Lesson 3.1 Exercises: Cryptographic Foundations for Finance

Module 3: The Trust Problem

Prof. Dr. Joerg Osterrieder

Digital Finance — BSc Course

(c) Joerg Osterrieder 2025–2026

Exercise 1: Matching Hash Properties to Attacks

Scenario: A FinTech company uses SHA-256 hashes to verify the integrity of loan contracts stored in its database. A security consultant presents three hypothetical attack scenarios:

1. **Attack A:** An attacker obtains the hash `e3b0c4...` and attempts to construct a fake contract that produces this exact hash.
2. **Attack B:** An attacker creates two contracts — one legitimate, one fraudulent — that produce the *same* hash, then submits the legitimate one for approval and later swaps it with the fraudulent one.
3. **Attack C:** An attacker changes the interest rate in a contract from 5.0% to 5.1% and hopes the hash remains close enough to pass a similarity check.

Tasks:

a. For each attack (A, B, C), name the specific hash function property that prevents it.
b. Explain in one sentence *why* each property defeats the corresponding attack.
c. If SHA-256 were replaced with a non-cryptographic hash function (e.g., CRC-32), which of the three attacks would become feasible? Explain.

Difficulty: Introductory — tests understanding of the three core hash properties.

Exercise 2: Exploring SHA-256 Behavior

Scenario: You have access to a SHA-256 calculator (online or via the command line: `echo -n "text" | sha256sum`).

Tasks:

a. Compute the SHA-256 hash of the string `"Pay Alice 100"` and the hash of `"Pay Alice 101"`. Write down both hashes (first 16 hex characters are sufficient).

b. Count how many of the first 16 hex characters differ between the two hashes. Express this as a percentage.

c. Does your result support or contradict the claim that "the avalanche effect flips approximately 50% of output bits"? Explain, noting that each hex character represents 4 bits.

d. Now compute the hash of the empty string `""`. Is the output also 64 hex characters? What does this tell you about the fixed-output property?

Difficulty: Introductory — hands-on exploration of SHA-256 determinism and avalanche effect.

Exercise 3: Build a Merkle Tree

Scenario: A block contains exactly four transactions with the following (simplified) hashes:

Transaction	**Leaf Hash**
T_1: "Alice $\rightarrow$ Bob, 2 BTC"	$H_1 = \texttt{a1b2}$
T_2: "Bob $\rightarrow$ Charlie, 0.5 BTC"	$H_2 = \texttt{c3d4}$
T_3: "Charlie $\rightarrow$ Dave, 1 BTC"	$H_3 = \texttt{e5f6}$
T_4: "Dave $\rightarrow$ Eve, 3 BTC"	$H_4 = 0789$

Assume a simplified hash function where $\text{Hash}(X \| Y) =$ first 4 hex characters of SHA-256(`"XY"`). For example, $\text{Hash}(\texttt{a1b2} \| \texttt{c3d4}) = \text{SHA-256}(\texttt{"a1b2c3d4"})$[first 4 chars].

Tasks:

a. Draw the complete Merkle tree with all intermediate hashes (compute them).

b. Write down the Merkle root.

c. List the sibling hashes needed for an inclusion proof of T_3.

d. If T_2 is modified (hash changes to `ffff`), which intermediate hashes change? Which remain the same?

Difficulty: Intermediate — requires computation and tree construction.

Exercise 4: Merkle Proof Scalability

Scenario: You are designing a lightweight verification system for a financial ledger. The ledger grows over time, and clients on mobile devices must verify individual transactions.

Tasks:

a. Complete the table by computing the Merkle proof size (number of hashes) for each ledger size:

Transactions (n)	**Full download (hashes)**	**Merkle proof (hashes)**
16	16	?
256	256	?
1,048,576 (2^{20})	1,048,576	?

b. If each SHA-256 hash is 32 bytes, how many bytes does a Merkle proof require for a block with 1,048,576 transactions?

c. A competitor proposes a "flat hash list" where the verifier downloads all n hashes and searches for a match. Compare the bandwidth cost for $n = 1{,}048{,}576$ between the flat list and the Merkle proof.

d. Explain why the logarithmic scaling of Merkle proofs is essential for mobile wallet viability.

Difficulty: Intermediate — requires log calculations and practical reasoning.

