Lesson 3.1 Quiz: Cryptographic Foundations for Finance

Module 3: The Trust Problem

Prof. Dr. Joerg Osterrieder

Digital Finance — BSc Course

Question 1

A payment processor stores a SHA-256 hash of every incoming transaction record. A junior auditor asks: "Why not just store the records themselves?" Which answer **best** explains the purpose of hashing in this context?

- (A) Hashing compresses the records to save storage space
- (B) Hashing encrypts the records so unauthorized users cannot read them
- (C) Hashing creates a fixed-size fingerprint that detects any unauthorized modification to the record
- (D) Hashing speeds up database queries by replacing text with numbers

Question 1

A payment processor stores a SHA-256 hash of every incoming transaction record. A junior auditor asks: "Why not just store the records themselves?" Which answer **best** explains the purpose of hashing in this context?

- (A) Hashing compresses the records to save storage space
- (B) Hashing encrypts the records so unauthorized users cannot read them
- (C) Hashing creates a fixed-size fingerprint that detects any unauthorized modification to the record
- (D) Hashing speeds up database queries by replacing text with numbers

Answer: (C) A cryptographic hash produces a fixed-size digest that changes completely if even one bit of the input is altered. This allows tamper detection — not compression, encryption, or query optimization.

Question 2

A blockchain developer tells a client: "SHA-256 is a one-way function." The client asks what "one-way" means. Which explanation is **most accurate**?

- A. The hash can only be computed on one specific type of hardware
- B. Given the hash output, it is computationally infeasible to find the original input
- C. The function can only be called once per input
- D. The hash output always has fewer characters than the input

Question 2

A blockchain developer tells a client: "SHA-256 is a one-way function." The client asks what "one-way" means. Which explanation is **most accurate**?

- (A) The hash can only be computed on one specific type of hardware
- (B) Given the hash output, it is computationally infeasible to find the original input
- (C) The function can only be called once per input
- (D) The hash output always has fewer characters than the input

Answer: (B) "One-way" refers to pre-image resistance: computing the hash from the input is fast, but reversing the process — finding an input that produces a given hash — is computationally infeasible.

Question 3

An intern hashes the string `"Transfer $500 to Alice"` and then hashes `"Transfer $500 to alice"` (lowercase ‘a’). The two hashes are completely different. Which property of cryptographic hash functions does this demonstrate?

- A. Collision resistance
- B. Pre-image resistance
- C. The avalanche effect
- D. Second pre-image resistance

Question 3

An intern hashes the string `"Transfer $500 to Alice"` and then hashes `"Transfer $500 to alice"` (lowercase 'a'). The two hashes are completely different. Which property of cryptographic hash functions does this demonstrate?

- (A) Collision resistance
- (B) Pre-image resistance
- (C) The avalanche effect
- (D) Second pre-image resistance

Answer: (C) The avalanche effect means that a tiny change in the input (one character case change) produces a completely different hash output. On average, about 50% of the output bits flip.

Question 4

A compliance officer needs to verify that a regulatory filing has not been altered since it was submitted. She compares the hash she computed today with the hash recorded at submission. Which property of hash functions makes this verification **trustworthy**?

- (A) The avalanche effect — any change would flip output bits
- (B) Pre-image resistance — the original cannot be derived from the hash
- (C) Collision resistance — it is infeasible that a different document produces the same hash
- (D) Determinism — the same input always produces the same output

Question 4

A compliance officer needs to verify that a regulatory filing has not been altered since it was submitted. She compares the hash she computed today with the hash recorded at submission. Which property of hash functions makes this verification **trustworthy**?

- (A) The avalanche effect — any change would flip output bits
- (B) Pre-image resistance — the original cannot be derived from the hash
- (C) Collision resistance — it is infeasible that a different document produces the same hash
- (D) Determinism — the same input always produces the same output

Answer: (C) The verification relies on collision resistance: if the hashes match, you can be confident no different document could have produced that same hash. Determinism is necessary but not sufficient — without collision resistance, a forged document could match the hash.

Question 5

A blockchain contains 1,024 transactions organized in a Merkle tree. A lightweight mobile wallet needs to verify that a specific transaction is included in the block. How many sibling hashes must the network provide in the inclusion proof?

