CS 426 - Introduction to Blockchains
End term Exam
Duration: 3 Hours

Instructions:
1. *There are 10 questions in this question paper. All questions are compulsory.*
2. *For each question, please justify how you arrived at your answer. Incomplete answers will lead to loss of marks.*
3. *Answer each question using the information provided in the question (no clarifications during the exam).*
4. *Assumptions made in each question should be clearly stated. The assumption should be rational and reasonably accurate. Wild assumptions should not be considered for arriving at an answer.*

Q1. Choose the right option(s) (more than one answer is possible, marks will be awarded if all the correct choices corresponding to a given question are selected. No justification is required here) [14 Marks]

i. Which of the following could be considered as stake in a PoS consensus:
   a. CPU and GPU resource
   b. Number of blocks successfully mined
   c. Cumulative Rewards earned
   d. Cumulative Rewards spent

Ans: c

ii. BLAKE2 and RIPEMD are ________
   a. Public key cryptography schemes
   b. Consensus protocols
   c. Hash functions
   d. Cryptocurrencies

Ans: c

iii. Alice intends to send 10 BTCs (bitcoins) to Bob. Alice and Bob will use _______ and ________ keys to sign and verify the transactions respectively.

   a. Public key of Bob, Private key of Bob
   b. Private key of Bob, Public key of Bob
   c. Private key of Alice, Public key of Alice
   d. Public key of Alice, Private key of Alice

Ans: c

iv. If values of prime integer numbers p and q are chosen to be 7 and 19 respectively, and the value of "e" (one of the keys) is chosen to be 29, then the value of the other key "d" as per RSA algorithm is ______

   a. 41
   b. 39
   c. 23
   d. 47

Ans: a

iv. Public blockchains give an incentive to encourage users to mine blocks and secure the network. What incentive is this?
   a. Public blockchains allow users to create tokens to sell on secondary markets.

b. Public blockchains do not offer rewards, because they are open source.
c. Public blockchains offer cash rewards for running mining nodes.
d. Public blockchains offer rewards for mining in the form of cryptocurrency.

Ans: d

v. Which characteristic of a blockchain network is also its protection?

a. The greater the number of full independent nodes, the harder it is to compromise the data in the blockchain.
b. The lower the number of miners in the blockchain, the higher the incentive is for securing the network.
c. The more centralized the control of the blockchain is, the harder it is to secure the data and avoid fraud.
d. The more complicated the Proof of Work (PoW) algorithm is, the more rewarding it is to secure the network.

Ans: a

vi. An attacker tried to corrupt the transaction history of a blockchain to be able to spend a token or a cryptocurrency twice. What is the most likely thing this attacker did?

a. The attacker changed the transaction on his node and propagated it in the network.
b. The attacker edited/changed the smart contract and recovered the investor's cryptocurrency.
c. The attacker gained control of more than 51% of the network's computing power.
d. The attacker hard-forked the network and created a new blockchain network.

Ans: c

vii. A competitive consensus algorithm that was developed because blockchains had difficulty meeting the transaction speed demands. Which consensus algorithm is this?

a. Delegated Proof of Stake (DPoS)
b. Proof of Burn
c. Proof of Stake (PoS)
d. Proof of Work (PoW)

Ans: c

viii. Which description fits only the Proof of Work (PoW) consensus algorithm?

a. A collaborative consensus algorithm, where approved accounts do the validation.
b. A low-cost and fast algorithm, where a node needs to deposit cryptocurrency to guarantee the transaction.
c. A noncompetitive consensus algorithm, where validation is done by elected nodes, which send cryptocurrency to an address, from which it cannot be retrieved.
d. An intensive and expensive, competitive algorithm where each node on the blockchain is competing to secure blocks

Ans: d

ix. Consider the following function written in solidity programming language.

```
        unit result = 0; // result is a state variable declared outside the function
getResult()
        function getResult() public view returns(uint){

                uint a = 1; // local variable
                uint b = 2; // local variable
                result = a + b;
                return result;
        }
```

What happens when you compile and execute/call the function?