Exercise 5: Tracing a Digital Signature Workflow

Scenario: A decentralized payment network processes the following transaction:

> **Transaction:** "Transfer 0.3 ETH from `0xABC...` to `0xDEF...`"
> **Sender's public key:** `0xABC_pub`
> **Signature:** `sig_7f2a...`

Tasks:

a. Describe the three steps the sender performed to create `sig_7f2a...` (be specific about which keys and algorithms are used at each step).

b. Describe the three steps a network node performs to verify the signature.

c. An attacker intercepts the transaction and changes the amount from 0.3 ETH to 3.0 ETH. Explain, step by step, why the verification process will reject this modified transaction.

d. The attacker instead replays the *original* unmodified transaction a second time. Does the signature still verify? What mechanism (beyond cryptography) prevents this replay attack?

Difficulty: Advanced — requires tracing multi-step cryptographic processes.

Exercise 6: Private Key Compromise Scenarios

Scenario: Consider three separate incidents at a cryptocurrency exchange:

1. **Incident A:** A hot wallet's private key is stolen by a hacker.
2. **Incident B:** A cold storage private key is accidentally destroyed (hardware failure, no backup).
3. **Incident C:** An employee's public key is leaked to the press.

Tasks:

a. For each incident, classify the impact as: *funds at risk of theft*, *funds permanently inaccessible*, or *no security impact*. Justify each answer.

b. Incident A: The exchange detects the breach within 10 minutes. What actions can they take? What actions are impossible once a private key is compromised?

c. Incident B: The cold storage held 500 BTC. Using the current BTC price, estimate the loss. Explain why this loss is *fundamentally irrecoverable* in a decentralized system (contrast with how a bank would handle this).

d. Propose a key management policy that mitigates *both* Incident A and Incident B simultaneously. Explain the trade-offs.

Difficulty: Advanced — requires security analysis and policy reasoning.

Exercise 7: Designing and Defeating a Double-Spend Attack

Scenario: Mallory holds 1 BTC in a system that uses digital signatures and a Merkle tree of transactions but has **no consensus mechanism**. Each node independently maintains its own copy of the transaction ledger.

Tasks:

a. Describe, step by step, how Mallory can execute a double-spend attack by sending 1 BTC to both Alice and Bob simultaneously. Show that both transactions are cryptographically valid.

b. Explain why the Merkle tree cannot resolve this conflict.

c. Explain why digital signatures cannot resolve this conflict.

d. Now suppose a consensus mechanism is introduced: a randomly selected leader node determines the "official" transaction ordering every 10 minutes. How does this prevent Mallory's attack? What new assumption does it introduce?

e. Identify one weakness of the single-leader approach and propose an improvement (hint: think about what Bitcoin does differently).

Difficulty: Advanced — requires synthesizing multiple concepts and evaluating system design.

Exercise 8: Design a Tamper-Evident Financial Audit System

Scenario: A regulator asks you to design a tamper-evident system for auditing quarterly financial statements from 200 banks. Requirements:

- Each bank submits a quarterly report (variable size, typically 5–50 pages)
- The regulator must be able to detect any post-submission modification
- Any bank must be able to verify that its own report is included in the audited set without seeing other banks' reports
- The system must work without trusting any single party (including the regulator)

Tasks:

a. Describe how you would use SHA-256 to create a tamper-evident fingerprint of each report.
b. Describe how you would use a Merkle tree to organize the 200 report hashes. How deep is the tree?
c. Describe the inclusion proof protocol: what does the regulator publish, and what does each bank verify?
d. How would you use digital signatures so that each bank can prove *it* submitted the report (not an impersonator)?
e. Identify one remaining vulnerability of your design and propose a mitigation using one of the cryptographic tools from this lesson.

Difficulty: Advanced–Integrative — combines all lesson concepts in a system design.

Answer Key

Exercise 1:

(a) Attack A: pre-image resistance. Attack B: collision resistance. Attack C: avalanche effect.

(b) A: Pre-image resistance makes it infeasible to find any input that maps to a given hash. B: Collision resistance makes it infeasible to find two different inputs with the same hash. C: The avalanche effect ensures even a 0.1% change completely alters the hash, so no "close enough" output exists.

(c) CRC-32 lacks all three cryptographic properties. Attacks A and B become feasible because CRC-32 collisions are trivial to find and pre-images can be constructed. Attack C also succeeds because CRC-32 does not exhibit a strong avalanche effect.

Exercise 2:

(a) Students compute via tool. Hashes will be completely different.

(b) Typically 12–14 of 16 hex characters differ ($\approx$75–88%), consistent with $\sim$50% bit flipping (some hex chars may partially match while their underlying bits differ).

(c) Supports the claim: about half the underlying bits flip, even though the hex-level comparison may show a different percentage due to the 4-bit grouping.

(d) Yes, the empty string also produces exactly 64 hex characters (256 bits). The output size is always fixed regardless of input size.

