Assignment 2
Suggested Solution

IEMS 5725 Blockchain and Applications
(Fall, 2024-25)
Full Mark: 99

1 True or False Questions (3 points each)

Please **briefly justify** your choice, especially for false statements; otherwise, no marks will be given.

1. When a function is difficult to compute, and the function can be generated randomly, it has already fulfilled the necessary properties to serve as a puzzle in proof-of-work.

   [Ch. 1, p. 35] False. The solution is required to be easily verified. Suppose the computation of the function is used as the puzzle used in proof-of-work. Verification should not mandate going through the exact computation entirely.

2. In a permissionless blockchain system, the consensus protocol can operate without necessarily requiring significant scarce resources such as but not limited to computational power or storage space of the participants of the protocol.

   [Ch. 1, p. 59] False. Requiring scarce resources poses a barrier to launching sybil attacks.

3. In the blockchain network, before a transaction is broadcasted, miners need to ensure the transactions are signed, probably by themselves.

   [Ch. 1, p. 37] False. In a blockchain network, transactions are signed by the sender, not the miners. The miners' role is to confirm the validity of transactions (which includes verifying the digital signatures) and then solve the challenges needed by the consensus mechanism that allows them to add a new block to the blockchain, not to sign the transactions themselves

4. One can safely accept a transaction once it is included in the latest block on the blockchain.

   [Ch. 1, p. 55] False. When there is a fork, a transaction might end up in a shorter chain. So, one should typically wait for 6 blocks to ensure the transaction is irreversible.

5. After a mining pool successfully mined a block, the reward will be evenly distributed to each participant.

   [Ch. 2, p. 40] False. Each participant is rewarded proportionally to their work, which is measured by the number of "partial solutions" submitted.

6. The responsibility of furnishing a `scriptPubkey` lies with the individual aiming to allocate funds from a previously unutilized transaction output.

   [Ch. 3, p. 14] False. The `scriptPubKey` ***is not furnished by the individual aiming to allocate funds from a previously unutilized transaction output (UTXO). Instead, the*** `scriptPubKey` ***is part of the transaction output and is created by the sender who is making the funds available. It specifies the conditions under which the funds can be spent. The individual who wants to spend the UTXO must provide a*** `scriptSig` ***(signature script) that satisfies the conditions set by the*** `scriptPubKey` ***in a subsequent transaction.***

7. A smart contract developer can directly update the contract's code after deploying it on a blockchain to fix vulnerabilities or bugs.
   [Ch. 3, p. 30] False. The data stored on a blockchain, such as the code in a smart contract, is immutable (i.e., cannot be altered once recorded).

8. The "`msg.sender`" variable always refers to the address of the contract that is currently executing.

   [Ch. 4, p. 20] False. In Solidity, the "`msg.sender`" variable refers to the address that initiated the current function call.

9. Hash functions can effectively encrypt data because reversing the hashing process is extremely difficult.

   [Ch. 5, p. 21] False. Due to their irreversible nature, hash functions cannot be used to decrypt or recover the original data, making them unsuitable as encryption methods.

10. In the UTXO model, a one-time-secure signature scheme can be more desirable than a multi-use one.

    [Ch. 5, p. 50] True. A one-time-secure signature scheme suffices as each UTXO is only used once.

11. The Diffie-Hellman protocol enables two parties to generate a shared secret key over an insecure communication channel, which then can be used to establish a secure channel for public-key encryption.

    [Ch. 6, p. 4] False. While the Diffie-Hellman protocol uses primitive operations also used in public-key encryption, it is for establishing a shared secret key for symmetric-key encryption.

12. Multi-signatures assume a fixed number of signers, and signers cannot be hidden from the group.

    [Ch. 6, p. 57] False. Multi-signatures can be produced by an ad hoc set with any number of signers.

13. A Merkle tree enables blockchain networks to efficiently verify the integrity of large sets of data by allowing nodes to download only parts of the blockchain.
    [Ch. 5, p. 44] False. Full nodes/miners in the blockchain network need to verify all the transactions included in a block. Users can prove their transactions are in certain blocks via efficient membership proof.

2 Multiple-Choice Questions (2 points each)

Please choose one for each. No justification is needed.

1. Which of the following statements about blockchain is right?

   (a) Public blockchain allows anyone to read, but only those with specific authorization can write.

   (b) Private blockchain is openly accessible for reading, but writing requires obtaining specific privileges.

   (c) 51% attack refers to a scenario where 51% or more of the cryptocurrency is owned by one entity.

