + 1$ nodes that sent the round-0 pre-commits for B_0 must have locked on B_0 at round 0, by quorum intersection, at least $f + 1$ nodes (call this set S_1) must have locked on B_0 at round 0, yet sent a round-1 pre-vote for $B_2 \neq B_0$, which is a protocol violation.

Now, c_1 and c_2 can ask the nodes in S_{c2} to reveal the $2f + 1$ round-2 pre-votes for B_2. The honest node within S_{c2} will then reveal these round-2 pre-votes, i.e., the set S_{p2}. Then, c_1 and c_2 can ask the nodes in S_{p2} to reveal the $2f + 1$ round-1 pre-votes for B_2. The honest node within S_{p2} will then reveal these round-1 pre-votes. Finally, by intersecting the $2f + 1$ round-0 pre-commits and the $2f + 1$ round-1 pre-votes, c_0 and c_1 can identify the nodes responsible for the safety violation.

---

c) (4 points) Now, suppose we know *for a fact* that x number of the nodes are honest. What is the minimum value for x so that in any given set of $2f + 1$ nodes, there will be at least one honest node?

Answer:

The answer is $f + 1$. When there are $f + 1$ honest nodes, any set of $2f + 1$ nodes must contain at least one honest node (by quorum intersection).

Tendermint description (for a set of $n = 3f+1$ nodes, and quorum size of $q = 2f+1$):

***

Tendermint attempts to confirm a single block per height. Within each height h, it proceeds in *rounds*, each with a unique, known leader that proposes a block. Each of these rounds attempts to confirm a block for the height h. Round structure of Tendermint is described below:
At each round $r = 0, 1, 2, \ldots$, each honest node keeps track of the *step* of the protocol within the round. It can be one of proposal, pre-vote, or pre-commit. Each step lasts Δ time, and thus, each round lasts 3Δ time.

At the beginning of the **proposal step** (time $t = 3\Delta r$), an honest leader proposes a round-r block B by sending a proposal message $\langle \mathsf{Proposal}, r, vr, B \rangle$. We will later see what vr refers to – it can be -1 or a positive integer denoting some round.

Let $\langle \mathsf{Proposal}, r, vr, B \rangle$ be the first round-r proposal message observed by an honest node. At the beginning of the **pre-vote step** (time $t = 3\Delta r + \Delta$), this honest node sends a round-r pre-vote, denoted by $\langle \mathsf{Prevote}, r, vr, h(B) \rangle$, for block B (vr is copied from the proposal message) if the following conditions are satisfied:

- When $vr = -1$, the node can directly send the pre-vote.
- When $vr > 0$, the honest node will send the pre-vote only if it has also observed $2f + 1$ round-vr pre-votes for B, i.e., $2f + 1$ messages of the sort $\langle \mathsf{Prevote}, vr, vr', h(B) \rangle$ (here, vr' does not matter).

If an honest node does not observe a round-r block it can vote for, it sends a round-r pre-vote, denoted as $\langle \mathsf{Prevote}, r, vr = -1, \perp \rangle$, for a special *nil* (empty) value.

At the beginning of the **pre-commit step** (time $t = 3\Delta r + 2\Delta$), each honest node sends a round-r pre-commit, denoted by $\langle \mathsf{Precommit}, r, h(B) \rangle$, for the round-$r$ block B, for which it has first observed $2f + 1$ round-r pre-votes of the form $\langle \mathsf{Prevote}, r, vr, h(B) \rangle$ by distinct nodes. If an honest node does not observe such $2f + 1$ pre-votes for any round-r block, it sends a round-r pre-commit, denoted as $\langle \mathsf{Precommit}, r, \perp \rangle$, for a special *nil* value.

Confirmation Rule: At the end of the round ($t = 3\Delta r + 3\Delta$) or later, if an honest node observes $2f + 1$ round-r pre-commits for B, it confirms B for its height (h), and terminates the protocol for height h. Otherwise, it goes into the next round $r + 1$.
A client c confirms a round-r block B, if it observes (at any time) $2f + 1$ round-r pre-commits for B by distinct nodes. In that case, we say that the round-r block B became confirmed by the round-r pre-commits.

Proposal Rule: Let r' be the largest round such that the node has observed $2f + 1$ round-r' pre-votes for a round-r' block B'. Then, if that node is elected as a leader in a future round r'', it proposes this block B' for round r'' by sending the message $\langle \mathsf{Proposal}, r'', vr = r', B' \rangle$. If there is no such round r', i.e., if the node has not observed $2f + 1$ (same-round) pre-votes for any block B', it can propose any block B'' by sending the message $\langle \mathsf{Proposal}, r'', vr = -1, B'' \rangle$.

Locking Rule: An honest node v locks on a round-r block B at round r upon sending a round-r pre-commit for B. In a future round r'', the node v does not send pre-votes for other blocks $B'' \neq B$ **unless the block** B'' **comes as part of a proposal** $\langle \mathsf{Proposal}, r'', vr = r', B'' \rangle$ **and** v **observes** $2f + 1$ **round** $vr = r'$ **pre-votes for** B''. *Note that an honest node never releases a lock permenantly: it can acquire a lock at a higher round in the future, or it might temporarily drop the lock to vote for block* $B'' \neq B$ *in the example above.*

***