a. The value of 'result' is 3

b. The value of ‘result’ is 0
c. Compilation Error
d. No Output

Ans: c

x. ________________ allows users to manage private keys and sign transactions from the web browser by integrating a web browser application with the local Ethereum blockchain environment.

a. Truffle
b. Metamask
c. Web3
d. Geth

Ans: b

xi. The output of the truffle compile command can be seen in ______ folder and the output files are seen in _________ format.

a. Contracts folder and . sol format
b. Migrations folder and .js format
c. Test folder and .js format
d. Build folder and .js format

Ans: d

xii. Which of the following prerequisite software package should be installed in the system before you install truffle framework:

a. Java runtime environment
b. Nodejs
c. C#
d. .NET framework

Ans: b

xiii. A cryptographic hash function exhibits which of the following properties:

a. Deterministic
b. Reversible
c. Collision Resistance
d. All of the above

Ans: (a) and (c)

xiv. Ethereum account addresses are ___ bytes and the private keys are ____ bytes.

a. 48, 32
b. 64, 20
c. 20, 32
d. 24, 48

Ans: c

Q2. Give the final contents of the hash table that results when you insert items with the keys E A S Y Q U T I O N in that order into an initially empty table of M = 5 lists, using standard chaining with unordered lists. Use the hash function *11 k mod M* to transform the *k*th letter of the alphabet into a table index, e.g., hash(I) = hash(9) = 99 % 5 = 4.

[3 Marks]

Ans: To create the hash table, we'll insert each letter in the given order using the hash function 11 * k mod M, where k is the position of the letter in the alphabet and M = 5.

a. E: hash(E) = hash(5) = 11 * 5 % 5 = 55 % 5 = 0. Hash table now looks like:
   0: E
   1:
   2:
   3:
   4:
b. A: hash(A) = hash(1) = 11 * 1 % 5 = 11 % 5 = 1. Hash table now looks like:
   0: E
   1: A
   2:
   3:
   4:
c. S: hash(S) = hash(19) = 11 * 19 % 5 = 209 % 5 = 4. Hash table now looks like:
   0: E
   1: A
   2:
   3:
   4: S
d. Y: hash(Y) = hash(25) = 11 * 25 % 5 = 275 % 5 = 0. Hash table now looks like:
   0: E -> Y
   1: A
   2:
   3:
   4: S
e. Q: hash(Q) = hash(17) = 11 * 17 % 5 = 187 % 5 = 2. Hash table now looks like:
   0: E -> Y
   1: A
   2: Q
   3:
   4: S
f. U: hash(U) = hash(21) = 11 * 21 % 5 = 231 % 5 = 1. Hash table now looks like:
   0: E -> Y
   1: A -> U
   2: Q
   3:
   4: S
g. T: hash(T) = hash(20) = 11 * 20 % 5 = 220 % 5 = 0. Hash table now looks like:
   0: E -> Y -> T
   1: A -> U
   2: Q
   3:
   4: S
h. I: hash(I) = hash(9) = 11 * 9 % 5 = 99 % 5 = 4. Hash table now looks like:
   0: E -> Y -> T
   1: A -> U

2: Q
3:
4: S -> I

i. O: hash(O) = hash(15) = 11 * 15 % 5 = 165 % 5 = 0. Hash table now looks like:
0: E -> Y -> T -> O
1: A -> U
2: Q
3:
4: S -> I

j. N: hash(N) = hash(14) = 11 * 14 % 5 = 154 % 5 = 4. Hash table now looks like:
0: E -> Y -> T -> O
1: A -> U
2: Q
3:
4: S -> I -> N

So, the resulting hash table with standard chaining and unordered lists is:
0: E -> Y -> T -> O
1: A -> U
2: Q
3:
4: S -> I -> N

Q3. Answer the following: [2 Marks]

a. What is the significance of the option “--reset” in the command - “truffle migrate --reset”
b. contains the following configuration parameters. What does the parameter network_id represent in the truffle-config.js file?