   (d) "Node" in blockchain refers to a digital wallet for storing cryptocurrencies.

   (e) None of the above.

   [Ch. 1-2] e. In a public blockchain, anyone can participate in the network, which means anyone can read and write (submit transactions) as long as they follow the consensus protocol.
   In a private blockchain, both reading and writing permissions are controlled and restricted to a specific group of entities.
   A 51% attack refers to a situation where a single entity or group controls 51% or more of the mining hash rate of a blockchain, potentially allowing them to double-spend coins and prevent other transactions from being confirmed.
   A node in a blockchain refers to a computer that participates in the blockchain network, maintaining a copy of the ledger and, depending on its capabilities, may validate transactions and blocks. A digital wallet, on the other hand, is software that stores the cryptographic keys used to interact with blockchain, allowing users to send and receive cryptocurrencies.

2. Which of the following statements about “hard fork” is right?

   (a) An interim division within the blockchain network that led to separate chains of record.

   (b) An update that introduces new features while ensuring backward compatibility with previous versions of the software.

   (c) A permanent divergence from the previous version of the blockchain.

   (d) A procedural enhancement aimed at streamlining consensus mechanisms without necessitating network-wide consensus on the change.

   (e) None of the above.

   [Ch. 2] c. Hard forks result in two separate chains that do not recognize the validity of the other's transactions post-fork.

3. Which of the following updates to Bitcoin can be done with a soft fork?

   (a) Increasing the maximum block size.

   (b) Decreasing the maximum block size.

   (c) Changing the method of calculating block size, *e.g.*, assigning weights to different parts.

   (d) Replacing the hash function for the proof-of-work puzzle.

   (e) Introducing new functionalities, *e.g.*, a new address type.

   [Ch. 2] b. Soft forks tighten the rules on blockchain transactions and are backward-compatible, meaning that nodes that have not upgraded can still validate transactions as long as they don't violate the new, stricter rules. c is also accepted, although in general (except like SegWit), changing the block verification should not be backwards-compatible.

4. What is the main benefit of using smart contracts?

   (a) Superseding the utility of digital currencies.

   (b) Minimizing dependence on centralized authority figures for transaction verification.

   (c) Completely eliminating the risk of operational discrepancies and code-based vulnerabilities.

   (d) Lowering the (financial and technical) barriers associated with executing and enforcing agreements.

   (e) Guaranteeing the accuracy and confidentiality of transactional data.

   [Ch. 3] b. Smart contracts enhance the functionality of digital currencies by enabling programmable transactions but do not necessarily supersede the utility of digital currencies themselves.
   Smart contracts rely on the underlying blockchain for transaction verification, which is a distributed rather than centralized process.
   Smart contracts can reduce some risks associated with traditional contracts, but they are prone to vulnerabilities if not written securely.
   The execution cost of smart contracts is subject to fluctuating gas prices, which can make operations costly, particularly during network congestion, and the direct act of writing a secure smart contract may present technical challenges. Optimizing them to minimize gas usage necessitates technical expertise and an in-depth understanding of the blockchain's programming model.
   While smart contracts can ensure that transactions occur as programmed, they do not inherently guarantee the accuracy of input data nor do they provide confidentiality of transactional data on a public blockchain.

5. Which of the following is NOT static (primitive) data types in Solidity? (You may check if they are passed by value to a function.)

   (a) uint8

   (b) string

   (c) bytes8

   (d) enum

   (e) address

   [Ch. 4] b. In Solidity, string is a dynamically sized type, not a primitive (static size) type. It is passed by reference, not value.

6. In Solidity, what is the purpose of the "public" keyword when used with a variable?

   (a) It makes the variable writable by any external contract.
   (b) It makes the variable non-writable by any external contract.
   (c) It restricts variable access to within the contract only.
   (d) It encrypts the variable for security and makes it readable externally.
   (e) It makes the variable readable externally.

   [Ch. 4] e. It makes the variable readable externally. The public keyword automatically generates a getter function for the variable, allowing it to be read from outside the contract.

7. Solidity, what is the "external" function visibility modifier used for?

   (a) To allow a function to be called only from within the contract itself.
   (b) To allow a function to be called from other contracts and transactions.
   (c) To restrict function access to certain addresses.
   (d) To mark a function for internal use only, not even by derived contracts.
   (e) All of the above.

