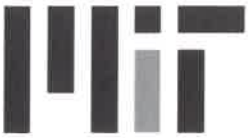


Department of Electrical Engineering and Computer Science
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.1800 Computer Systems Engineering: Spring 2025

Quiz II

There are 12 questions and 12 pages in this quiz booklet. Answer each question according to the instructions given. You have two hours to answer the questions.

- The questions are organized loosely by topic. They are not ordered by difficulty nor by the number of points they are worth.
- **If you find a question ambiguous, write down any assumptions you make.** Be neat and legible.
- You are not required to explain your answers unless we have explicitly asked for an explanation. You may include an explanation with any answer for possible partial credit.
- Some students will be taking a make-up exam at a later date. **Do not** discuss this exam with anyone who has not already taken it.
- Write your name and kerberos ID in the space below. Write your initials at the bottom of each page.

This is an open-book, open-notes, open-laptop exam, but you may **NOT** use your laptop, or any other device, for communication with any other entity (person or machine).

Turn all network devices, including your phone, off.

Name: SOLUTIONS

Kerberos ID: solutions **@mit.edu**

I. GFS and ZFS

1. [12 points] Martin has found a list of claims about file systems. Answer True or False for each of them:

   A. **True / False** In GFS, chunk servers don't know whether the file name space is consistent.

   B. **True / False** In ZFS, the SPA doesn't know whether the file name space is consistent.

   C. **True / False** In GFS, the controller doesn't know whether the file name space is consistent.

   D. **True / False** In ZFS, the ZPL doesn't know whether the file name space is consistent.

   E. **True / False** If we wanted to use ZFS as the storage layer of GFS, we could take advantage of the Linux-like interface of ZPL.

   F. **True / False** If we wanted to use GFS as the storage layer of ZFS, we could take advantage of the vdev-like capabilities of GFS.

A. The namespace is managed by the controller largely and abstracted away from the chunkservers.

B. The DMU handles this, on top of the SPA.

C. See explanation for part A.

D. See explanation for part B, & note that the ZPL sits on top of the DMU

E. A posix layer gives us a Linux-like interface

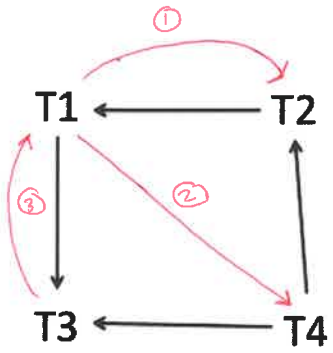
F. GFS does not have vdev-like capabilities

Initials:

II. Transactions

2. [10 points] Emeka has the following (complete) schedule and (possibly incomplete) conflict graph for transactions T1, T2, T3, T4. For the schedule, time increases as you move down the page.

T1	T2	T3	T4
write(y, 0)			
write(z, 0)			
write(x, 10)			
		read(x)	
			write(y, 20)
		read(y)	
	read(y)		
	write(z, 30)		
read(z)			
write(x, 20)			
			read(x)



① Conflict between write(y, 0) in T1 & read(y) in T2
② Conflict between write(y, 0) in T1 & write(y, 20) in T4
③ Conflict between read(x) in T3 & write(x, 20) in T1.

A. Add arrows to the diagram above so that it is complete. If you believe the diagram is already complete, circle "Already complete" below.

Already complete

B. Identify a SINGLE action in the above schedule that can be removed to make this schedule of transactions conflict-serializable.

One approach is to remove each action, recompute the conflict graph, and see if it's acyclic.

More efficiently: T1 is part of 3 separate cycles. If we want an acyclic graph by removing only one action, it must involve T1. And removing write(x, 10) still leaves a cycle between T1 & T2 (and the T1→T4→T2→T1 cycle)

a. T3: read(x)
b. T2: read(y)
c. T1: write(x,10)
d. T4: write(y, 20)
e. T2: write(z, 30)
f. There is no single action that can be removed to produce a conflict-serializable schedule
g. The schedule is already conflict-serializable

Initials:

III. Replicated State Machines

3. [9 7 points] Cathy is dealing with a replicated state machine implemented by N+1 replicas. There is one primary P and N backups: B_1, B_2, .., B_N. However, there is no view server.