Ans:
a. “truffle migrate” will run all migrations located within your project's migrations directory. If your migrations were previously run successfully, truffle migrate will start execution from the last migration that was run, running only newly created migrations. If no new migrations exist, truffle migrate won't perform any action at all.

You can use the --reset option to run all your migrations from the beginning.

b. Ethereum networks are groups of connected computers that communicate using the Ethereum protocol. There is only one Ethereum Mainnet, but independent networks conforming to the same protocol rules can be created for testing and development purposes. There are many independent "networks" that conform to the protocol without interacting with each other. You can even start one locally on your own computer for testing your smart contracts and web3 apps.
The testnet networks are used by protocol developers or smart contract developers to test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet.

Network_id refers to the ID of these testnet networks. Example: network_id of Sepolia is 11155111, goerli is 5.

https://ethereum.org/en/developers/docs/networks/

Q4. Consider the hash cash protocol proposed by Adam Back in 1997 for preventing email spammers who spam or execute DoS attacks on an email server. The hash-cash protocol adds a hash-cash stamp to the email header as shown in the figure below. Essentially, the stamp is created by appending the version, date, destination email address, and challenge details (such as number of zeros at the beginning). It is then passed to the SHA-1 function, to generate corresponding hash output which has the required number of zeros at the beginning as described in the challenge. In the process, the nonce value generated is noted down and is finally appended at the end of the stamp and sent to the server. The stamp is verified by passing the stamp along with the nonce value to SHA - 1 and getting the hash output with the required number of zeros. If everything is fine, then the sender is most likely not a spammer.
The following is an example of an email addressed to Adam with a hashcash stamp in the email headers:

```
From: Someone <test@test.invalid>
To: Adam Back <adam@cypherspace.org>
Subject: test hashcash
Date: Thu, 26 Jun 2003 11:59:59 +0000
X-Hashcash: 0:030626:adam@cypherspace.org:20:6470e06d773e05a8

<< email body or content containing the original message >>

-Someone
```

Answer the following questions: [2+2+2 = 6 Marks]

(a) Why are hash functions used to create the hash-cash stamps? Alternatively, why can't we use a modulo or a similar math function to create hash-cash stamps in this particular scenario.
(b) Explain how such hash-cash stamps can be used to demotivate the spammers to send junk emails.
(c) Can you think of an application of hash-cash protocol in the context of Blockchains. Elaborate on the application and show its connection to hash-cash protocol.

Ans:

a. Hash functions are used for this purpose because of two reasons:
   1. It makes solving the challenge difficult because of the puzzle friendliness property. This means finding a particular value of nonce such that when it is appended with the rest of the stamp satisfies the given requirement is a difficult problem to solve.
   2. It makes verification of the challenge easy. This means once the nonce value has been found out for a given challenge, verification of that nonce to be the right answer to the challenge is a trivial process and does not require much time to verify.

b. The idea is that we want to make it moderately difficult for the spammers to send spam emails to the server. In order to do so, we ask spammers to solve a challenge which is to find a nonce that satisfies this required property such that hashing the whole hash stamp together, including that nonce, is going to result in a particular type of output. This challenge being difficult to solve, will discourage spammers from sending spam emails. As every time they try to send an email, they will have to do some work in computing the value of nonce, which will discourage them from sending spam emails. Please do recall that the only way to succeed in solving this hash puzzle is to just try enough nonces one by one until you get lucky.

c. Application of hash-cash protocol is in Proof of Work (PoW) scheme used for achieving consensus in a blockchain distributed P2P network. Before adding the latest block, the miners solve a challenge where they find a value of nonce which when appended along with the rest of the block data including the transactions result in a target which ultimately becomes the new block's hash value. Solving this problem is computationally difficult and takes a lot of resources and is time consuming. However, verification of the challenge is a trivial process and happens

Q5. Explain puzzle friendliness property of hash functions. [2 Marks]

Ans: A hash function H is said to be puzzle‑friendly if for every possible n-bit output value y , if k is chosen from a distribution, then it is infeasible to find x such that H(k ‖ x) = y in time significantly less than 2^n .
Intuitively, what this means is that if someone wants to target the hash function to come out to some particular output value y , that if there's part of the input that is chosen in a suitably randomized way, it's very difficult to find another value that hits exactly that target.