Answer Key (continued)

Exercise 3:

(a)–(b) Students compute intermediate hashes: $H_{12} = \text{Hash}(\texttt{a1b2} \| \texttt{c3d4})$, $H_{34} = \text{Hash}(\texttt{e5f6} \| \texttt{0789})$, Root $= \text{Hash}(H_{12} \| H_{34})$. Exact values depend on the SHA-256 computations.

(c) To prove T_3: need sibling $H_4 = 0789$ (to compute H_{34}) and sibling H_{12} (to compute root). Total: 2 hashes.

(d) Changing T_2 changes H_{12} and the root. H_{34} remains unchanged because it depends only on T_3 and T_4.

Exercise 4:

(a) 16: $\log_2(16) = 4$ hashes. 256: $\log_2(256) = 8$ hashes. 2^{20}: $\log_2(2^{20}) = 20$ hashes.

(b) $20 \times 32 = 640$ bytes for the Merkle proof.

(c) Flat list: $1{,}048{,}576 \times 32 = 33{,}554{,}432$ bytes ($\approx$32 MB). Merkle proof: 640 bytes. The Merkle proof is $\approx$52,000$\times$ smaller.

(d) Mobile devices have limited bandwidth and storage. Downloading 32 MB per block verification is impractical; 640 bytes is trivial even on a slow connection. Logarithmic scaling means the proof grows negligibly as the ledger grows.

Answer Key (continued)

Exercise 5:

(a) (1) Serialize the transaction data. (2) Compute SHA-256 hash of the serialized data. (3) Sign the hash with the sender's private key using ECDSA, producing the signature.

(b) (1) Serialize the transaction data the same way. (2) Compute SHA-256 hash. (3) Use ECDSA verify with the sender's public key, the hash, and the signature. If it returns valid, accept.

(c) The modified transaction has a different hash (avalanche effect). The original signature was computed over the original hash. ECDSA verify with the new hash and the old signature will return invalid.

(d) The signature still verifies (it matches the unchanged transaction). Replay attacks are prevented by transaction-level mechanisms: nonces (sequential counters) or UTXO tracking, not by cryptography alone.

Exercise 6:

(a) A: funds at risk of theft (attacker can sign transactions). B: funds permanently inaccessible (no one can sign). C: no security impact (public keys are meant to be public).

(b) Within 10 minutes: transfer remaining funds to a new address with a new key pair. Impossible: revoke the stolen key (no central authority), undo transactions already signed by the attacker.

(c) At any BTC price, the 500 BTC is irrecoverable. No entity can regenerate the private key or override the blockchain. A bank would simply update its database and restore the account.

(d) Multi-signature wallets (e.g., 2-of-3): no single key compromise loses funds (mitigates A), and losing one key does not lock funds (mitigates B). Trade-off: slower transactions, more complex key management.

Answer Key (continued)

Exercise 7:

(a) Mallory signs "Pay Alice 1 BTC" and "Pay Bob 1 BTC" with her private key. Both signatures are valid. She sends each to different nodes at the same time.

(b) The Merkle tree organizes transactions within a block but does not decide *which* transactions to include. Two nodes could build different trees containing different conflicting transactions.

(c) Both signatures are cryptographically valid. Signatures prove who authorized the transaction, not which transaction should take priority.

(d) The leader selects one transaction (e.g., Alice's) and rejects the other. All nodes follow the leader's ordering. New assumption: the leader is honest.

(e) A single leader can be bribed or compromised. Bitcoin improves this with Proof of Work: the leader is selected by solving a computational puzzle, making it costly to cheat and rotating leadership unpredictably.

Exercise 8:

(a) Each bank computes $h_i = \text{SHA-256}(\text{report}_i)$. Any modification changes h_i.

(b) 200 leaves $\Rightarrow$ pad to 256 leaves. Tree depth: $\log_2(256) = 8$ levels.

(c) The regulator publishes the Merkle root. Each bank receives the $\log_2(256) = 8$ sibling hashes for its leaf. The bank recomputes the path to the root and verifies it matches the published root.

(d) Each bank signs its report hash with its private key before submission: $\text{sig}_i = \text{Sign}(sk_i, h_i)$. The regulator stores $(\text{sig}_i, h_i, pk_i)$. Anyone can verify the bank's authorship.

(e) Vulnerability: the regulator could publish a false Merkle root or omit a bank's report. Mitigation: publish the Merkle root to multiple independent parties (or a blockchain) so no single party can alter it after publication. Alternatively, require the regulator to sign the root, providing non-repudiation.