Module 3 Summary: The Trust Problem

Prof. Dr. Joerg Osterrieder

Digital Finance — BSc Course

Module 3: Prerequisites

Theme: Trust (cryptography, consensus, smart contracts, DeFi) **From prior modules:**

- **M1L2:** why intermediaries exist economically
- **M2L2:** credit and counterparty risk language

External knowledge assumed:

- Hash-function intuition (one-way, collision-resistant) at the level of one paragraph, not implementation
- Modular arithmetic basics for elliptic-curve preliminaries
- Basic probability for double-spend, finality, and security analysis

Will be introduced this module: We build the trust stack from the bottom: cryptographic primitives, consensus mechanisms, smart contracts, and DeFi stablecoins. Forward references to M6 (payment rails) are flagged as not-yet-introduced.

Prerequisites are advisory; lessons remain self-contained where feasible. Forward references inside lessons flag any concept used before its canonical introduction.

Key Concepts by Lesson

L1: Cryptographic Foundations

- Hash functions: deterministic, fixed-length, one-way
- SHA-256: 2^{256} possible outputs, avalanche effect
- Merkle trees: efficient transaction verification via inclusion proofs

L2: Consensus Mechanisms

- Byzantine Fault Tolerance: $n \geq 3f + 1$
- Proof of Work: SHA-256(header $\|$ nonce) $<$ target
- Proof of Stake: validators lock capital as collateral
- Blockchain trilemma: security, decentralization, scalability

L3: Smart Contracts & dApps

- EVM: deterministic execution across all nodes
- Gas model: 21,000 (transfer) to 500,000+ (DeFi)
- Token standards: ERC-20 (fungible), ERC-721 (NFT)
- Layer-2 scaling: rollups, sidechains, state channels

L4: DeFi & Stablecoins

- AMM constant product: $x \cdot y = k$
- Stablecoin types: fiat-backed, crypto-collateralized, algorithmic
- Flash loans, reentrancy attacks, oracle manipulation

Module 3 answers: How can strangers transact without a trusted intermediary?

Key Formulas and Definitions

Byzantine Fault Tolerance Threshold

$$n \geq 3f + 1 \quad \text{where } n = \text{total nodes},\ f = \text{faulty/malicious nodes}$$

Proof of Work Puzzle

$$\text{SHA-256}(\text{block\_header} \| \text{nonce}) < \text{target}$$

Miners iterate over nonce values until a valid hash is found. Difficulty adjusts to maintain constant block time.

Constant Product AMM (Uniswap)

$$x \cdot y = k \qquad \text{Price: } P = \frac{x}{y}$$

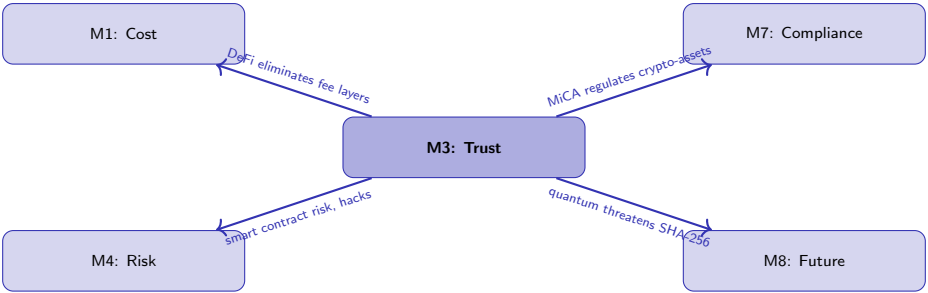
Larger trades cause greater slippage. The AMM never runs out of either token — price approaches infinity.

Merkle Tree Verification

Verify a single transaction in $O(\log n)$ hash computations (vs. $O(n)$ for a flat list). Only the **inclusion proof** (sibling hashes along the path) is needed, not the full block data.

Cryptography + consensus + smart contracts + DeFi = a complete financial system without banks.

Connections to Other Modules



- **Trust → Cost (M1):** DeFi protocols replace intermediaries with code, collapsing the MDR fee stack
- **Trust → Risk (M4):** Smart contract bugs, oracle failures, and flash loan exploits create a new category of financial risk
- **Trust → Compliance (M7):** MiCA classifies crypto-assets (EMTs, ARTs, utility tokens) and imposes reserve/disclosure requirements
- **Trust → Future (M8):** Quantum computers running Shor's algorithm could break elliptic-curve signatures underlying all blockchains

Blockchain replaced trust in institutions with trust in mathematics — but the code still needs auditing and the math faces a quantum threat.

Module 3: Integrative Questions

Two questions that need more than one lesson to answer:

1. M3L2 (consensus) secures Bitcoin; M3L3 (smart contracts) underpins DeFi (M3L4). Name one vulnerability that breaks both layers simultaneously and explain the mechanism.
2. Compare The DAO 2016 (M3L3) with Terra-Luna 2022 (M3L4): same root cause class or different? Use specific lesson concepts in the answer.

Use these as study prompts before the module exam; each integrates concepts that span lessons.

Module 3: Worked Multi-Lesson Example

Worked example: a student deploys an ERC-20 stablecoin. Trace M3L1 (hash function for contract address), M3L2 (consensus finality assumptions), M3L3 (smart-contract reentrancy guard), M3L4 (liquidity-pool peg defence). Identify the weakest link. **Pedagogical pattern:** the example is intentionally end-to-end. Solve it lesson-by-lesson, then step back and identify the lesson whose assumption was the binding constraint.

The exam-style version of this example appears in extttv4/exam_prep/exam_bank.tex for module 3.

Module 3: Forward-Pointing Dependency Map

Concepts from Module 3 that later modules will use:

- **M4L3:** M3L4 DeFi liquidations feed into M4L3 institutional-risk: counterparty + leverage map across protocols
- **M6L4:** M3L2 consensus underlies the next-gen-infrastructure design choices in M6L4 (DLT settlement vs CBDC)
- **M7L4:** Smart-contract audit (M3L3) is a model-risk-governance control (M7L4) for institutions adopting DeFi

---

Forward-pointing dependencies; concepts not in this map are local to Module 3.