Application: In Proof of Work (PoW): Given a target value y, to find what value of nonce (k) when combined with the rest of the block level data (x) resulting in a hash output y (i.e, H(k||x) == y), will take significantly more time to compute as it is a very difficult problem

Q6. You are designing SecureBox, an authenticated online file storage system. For simplicity, there is only a single folder. Users must be able to add, edit, delete, and retrieve files, and to list the folder contents. When a user retrieves a file, SecureBox must provide a proof that the file hasn't been tampered with since its last update. If a file with the given name doesn't exist, the server must report that — again with a proof. Naturally, to be able to verify proofs, users must at all times store some nonzero amount of state derived from the folder contents. Other than this digest the user has no memory of the contents of the files she added. We want to minimize the size of these proofs, the time complexity of verifying them, and the size of the digest that the user must store between operations (these are the objectives). [2+2+2 = 6 Marks]

(a) Here's a naive approach. The user's digest is a hash of the entire folder contents, and proofs are copies of the entire folder contents. With respect to the naive approach, are we able to attain ALL of the objectives mentioned above? If "yes", explain with reasons how we are able to achieve each of the objectives. If "no", explain with reasons why we are not able to achieve any/ALL of the objectives?

(b) Alternatively, the digest could consist of a separate hash for each file, and each file would be its own proof. With respect to this approach, are we able to attain any/ALL of the objectives mentioned above? If "yes", explain with reasons how we are able to achieve the objectives. If "no", explain with reasons why we are not able to achieve any/ALL of the objectives?

(c) Similar to (a) and (b), can you devise an approach where proof size, verification time, and digest size are all sublinear? You might need to explain how this approach will work by designing a protocol that involves some amount of two‑way communication for the user to be able to update her digest when she executes and add, delete, or edit.

Ans: (a) We are NOT ABLE TO MEET ALL the objectives. The user's digest is a hash of the entire folder contents, and proofs are copies of the entire folder contents. Besides, before executing add/delete/edit operations, the user must retrieve the entire folder so that she can recompute the digest.This results in a small digest but large proofs and long verification times.

(b) We are NOT ABLE TO MEET ALL the objectives. In the second approach, the digest could consist of a separate hash for each file, and each file would be its own proof. We are able to minimize the size of the proofs as instead of downloading the entire folder, the respective file can be only downloaded. We are also able to minimize the verification time. The downside of this approach is that it requires digest space that is linear in the number of files in the system.

(c) We will use Merkle trees TO MEET ALL THE objectives. In this approach, initially both the user and the SecureBox storage system will first hash each file and then build a merkle tree consisting of hash values of each file put together as a tree. The root of this merkle tree will contain the hash representing all the files in the folder.

1. Everytime a file undergoes a legitimate change by the user (such as add/update/delete), the merkle tree needs to be recomputed by the user and the root hash will be stored and kept aside for verification later.

2. During add/update/delete operations, SecureBox will first send the merkle root hash as proof to the user.

3. Upon receiving the root hash value, user will compare the same with the previously stored root hash value. If everything is fine, then this means that there is no tampering.
   If the root hash values do not match, then the corresponding file in question has been tampered with. Next, the user will request the two hashes below the root in the canonical Merkle tree, and figure out which hash doesn't match up with our client-side tree. Once we've figured out which subtree is faulty, we can repeat this for the two children of that subtree, and so on until we reach the base. Assuming there's a single faulty block, this will let you pinpoint that block with only O(logn) comparisons (where n is the number of underlying data blocks).

   In such a scenario, the verification time takes O(logn) comparisons
   The proof size and digest size is also sub linear because instead of storing the entire file, both the client and the server will store the merkle root hash.