Instead, this replicated state machine uses the IDs of the backups as an ordering to determine who becomes the next primary in case of failures. Specifically: if P fails, everyone in the system has agreed that B_1 will be the next primary. If B_1 fails, B_2 will become primary, and so on through B_N.

Assume that there is an efficient heartbeat protocol that allows any replica to check the status of any other replica by sending a request and receiving a corresponding reply. The effect is that every replica knows instantaneously when some other replica is not responding. Further assume that each replica is either "up" (fully functioning, responding to messages) or "down" (not responding to messages), so you do not need to be concerned about partial failure of a replica or recovery of a replica.

We'll call it a *partition* whenever there is any impairment of replica-to-replica communication. One particular kind of partition is an *asymmetry in message delivery*: that's a partition where some replica A can't send to some other replica B, even though B can send to A.

Answer True or False to each of the following completions of the claim that "This scheme ensures that there is always **at most one** replica acting as primary **out of the entire set of replicas**…"

A. **True** / **False** …in any network that is guaranteed to have no partitions.

B. **True** / **False** …even in a network that allows partitions, as long as there are no asymmetries in message delivery.

C. **True** / **False** … even in a network that has asymmetries in message delivery.

B. Consider this partition, with no asymmetries in message delivery:

P	B_1
B_2	B_3
B_4	B_5
⋮	⋮

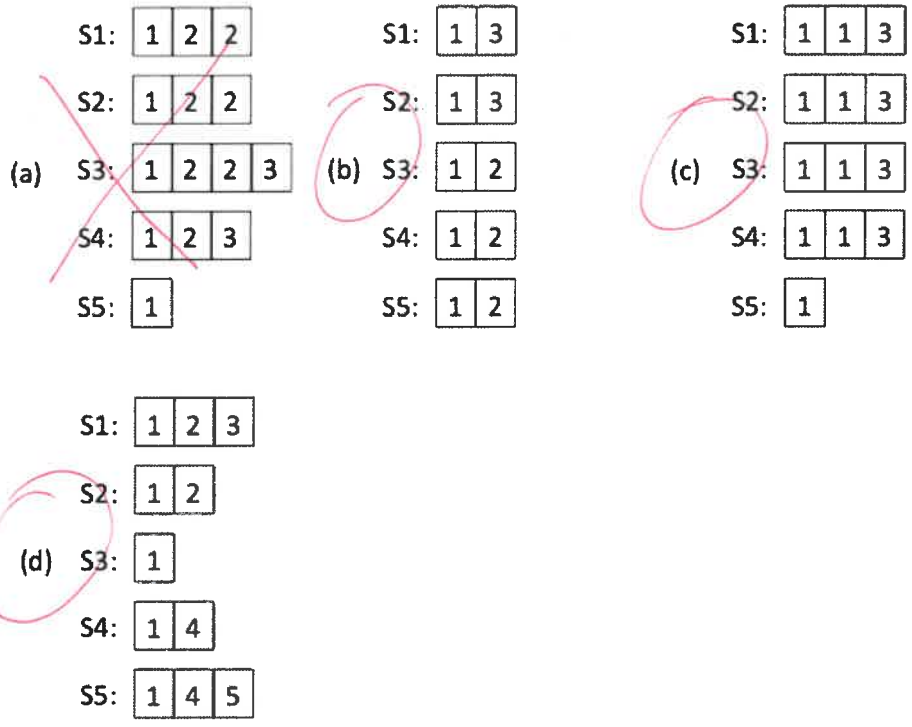
On the left side, P is primary.
On the right side, B_1 is primary

C. If B is false, then C must also be false.

Initials:

IV. Raft

4. [12 points] Umang has the following diagrams, showing the logs at five servers S1 through S5. Assume that the five servers are implementing five Raft replicas of a single replicated state. As in the paper, the log entries have been abstracted to show term numbers only.