   [Ch. 4] b. The external visibility restricts the function to be callable only from outside the contract. They cannot be called internally, except when using `this.functionName()`***, which is treated as an external call.***

8. What are the roles played by cryptographic hashes in blockchain?

   i. Ensure data integrity by verifying that transaction data has not been altered.
   ii. Encrypt transaction data for privacy.
   iii. Link blocks together securely by including the previous block's hash in the next block.
   iv. Enable the mining process by requiring miners to find a hash below a certain target.
   v. Generate public and private key pairs for users.

   (a) (i), (iii), (iv)
   (b) (i), (ii), (iii)
   (c) (ii), (iii), (iv), (v)
   (d) (i), (iii), (iv), (v)
   (e) (i), (iii)

   [Ch. 5] a. Cryptographic hashes are used to ensure data integrity, link blocks securely, and facilitate the mining process. They do not encrypt data for privacy nor generate key pairs.

9. Which of the following best describes how digital signatures contribute to the security of blockchain?

   (a) Digital signatures encrypt the transaction data, making it unreadable to unauthorized parties.
   (b) Digital signatures ensure that the transaction cannot be altered without detection.
   (c) Digital signatures speed up the verification process of transactions.
   (d) Digital signatures decrease the size of transaction data.
   (e) Digital signatures verify the identity of the transaction participants by comparing the signature to a list of known signatures.

   [Ch. 6] b. Digital signatures ensure that the transaction cannot be altered without detection. They provide integrity and non-repudiation by proving that the signer has consented to the transaction.

10. Which of the following about textbook RSA is right?

    (a) It is probabilistic.
    (b) Its decryption algorithm should be used to generate a digital signature.
    (c) It can only be used to encrypt a number less than the RSA modulus.
    (d) It is a symmetric encryption scheme.
    (e) Its security depends on the difficulty of solving discrete logarithm problem.

    [Ch. 6] c. It is a deterministic public-key encryption scheme. While textbook RSA decryption algorithm can be used to generate a digital signature, it is forgeable and hence insecure. If the plaintext is larger than the RSA modulus, then the result of the modulo operation will not represent the original plaintext, leading to incorrect decryption. Its security depends on the integer factoring problem.

3 Short Questions (4 points each)

Please keep the answer concise. **within** 250 **words, roughly** for each question.

1. Compared with physical currency, what is the main problem with digital money? How does a centralized digital bank prevent it? How does Bitcoin prevent it?

   The main problem is the double-spending issue. The traditional digital bank prevents it by relying on a trusted third party to verify and record transactions. Bitcoin prevents it by recording transactions on a public ledger, which is maintained by a network of distributed nodes.

2. Explain the concept of consensus algorithms in blockchain technology and discuss two popular consensus mechanisms used in public blockchains.

   Consensus algorithms are used in blockchain technology to ensure that all nodes in the network agree on the state of the blockchain. Two popular consensus mechanisms used in public blockchains are Proof-of-Work (PoW) and Proof-of-Stake (PoS). In PoW, miners compete to solve a cryptographic puzzle to add a new block to the chain, and the first one to solve it is rewarded with new coins. In PoS, validators are chosen based on the number of coins they hold, and they are responsible for creating new blocks and validating transactions.

3. Describe the meanings of, and differences between, threshold signatures and multisignatures.

   Threshold signatures enable a subset of participants (e.g., 3 ***out of*** 5***) to produce a collective signature, without revealing who exactly participated.***

   Multisignatures require all signatories claimed by a signature to sign.

   The difference is that multisignatures is less flexible than threshold signatures, as it requires everyone purported by the signature to sign.

4. Alice wants to buy a book from Bob using Bitcoin. Alice proposes to transfer 1 Bitcoin to Bob's address, and Bob shall immediately give the book to Alice when the transaction is included in the most recently mined block. Explain why this is not a good proposal for Bob, and how Bob can make sure he receives the Bitcoin before handing out the book.

   It is risky for Bob since transactions first enter an unconfirmed state in Bitcoin's transaction confirmation process before being included in a block and the Bitcoin network typically requires at least 6 ***confirmations to consider a transaction irreversible. For Bob, immediate handover of the book post-transaction poses:***

   - ***Transaction Delay Risk: The transaction might take longer than expected to confirm, delaying his receipt of payment.***
   - ***Double Spending Risk: Alice could spend the same Bitcoin elsewhere before the transaction is fully confirmed.***

   To mitigate these risks, Bob can:

   - ***Wait for Confirmations: Hold off on handing over the book until the transaction receives at least 6 confirmations, ensuring its irreversibility.***
   - ***Use a Payment Processor: Employ a Bitcoin payment processor that offers instant confirmations, reducing the wait time while ensuring payment security.***
   - ***Opt for Escrow: Engage a Bitcoin escrow service to securely hold the Bitcoin until the transaction meets all confirmation requirements.***

5. For a Merkle tree with 512 leaves using SHA-256, what is the bit size of the proof needed to verify the membership of a particular data item? Show your calculation.