Q7. Consider the following scenarios: [2+2 = 4 Marks]

a. In ScroogeCoin cryptocurrency discussed in class, suppose an user Mallory tries generating (sk, pk) pairs until her secret key (sk) matches someone else's. What will she be able to do? How long will it take before she succeeds?

b. What if Scrooge coin's random number generator has a bug and the key generation procedure produces only 1,000 distinct pairs? What will happen in such a scenario?

Ans:
a. If the secret key (sk) matches another legitimate user of the cryptocurrency platform, then Mallory will be able to impersonate the user and hence create a transaction by signing it using the secret key she just found out. This transaction will be positively verified by the validators or the system. However, this transaction is in reality a fake transaction created by Mallory, and could imply a stealing of coins from the user's account to her account. Considering the size of the secret key to be n bits, Mallory will in turn try out 2^n combinations in the worst case before she is successful. This can be a lot of time, in the range from a few years to a few thousand years.

b. If the key generation algorithm produces only 1000 distinct pairs, then it means that there can be only 1000 unique (secret key(sk), public key(pk)) combinations. In such a scenario, more than one user of the cryptocurrency platform may share the same combination of sk and pk thus leading to confusion and security vulnerabilities. For instance, two different users with the same pair of (sk and pk) will be having access to the same wallet and hence they will also have access to the same bitcoins in their wallets. The network or the system will not be able to identify these users distinctly.

Q8. Consider the two well known consensus schemes in distributed computing systems - Paxos and Raft. Paxos is a consensus algorithm used in distributed systems to achieve agreement among a group of unreliable processes. Raft provides a robust and reliable way to achieve consensus in distributed systems by electing a leader, replicating a log, and ensuring consistency through a well-defined set of rules. Based on the above, answer the following:
[2+2 = 4 Marks]

(a) Based on the above, why can't we use Paxos and Raft consensus schemes in the Bitcoin blockchain platform?

(b) In relation to the response provided for Q8(a), explain how bitcoin achieves consensus by overcoming the problems mentioned in response to Q8(a).

Ans:

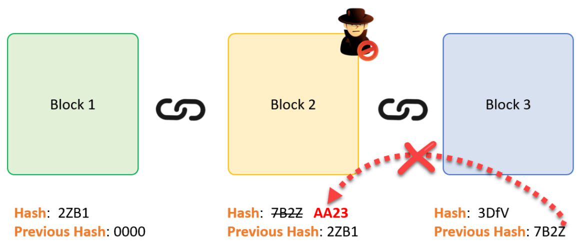
a. Bitcoin blockchain platform being an open and public platform poses two main challenges for conventional consensus schemes. The first challenge is (a) Nodes do not have any fixed identity as the number of nodes continuously change in a dynamic bitcoin P2P network. The second challenge is (b) there is no clear indication on the waiting time for collecting all transactions from the other peer nodes and starting the consensus protocol, because it is unclear as to how much time a node needs to wait when there is no knowledge available regarding the identity of the nodes participating in the consensus round.The third challenge is (c) presence of malicious byzantine category of nodes in a typical bitcoin P2P network making the entire network untrustworthy. Thus, well known consensus schemes which rely on the identity as well as the number of the nodes will not be able to achieve consensus.

b. Bitcoin solves it in the specific context of a currency system.

   1. Bitcoin embraces the notion of randomness. Bitcoin's consensus algorithm relies heavily on randomization. A random node is selected based on a proofing technique (such as PoW, PoS, PoB, etc.), and the remaining nodes agree by including the block (containing the transactions) proposed by this selected node.
   Also, it does away with the notion of a specific starting point and ending point for consensus. Instead, consensus happens over a period of time. As time goes on, the remaining nodes will be able to verify the proof and include the block thus bringing consensus in the whole system.

   2. Second, it introduces the idea of incentives, which is novel for a distributed consensus protocol ensuring that the randomly selected node in the previous point, dutifully proposes a block with legitimate transactions. This is only possible in Bitcoin because it is a currency and therefore has a natural mechanism to incentivize participants to act honestly.