Circle each diagram that is legal in Raft (if any) and draw an X through each diagram that is not legal in Raft (if any). Any diagram that is neither circled nor X'ed will be marked incorrect.

(a) S4 is missing the second entry for term 2 but has the entry for term 3. The only way to align the logs would be for that "hole" in S4's log to get filled in, but Raft doesn't allow holes.
Alternate line of reasoning: Either S3 or S4 is leader in term 3. If it's S3, then S4's log is wrong, and vice versa.

(b) Possible scenario: S1, S2 crash after entry for term 1. S5 elected leader, logs entry for term 2. S3 crashes before receiving entry for term 2. S4, S5 crash; S1, S2, S3 recover. S1 elected leader. S1, S2 log entry for term 3, S3 crashes before receiving entry for term 3. S1, S2 crash; S3, S4, S5 recover. S5 elected leader. S5 updates S3's log to match S5's.

(c) S5 got first update in term 1 & then failed. Second update in term 1 was fine. Leader in term 1 failed, new one was elected for term 2 but no updates. That leader fails, original leader back online & elected, & then previous leader back online. S1-S4 get term 3 update

(d) Possible scenario: S1 leader in terms 1-3, updates not yet out to all servers. Then S1, S2 fail. S5 leader in terms 4-5, similar sequence.

Initials:

V. Concurrency control, logging, and recovery

5. [6 points] Ricardo has the following excerpt from a simple write-ahead log. In this system, there is no cache and no cell storage.

   UPDATE records are of the form:

   UPDATE <var name>=<old value>; <var name>=<new value>.

   After entry 41, the system crashed.

Entry #	Trans ID	Entry
30	111	BEGIN
31	111	UPDATE A=0; A=1
32	111	UPDATE B=0; B=1
33	111	COMMIT
34	112	BEGIN
35	112	UPDATE A=1; A=2
36	112	UPDATE B=1; B=3
37	113	BEGIN
38	113	UPDATE C=0; C=1
39	113	UPDATE A=2; A=0
40	113	COMMIT
41	112	UPDATE B=3; B=4

   A. During the database's recovery process, which transaction(s) get undone? Circle one or more of the transaction IDs, or circle None if you believe none are undone.

   None 111 112 113

   111 & 113 both committed; 112 did not.

   B. **True / False** This log excerpt is consistent with a system that uses strict 2-phase locking (holds all locks through the commit point).

   For line 35 to execute, transaction 112 must have A's lock. But transaction 113 would not be able to make the update in line 39 if we're using 2PL, b/c transaction 112 still has A's lock.

VI. Two-phase commit

6. [7 points] Aileen is working on a system where servers S1, S2 have modified data on behalf of distributed transaction T. Server C is acting as the coordinator for the transaction commit. The servers may fail, and they communicate via a network where both network failures and partitions are possible.

   In contrast to the system described in lecture, there are no timeouts. Accordingly, when a remote server is not responding there is no reliable way of determining whether it has crashed or is up but unreachable. The transaction T has reached its end and is trying to commit. The following actions have happened as part of the two-phase commit:

   i. C writes "prepare" to its log
   ii. C tells S1 to prepare
   iii. C tells S2 to prepare
   iv. S2 writes changes and "prepared" to its log
   v. S1 writes changes and "prepared" to its log
   vi. S1 tells C that S1 has prepared
   vii. S2 tells C that S2 has prepared
   viii. C writes "commit" to its log

Immediately after step viii, C crashes. Answer the next two questions assuming that C has not yet recovered. Pick the BEST response.