- A 5
- B 10
- C 512
- D 1,024

Question 5

A blockchain contains 1,024 transactions organized in a Merkle tree. A lightweight mobile wallet needs to verify that a specific transaction is included in the block. How many sibling hashes must the network provide in the inclusion proof?

- (A) 5
- (B) 10
- (C) 512
- (D) 1,024

Answer: (B) A Merkle inclusion proof requires $\log_2(n)$ hashes. $\log_2(1{,}024) = 10$. The wallet receives 10 sibling hashes and recomputes the path to the Merkle root, comparing it with the known root in the block header.

Question 6

You are constructing a Merkle tree from four transactions: T_1, T_2, T_3, T_4. You compute $H_1 = \text{Hash}(T_1)$ through $H_4 = \text{Hash}(T_4)$. What is the **next** step?

- (A) Hash all four leaf hashes together in one operation to get the Merkle root
- (B) Concatenate $H_1 \| H_2$ and hash them; concatenate $H_3 \| H_4$ and hash them
- (C) Sort the hashes alphabetically, then hash the sorted list
- (D) XOR all four hashes to produce the root

Question 6

You are constructing a Merkle tree from four transactions: T_1, T_2, T_3, T_4. You compute $H_1 = \text{Hash}(T_1)$ through $H_4 = \text{Hash}(T_4)$. What is the **next** step?

- (A) Hash all four leaf hashes together in one operation to get the Merkle root
- (B) Concatenate $H_1 \| H_2$ and hash them; concatenate $H_3 \| H_4$ and hash them
- (C) Sort the hashes alphabetically, then hash the sorted list
- (D) XOR all four hashes to produce the root

Answer: (B) Merkle trees are built bottom-up by hashing pairs: $H_{12} = \text{Hash}(H_1 \| H_2)$ and $H_{34} = \text{Hash}(H_3 \| H_4)$. Then the root is $\text{Hash}(H_{12} \| H_{34})$.

Question 7

Alice wants to send Bob an encrypted message using public-key cryptography. She has Bob's public key and her own key pair. Which key does she use to **encrypt** the message?

A. Her own private key
B. Her own public key
C. Bob's public key
D. Bob's private key

Question 7

Alice wants to send Bob an encrypted message using public-key cryptography. She has Bob's public key and her own key pair. Which key does she use to **encrypt** the message?

- (A) Her own private key
- (B) Her own public key
- (C) Bob's public key
- (D) Bob's private key

Answer: (C) To encrypt a message *for Bob*, Alice uses Bob's public key. Only Bob's private key can decrypt it, ensuring only the intended recipient can read the message.

Question 8

A cryptocurrency wallet shows a user the following signing process: (1) hash the transaction data, (2) sign the hash with the private key, (3) broadcast the transaction + signature. Why is the **hash** signed rather than the raw transaction data?

- (A) Signing the hash hides the transaction details from the network
- (B) Hashing first converts the variable-length transaction into a fixed-size input, making the signing operation efficient
- (C) Signing the raw data would reveal the private key
- (D) The hash adds an extra layer of encryption to the signature

Question 8

A cryptocurrency wallet shows a user the following signing process: (1) hash the transaction data, (2) sign the hash with the private key, (3) broadcast the transaction + signature. Why is the **hash** signed rather than the raw transaction data?

- (A) Signing the hash hides the transaction details from the network
- (B) Hashing first converts the variable-length transaction into a fixed-size input, making the signing operation efficient
- (C) Signing the raw data would reveal the private key
- (D) The hash adds an extra layer of encryption to the signature

Answer: (B) Digital signature algorithms operate on fixed-size inputs. Hashing the transaction first produces a fixed-size digest (e.g., 256 bits), which is then signed. This is computationally efficient and mathematically well-defined.

Question 9

A bank implements a commitment scheme for sealed-bid auctions: each bidder submits Hash(bid||nonce) before the deadline, then reveals the bid and nonce afterward. Which hash property ensures a bidder **cannot change their bid** after submission?

- (A) The avalanche effect — changing the bid changes the hash
- (B) Pre-image resistance — the auctioneer cannot reverse-engineer the bid
- (C) Collision resistance — the bidder cannot find a different bid with the same hash
- (D) Determinism — the same bid always produces the same hash

Question 9

A bank implements a commitment scheme for sealed-bid auctions: each bidder submits Hash(bid||nonce) before the deadline, then reveals the bid and nonce afterward. Which hash property ensures a bidder **cannot change their bid** after submission?