   [Ch. 5, p. 33,39] The proof size equals the Merkle tree height multiplied by the output length of the hash function used. A Merkle tree with 512 ***leaves has height*** $\log_2 512 = 9$***, and the output of SHA-256 is*** 256 ***bits. So, the proof size is*** $9 \times 256 = 2304$ ***bits. (excluding the value to be proven)***

4 Long Question (20 points)

Consider the code in listing 1. At a high level, a user can call `saveHash()` with some ethers and a hash ($h(x)$ with secret x) to register and lock the ethers. The contract will save the hash, the account balance, and the list of registered users (owners). An owner can retrieve the deposit via `withdrawMoney()` with a correct hash pre-image x that matches the stored hashPuzzle.

1. List two potential vulnerabilities (or issues) in the contract.

   Any 2 of the following:
   - balance is deducted before transfer/reentrancy attack.
   - saveHash does not check for double registration [Functionality issue]
   - use of tx.origin
   - incorrect usage of the hash function.
   - Line 27: i = j should be j = 1
   (or any mistakes that make sense)

2. Suppose this contract has been deployed; describe how an attacker can exploit the mistake(s). You may assume users have made transactions to the contract.

   Reentrance may not work for using "send" which has 2300 gas limit.
   However, if the gas cost is reduced (for some forks) where send can cost more than 2300 gas, then a reentrance attack would be possible
   Phishing attack to get the victim transferred to the badContract when the attacker can work out the preimage (bad use of hash function)

3. For the two vulnerabilities listed, describe how to fix them. Furthermore, add a line before Line 14 to prevent double registration.

   Uses reentrancy guard; checks for double registration; use msg.sender; use salt to make recovery attack harder.

4. Briefly describe how hash functions can be used to realize a (cryptographic) commitment scheme, and briefly explain how the two security properties of commitments are achieved.

   [Ch. 5, p. 21+] Cryptographic commitment schemes allow one party to commit to a value while keeping it hidden, with the ability to reveal it later.
   To commit, the committing party selects a value x and uses a hash function H to compute $H(x||r)$, where r is a randomly chosen value (nonce) to ensure that the commitment is unique. This hash value is then shared with the other party. Due to the one-wayness of hash functions, the original value x is concealed, achieving the hiding property, i.e., without the nonce r, it is computationally infeasible for anyone to determine x from $H(x||r)$.
   To reveal, the committing party discloses the original value x and the nonce r. The receiving party computes $H(x||r)$ using the same hash function and verifies it against the previously received hash value. This process ensures the binding property, meaning once a commitment is made, the committer cannot change the value x without altering the hash output, which is computationally infeasible due to the collision resistance of the hash function.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.24;
3
4  contract badContract {
5
6    mapping (address => bytes32) public hashPuzzle;
7    mapping (address => uint256) public balance;
8    address[] owners;
9    receive() external payable {}
10   // Each user can register to deposit to the contract with a puzzle lock
11   // User will supply the Hash(guess) as the lock
12   // and provide the "guess" to withdraw in withdrawMoney()
13   function saveHash(bytes32 hash) payable public {
14     hashPuzzle[msg.sender] = hash;
15     require(msg.value > 0, "deposit some money!");
16     balance[msg.sender] = msg.value;
17     owners.push(msg.sender);
18   }
19
20   // A registered user can withdraw money by supplying a correct guess
21   function withdrawMoney(string memory guess) public returns (bool){
22     bool check=false;
```

```
    uint j;
    for (uint i=0; i<owners.length; i++) {
      if (owners[i]==tx.origin) {
        check = true;
        i = j;
      }
    }
    require(check==true, "user not exist");
    uint256 amount = balance[tx.origin];
    require(sha256(abi.encode(guess))==hashPuzzle[tx.origin],"error in hash");
    if (amount > 0) {
      if (!payable(tx.origin).send(balance[tx.origin])) {
        balance[tx.origin] = amount;
        return false;
      }
      balance[tx.origin] = 0;
    }
    return true;
  }
}
```

Listing 1: Buggy Contract

- End -