A. Can server S1 commit?

S1 does not know about this

   a. Yes, because C has committed
   b. No, because S1 does not know the outcome
   c. Yes, because both S1 and S2 have prepared
   d. No, because any failure causes an abort

B. Can server S2 abort?

   a. Yes, because C has crashed
   b. No, because S2 does not know the outcome
   c. No, because both S1 and S2 have prepared
   d. Yes, because any failure causes an abort

C. Now server C recovers and re-establishes communication with servers S1 and S2. Can server S1 commit now?

In fact S1 must commit, b/c C has written COMMIT to its log.

   a. Yes, after C informs S1 that the outcome is commit
   b. No, because the transaction already committed while C was crashed
   c. No, because the transaction already aborted while C was crashed
   d. No, because now the transaction aborts although it couldn't abort before

Initials:

VII. Passwords

In this, and all future questions, "|" indicates string concatenation.

7. [7 points] Phoebe has been asked to evaluate the following schemes for storing username and password information. Each scheme is described by a single line containing the fields that are needed for a single user's login lookup; the table consists of many such lines.

   The different fields in the line are separated by commas (field1, field2, field3). "salt" is a randomly generated 10-digit number. Assume there are no issues with the random number generator. The hash function is "slow" – it takes a substantial amount of computation.

   The chosen scheme(s) must defend against a so-called "rainbow table" attack, in which the adversary has already computed hashes of many common passwords.

   For each scheme, circle the correct answer to the question: Is this scheme resistant to rainbow table attacks?

   A. **Yes** / **No** username, salt, hash(password) | hash(salt)

   B. **Yes** / **No** username, salt, hash(password) | hash(password | salt)

   C. **Yes** / **No** username, salt, hash(hash(password) | salt)

A. Salt is not helping here, because we have h(password) | h(salt), not h(password|salt)

B. Even with h(password|salt) included, we also have h(password), which the adversary could look up in the table

C. The inner hash is unnecessary, but does not make the scheme less secure

Initials:

VIII. Authentication

Katrina wants to distribute her software via her web site. She wants to ensure that her customers get accurate versions, despite the ongoing presence of attackers who might change packets on the network. Conveniently, the software consists of a single file. Your job is to give Katrina advice about two different authentication schemes she is contemplating.

The first table shows information related to Katrina's choices. V is a version number for a release; R_v is the release file for version V; K_i is a secret key that is shared by Katrina and customer i. $T_{v,i}$ is the tag computed for version V and customer i.

	Katrina calls customer with	Release file on Katrina's website	Hash or tag on Katrina's website
Scheme 1	*(nothing)*	R_v	$H_v \leftarrow \mathrm{H}(V \mid R_v)$
Scheme 2	K_i	R_v	$T_{v,i} \leftarrow \mathrm{MAC}(V \mid R_v, K_i)$

The second table shows information related to customer actions. V is a version number for a release; R_v' is the received release file for version V; H_v' is the received hash for version V; $T_{v,i}'$ is the received tag for version V and customer i; K_i is a secret key that is shared by Katrina and customer i.

	Key from Katrina	Received file	Received hash or tag	Customer checks
Scheme 1	*(none)*	R_v'	H_v'	$H_v' = \mathrm{H}(V \mid R_v')$?
Scheme 2	K_i	R_v'	$T_{v,i}'$	$T_{v,i}' = \mathrm{MAC}(V \mid R_V', K_i)$?

Scheme 1 calculates a hash of the content of each release and puts the hash along with the release on the web server. Katrina uses a cryptographic hash function H to generate H_v as indicated in the first table.

She tells her customers to check the received file by computing the hash value on the received file, then comparing that to the received hash, as indicated in the second table.

Katrina does all her development, compiling, and computing of hash values on her laptop, which only she has access to and which she uses for no other purpose.

Initials:

We are omitting the details of exactly how releases and hashes are named; you can assume that customers successfully navigate Katrina's web site to find the release(s) and hash(es) they seek.

8. [4 points] For each of the following statements about Katrina's ***Scheme 1***, indicate whether it is true or false.