- (A) The avalanche effect — changing the bid changes the hash
- (B) Pre-image resistance — the auctioneer cannot reverse-engineer the bid
- (C) Collision resistance — the bidder cannot find a different bid with the same hash
- (D) Determinism — the same bid always produces the same hash

Answer: (C) The commitment is binding because collision resistance prevents the bidder from finding a different (bid, nonce) pair that produces the same hash. Pre-image resistance keeps the bid hidden, but collision resistance is what prevents changing it.

Question 10

An exchange publishes a Merkle tree of all customer balances for a proof-of-reserves audit. Customer X wants to verify that their balance of 2.5 BTC is included. What information does Customer X need from the exchange?

- A. The complete list of all customer balances
- B. Only the Merkle root
- C. Their leaf hash plus the sibling hashes along the path to the root
- D. The private key used to construct the tree

Question 10

An exchange publishes a Merkle tree of all customer balances for a proof-of-reserves audit. Customer X wants to verify that their balance of 2.5 BTC is included. What information does Customer X need from the exchange?

- (A) The complete list of all customer balances
- (B) Only the Merkle root
- (C) Their leaf hash plus the sibling hashes along the path to the root
- (D) The private key used to construct the tree

Answer: (C) The customer needs their own leaf data (to compute their leaf hash) plus the sibling hashes at each level of the tree. They recompute the path up and compare the result with the published Merkle root. This verifies inclusion without revealing other customers' data.

Question 11

A node on the Bitcoin network receives a transaction claiming to transfer 0.5 BTC from address `1A3x...` to address `1B7y...`. The transaction includes an ECDSA signature. What does the node verify **first**?

- That the sender has sufficient balance (0.5 BTC or more)
- That the signature was produced by the private key corresponding to `1A3x...`
- That the transaction amount is below the network maximum
- That the recipient address `1B7y...` exists on the blockchain

Question 11

A node on the Bitcoin network receives a transaction claiming to transfer 0.5 BTC from address `1A3x...` to address `1B7y...`. The transaction includes an ECDSA signature. What does the node verify **first**?

- (A) That the sender has sufficient balance (0.5 BTC or more)
- (B) That the signature was produced by the private key corresponding to `1A3x...`
- (C) That the transaction amount is below the network maximum
- (D) That the recipient address `1B7y...` exists on the blockchain

Answer: (B) Signature verification is the first cryptographic check: does the ECDSA signature match the public key associated with the sender address? If the signature is invalid, the transaction is rejected immediately, regardless of balance.

Question 12

ECDSA uses 256-bit keys while RSA requires 3,072-bit keys for equivalent security. A mobile banking app is choosing between the two. What is the **primary practical advantage** of ECDSA in this context?

- A ECDSA signatures are impossible to forge, while RSA signatures can be
- B ECDSA's smaller keys mean less storage, bandwidth, and computation — critical for mobile devices
- C ECDSA does not require a private key, simplifying key management
- D ECDSA is a symmetric algorithm, which is always faster

Question 12

ECDSA uses 256-bit keys while RSA requires 3,072-bit keys for equivalent security. A mobile banking app is choosing between the two. What is the **primary practical advantage** of ECDSA in this context?

- (A) ECDSA signatures are impossible to forge, while RSA signatures can be
- (B) ECDSA's smaller keys mean less storage, bandwidth, and computation — critical for mobile devices
- (C) ECDSA does not require a private key, simplifying key management
- (D) ECDSA is a symmetric algorithm, which is always faster

Answer: (B) At equivalent security levels, ECDSA keys are roughly 12x smaller than RSA keys. This translates to smaller signatures, less network bandwidth, and faster computation — all critical for battery-limited, bandwidth-limited mobile devices.

Question 13

An attacker modifies transaction T_3 in a Merkle tree of 8 transactions. Which of the following hashes in the tree will **change** as a result?

- (A) Only the leaf hash of T_3
- (B) The leaf hash of T_3 and all hashes on the path from T_3 to the root
- (C) Every hash in the entire tree
- (D) Only the Merkle root

Question 13

An attacker modifies transaction T_3 in a Merkle tree of 8 transactions. Which of the following hashes in the tree will **change** as a result?