Q9. Consider a simple blockchain shown below. The blockchain infrastructure consists of multiple peer nodes who are working together to maintain the state of the blockchain. Now suppose an attacker is able to change the data present in the Block 2. What will be the most likely sequence of events/steps that he will take next to ensure that his change remains undetected in the blockchain system? Will he be able to succeed in his mission? If "yes", explain how he will be able to do so, otherwise also, explain why he will not be able to succeed in his mission with respect to the steps that he will take which you have identified earlier. [3 Marks]



Ans: To remain undetected in the blockchain system, the attacker needs to perform the following steps in the specified order as follows:

a. Upon changing the data in block 2, he will recompute the new hash value of Block 2 as per the conditions of the challenge in a PoW environment. Recall that to create a new hash of the block, the hash puzzle aka challenge needs to be solved such that it matches a certain target. Hence, this process may require him to solve the PoW puzzle once again to find a new block hash value.
b. Upon finishing recomputation of the new hash for Block 2, he will proceed to recompute the new hash value of Block 3 using the same principles as mentioned above in point (a). This is because Block 2's new hash value needs to be updated in Block 3's corresponding metadata field thus requiring him to recompute a new hash value for Block 3.
   This process of recalculation of new block hashes may be required for all the other blocks (if present) after Block 3 in the blockchain.

Steps (a) and (b) themselves are very difficult for an attacker to perform as recomputing the block hash for all the blocks in a blockchain by repeatedly solving the challenge (puzzle) is tedious and requires a lot of computational resources.

Will he be able to succeed in his mission?

Even after performing steps (a) and (b), he will not be able to hide his changes in all other blockchain blocks present in the peer nodes of the distributed P2P network. Recall that a copy of the blockchain will be present in all the peer nodes of the P2P network. Hence, in order to hide his changes completely (i.e., go undetected), he will have to perform the operations mentioned above in points (a) and (b) for all the peer nodes in the P2P network. Since, no information about the identity of the peer nodes along with the total number of nodes is available to the attacker, execution of such an attack is least likely to be successful.

Q10. A Database Management System such as MySQL and Oracle provides a platform for efficient storage and retrieval of user and application data. In a similar way, Blockchain also stores sensitive application data in the form of a decentralized, distributed ledger. However, both the tools function in different ways. Hence, can you point at least 4 differences between Databases and Blockchains? [2 Marks]

Differences	Databases	Blockchains
Data structure:	The data is organized in a structured table consisting of rows and columns. Each row is called a record, and a record refers to an object or entity.	Data/Information is collected in a group known as blocks. These blocks hold information about the transaction and are linked to the previously filled block.

	Non-relational databases, also known as NoSQL databases, utilize various data structures like key-value pairs, documents, graphs, and wide-column stores to store and manage data.	
Architecture	Database employs a centralized model of storing and retrieving data.	Blockchain is decentralized. The distributed ledger is stored in all the peer nodes of the network.
Type of Data Stored	A group of related data belonging to a person, organization, object, etc. There is no requirement of storing them in a timestamped/historical fashion and one data record may/may not be related to another data record. There is no requirement to store only transactions in a database.	It stores transactions in a timestamped fashion. The transactions can be traced back to earlier transactions thereby maintaining a history of transactions.
Data Persistence	Data stored in databases are mutable and can be updated or deleted later on.	Transactions stored in blockchain are immutable.
Administration	For a Database, there is a need for a database admin or database administrator to handle the stored data.	No administration is required as the system is completely autonomous for storing and retrieval operations.
Speed and Performance	Databases are known for faster execution time and can also handle millions of data at any given time.	Blockchain is considerably slower when compared to databases due to information broadcast, signature verification, consensus mechanism.
Availability/Reliability/Chance of Failure	Due to centralized architecture, it has a high chance of failure/Low reliability/availability.	Due to the decentralized architecture, the chance of failure is low with high availability/reliability.
Control and authority	Relies on a central authority who may be the administrator or the organization who has installed or provided the database to the users.	No specific organization/individual controlling the system. Through various consensus mechanisms, the peer nodes come to agreement regarding the set of transactions to be included in the latest block.
Security	No inbuilt security or trust provided by the database	Blockchains come with in-built cryptography schemes providing both security and trust.