   A. **True / False** If an attacker can change (***only***) the R_v' received, a customer might accept a release even though it was different from what Katrina intended. The attacker would need to change the hash (H_v) as well

   B. **True / False** If an attacker can read and modify files on the web server, a customer might accept a release even though it was different from what Katrina intended. In this way they could change R_v & H_v, because the hash function doesn't require a key

Scheme 2 generates authentication tags for each software version using a shared secret message authentication code (MAC). Katrina also generates a separate key K_i for each of her customers and contacts each customer on the telephone to give them their K_i, as indicated in the first table. For each new release V and each customer i, Katrina calculates an authentication tag $T_{v,i}$ also as indicated in the first table. Then Katrina puts all the tags and the associated release on the web server.

She tells her customers to check the received file by checking it against the received authentication tag, as indicated in the second table.

Once again, Katrina does all her development, compiling and computing of MAC values on his laptop, and she stores the K_i keys only on her laptop. Only Katrina has access to this laptop, and she uses it for no other purpose.

Again, we are omitting the details of exactly how releases and MACs are named; you can assume that customers successfully navigate Katrina's web site to find the release(s) and MAC(s) they seek.

9. [4 points] For each of the following statements about Katrina's ***Scheme 2***, indicate whether it is true or false.

   A. **True / False** If an attacker can change (***only***) the R_v' received, a customer might accept a release even though it was different from what Katrina intended. Same reasoning as 9A

   B. **True / False** If an attacker can read and modify files on Katrina's server, a customer might accept a release even though it was different from what Katrina intended. The Attacker doesn't have K_i and so can't recompute the MAC

Initials:

IX. Secure Channels

10. [10 points] David has built his own secure channel between two servers A and B. He has an efficient, reliable, and secure encryption engine that can take 512 cleartext bytes and a secret key to produce 512 encrypted bytes. It can also take 512 encrypted bytes and a (matching) secret key to produce the original 512 cleartext bytes.

    David's applications always send each other messages that are exactly 512 bytes long, via some sensible and workable but otherwise-unremarkable transport protocol. His implementation has one secret key X that's used for traffic going from A to B, and a different secret key Y for traffic going from B to A. The secret keys X and Y are both "good enough" (long enough, random enough) and are simply hardcoded into the servers A and B.

    Assume a threat model in which the adversary can see any traffic on the network, can create messages on the network, and can destroy messages on the network. However, the adversary cannot compromise server A or server B, and cannot determine any secret key (X or Y) from any number of collected messages.

    A. **True** / **False** David's channel may be vulnerable to a replay attack by the adversary No sequence numbers

    B. **True** / **False** David's channel may be vulnerable to a reflection attack by the adversary Different keys in each direction

    C. **True** / **False** David's channel may be vulnerable to the adversary pretending to be A or B. Key exchange was done safely & out-of-band

    D. **True** / **False** David's channel is vulnerable to the adversary reading the application data (cleartext) of messages being sent between A and B.

    E. **True** / **False** The adversary may be able to infer something about David's application, or his use of the application, from the communication pattern between A and B.

X. Botnets, Blockchain, Bitcoin

11. [9 points] Jorge has come up with an idea to combine botnets and blockchain. He figures that the current running Bitcoin system can be used as a coordination mechanism between bots and his C&C server. Each new bot will post its availability in a transaction on the Bitcoin blockchain so the C&C server can see it. Likewise, the C&C server will post commands for each of the bots in the same blockchain.

    Assume that Jorge has written all the code for the bots and the C&C server.

    Answer True or False for each of these claims about the resulting system.

    A. **True / False** If Jorge's coordination scheme depends on his bots adding blocks directly (mining), it may be slow and unpredictable.

    B. **True / False** If Jorge's scheme depends on sending transactions via other miners, he (or someone else) will have to pay them.

    C. **True / False** A long consistent public history is important whenever a bot is trying to coordinate with a C&C server.

XI. Cryptosystem failure

12. [8 points] Mark has found a list of claims about cryptography and cryptosystems. Answer True or False for each of them:

    A. **True / False** Weak cryptographic algorithms are the primary problem with the security of systems deployed in the real world.

    B. **True / False** Back-door capabilities are a risk to security.

    C. **True / False** Secret algorithms and secret implementations always strengthen the security of a system.

    D. **True / False** Trustworthy components assembled together guarantee a secure system.