- (A) Only the leaf hash of T_3
- (B) The leaf hash of T_3 and all hashes on the path from T_3 to the root
- (C) Every hash in the entire tree
- (D) Only the Merkle root

Answer: (B) Due to the avalanche effect, changing T_3 changes its leaf hash. This propagates upward: every parent hash on the path to the root changes. However, sibling branches that do not contain T_3 remain unchanged. This is exactly $\log_2(n) + 1$ hashes.

Question 14

A digital signature provides three guarantees: authentication, integrity, and non-repudiation. A disgruntled employee signs a fraudulent transaction and later claims they did not authorize it. Which property of digital signatures **directly refutes** this claim?

- (A) Integrity — the transaction has not been modified
- (B) Authentication — the signature proves the identity of the signer
- (C) Non-repudiation — the signer cannot deny having signed, since only their private key could have produced the signature
- (D) Confidentiality — the transaction was encrypted during transmission

Question 14

A digital signature provides three guarantees: authentication, integrity, and non-repudiation. A disgruntled employee signs a fraudulent transaction and later claims they did not authorize it. Which property of digital signatures **directly refutes** this claim?

- (A) Integrity — the transaction has not been modified
- (B) Authentication — the signature proves the identity of the signer
- (C) Non-repudiation — the signer cannot deny having signed, since only their private key could have produced the signature
- (D) Confidentiality — the transaction was encrypted during transmission

Answer: (C) Non-repudiation means the signer cannot credibly deny signing. Since the signature can only be produced by the holder of the corresponding private key, the employee's denial is cryptographically refuted (assuming proper key management).

Question 15

Alice signs a transaction "Pay Bob 1 BTC" and broadcasts it. An attacker intercepts the signed transaction and changes it to "Pay Eve 1 BTC." Will the attack succeed?

A. Yes — the attacker can modify the transaction and reuse Alice's signature

B. Yes — the attacker can compute Alice's private key from the signature

C. No — modifying the transaction invalidates the signature because the hash of the modified message will not match

D. No — but only because the network rejects all transactions from intercepted broadcasts

Question 15

Alice signs a transaction "Pay Bob 1 BTC" and broadcasts it. An attacker intercepts the signed transaction and changes it to "Pay Eve 1 BTC." Will the attack succeed?

- (A) Yes — the attacker can modify the transaction and reuse Alice's signature
- (B) Yes — the attacker can compute Alice's private key from the signature
- (C) No — modifying the transaction invalidates the signature because the hash of the modified message will not match
- (D) No — but only because the network rejects all transactions from intercepted broadcasts

Answer: (C) The digital signature is bound to the exact hash of the original message. Any modification to the transaction changes the hash, and the original signature will fail verification against the new hash. The attacker cannot forge a new valid signature without Alice's private key.

Question 16

In the double-spend problem, Alice sends 1 BTC to Bob and simultaneously sends the *same* 1 BTC to Charlie. Both transactions have valid digital signatures. Why does cryptography **alone** fail to prevent this?

- (A) Digital signatures cannot verify the sender's identity
- (B) Hash functions cannot detect duplicate transactions
- (C) Cryptography ensures each signature is valid but cannot determine the global ordering of transactions
- (D) Public-key encryption prevents nodes from seeing both transactions

Question 16

In the double-spend problem, Alice sends 1 BTC to Bob and simultaneously sends the *same* 1 BTC to Charlie. Both transactions have valid digital signatures. Why does cryptography **alone** fail to prevent this?

- (A) Digital signatures cannot verify the sender's identity
- (B) Hash functions cannot detect duplicate transactions
- (C) Cryptography ensures each signature is valid but cannot determine the global ordering of transactions
- (D) Public-key encryption prevents nodes from seeing both transactions

Answer: (C) Both transactions are cryptographically valid — both carry correct signatures from Alice's private key. The problem is deciding which came first. Cryptography provides authentication and integrity, but establishing a canonical ordering requires a consensus mechanism.

Question 17

A startup proposes using MD5 (128-bit output) instead of SHA-256 (256-bit output) for their blockchain to "save storage space." An advisor warns against this. What is the **strongest** technical argument against MD5?

A. MD5 is slower to compute than SHA-256

B. MD5's collision resistance is broken — practical collisions have been demonstrated, making the blockchain vulnerable to transaction substitution

C. MD5 produces output that is too short to be displayed in a user interface

D. MD5 is not supported by modern programming languages

Question 17

A startup proposes using MD5 (128-bit output) instead of SHA-256 (256-bit output) for their blockchain to "save storage space." An advisor warns against this. What is the **strongest** technical argument against MD5?

- (A) MD5 is slower to compute than SHA-256
- (B) MD5's collision resistance is broken — practical collisions have been demonstrated, making the blockchain vulnerable to transaction substitution
- (C) MD5 produces output that is too short to be displayed in a user interface
- (D) MD5 is not supported by modern programming languages

Answer: (B) MD5's collision resistance was broken in 2004 (Wang et al.), and practical collision attacks now take seconds on standard hardware. An attacker could craft two different transactions with the same MD5 hash, enabling undetectable substitution.

Question 18

A user loses their private key but still has their public key and blockchain address. Which of the following is true?

- A. They can derive the private key from the public key using elliptic curve math
- B. They can still receive funds but can never spend them, because signing transactions requires the private key
- C. They can contact the blockchain's central authority to reset their key
- D. They can use their public key to sign transactions temporarily

Question 18

A user loses their private key but still has their public key and blockchain address. Which of the following is true?

- (A) They can derive the private key from the public key using elliptic curve math
- (B) They can still receive funds but can never spend them, because signing transactions requires the private key
- (C) They can contact the blockchain's central authority to reset their key
- (D) They can use their public key to sign transactions temporarily

Answer: (B) The public key can still receive incoming transactions (others encrypt/address to it). However, spending requires signing with the private key. Since the public-to-private derivation is computationally infeasible, and blockchains have no central key recovery authority, the funds are permanently inaccessible.

Question 19

A government regulator proposes requiring all cryptocurrency users to register their public keys with a central authority, which would hold backup copies of private keys. Evaluate this proposal from a cryptographic security perspective. Which concern is **most critical**?

- (A) The central authority would need too much storage for all the keys
- (B) A single breach of the central authority would compromise every user's funds simultaneously
- (C) The public keys would become visible to everyone, reducing privacy
- (D) The registration process would slow down transaction processing

Question 19

A government regulator proposes requiring all cryptocurrency users to register their public keys with a central authority, which would hold backup copies of private keys. Evaluate this proposal from a cryptographic security perspective. Which concern is **most critical**?

- (A) The central authority would need too much storage for all the keys
- (B) A single breach of the central authority would compromise every user's funds simultaneously
- (C) The public keys would become visible to everyone, reducing privacy
- (D) The registration process would slow down transaction processing

Answer: (B) Centralizing private keys creates a catastrophic single point of failure. A breach exposes every user's signing capability, enabling mass theft. This directly contradicts the security model of public-key cryptography, where the private key must remain exclusively with the owner.

Question 20

A financial institution currently uses a trusted third party (a bank) to prevent double-spending. A consultant proposes replacing this with a system using only cryptographic hash functions, Merkle trees, and digital signatures — but **no consensus mechanism**. Is this system viable?

A. Yes — digital signatures alone can prevent double-spending because they prove ownership

B. Yes — Merkle trees can detect conflicting transactions and automatically reject the second one

C. No — without a consensus mechanism, there is no way to establish which of two conflicting transactions came first across a distributed network

D. No — hash functions are too slow for real-time transaction processing

Question 20

A financial institution currently uses a trusted third party (a bank) to prevent double-spending. A consultant proposes replacing this with a system using only cryptographic hash functions, Merkle trees, and digital signatures — but **no consensus mechanism**. Is this system viable?

- (A) Yes — digital signatures alone can prevent double-spending because they prove ownership
- (B) Yes — Merkle trees can detect conflicting transactions and automatically reject the second one
- (C) No — without a consensus mechanism, there is no way to establish which of two conflicting transactions came first across a distributed network
- (D) No — hash functions are too slow for real-time transaction processing

Answer: (C) Cryptographic tools provide integrity (hashing), efficient verification (Merkle trees), and authentication (signatures), but they cannot solve the ordering problem in a distributed setting. Without consensus, two conflicting transactions with valid signatures cannot be resolved — this is precisely why Bitcoin introduced Proof of Work.