Module 3: Cryptographic Foundations

Building Trust Without Institutions

Joerg Osterrieder
Digital Finance

The Big Question

How can two strangers, who have never met and have no reason to trust each other, agree on who owns what—without any bank, government, or company standing in between?

This is not a hypothetical puzzle. It is the central problem that every financial system in human history has tried to solve. For thousands of years the answer was the same: find a trusted middleman. Temples kept grain tallies in Mesopotamia. Goldsmiths issued receipts in medieval Europe. Today, banks maintain ledgers, courts enforce contracts, and governments back currencies. Every one of these solutions depends on a single assumption—that the middleman will behave honestly.

Cryptography offers a radically different answer. Instead of trusting an institution, you verify the mathematics. Instead of relying on a judge to enforce a contract, you rely on a digital signature that cannot be forged. Instead of hoping a bank will not alter your balance, you rely on a hash function that makes tampering instantly detectable. The trust shifts from people and organisations to algorithms and proofs.

Module 3 takes you on a journey through that shift. You will learn the three cryptographic primitives that make trustless systems possible, see how they combine into the machinery of a blockchain, discover the practical challenges users face when interacting with that machinery, and compare the two most influential blockchain designs ever created. By the end, you will understand not only *how* these systems work, but *why* they were designed the way they were—and what they sacrifice in return for removing the middleman.

The Story

Two Strangers, No Middleman

Imagine two people on opposite sides of the world. They have never met. They speak different languages, live under different legal systems, and have no mutual friends. One of them owns a digital asset and wants to sell it. The other wants to buy it. There is no bank to verify identities, no court to enforce the deal, and no escrow agent to hold the funds. How can they possibly trade?

The first problem is **identity**. The buyer needs to know that the seller actually controls the asset being sold, and the seller needs to know that the payment, once received, cannot be faked. In the physical world you might show a passport or hand over cash. In a digital world, copying is trivial—so how do you prove who you are without revealing a secret that someone else could steal?

The answer is **public-key cryptography**. Each person generates a pair of mathematically linked keys: one kept secret, one shared freely. The secret key can produce a **digital signature** that anyone can verify using the public key, but that nobody else can forge. It is as if each person carries an unforgeable stamp that proves their approval of any message they sign. No passport office required.

The second problem is **integrity**. Suppose the seller signs a message saying "I transfer this asset to the buyer." How does the buyer know that nobody—not the seller, not a hacker, not a corrupt server—altered the message after it was signed? If even a single character changes, the entire deal could be different.

The answer is **hash functions**. A hash function takes any input—a word, a book, an entire database—and produces a fixed-size digital fingerprint. Change even one character of the input and the fingerprint changes completely. By hashing the transaction message before signing it, the seller locks in the exact content. Any tampering is immediately visible because the fingerprint no longer matches.

The third problem is **agreement**. The buyer and the seller can now identify each other and detect tampering, but they still need a shared record that both trust. Who maintains that record? A single server could be hacked. A single company could go offline. And neither party wants to trust the other to keep it.

The answer is **consensus**. Thousands of independent computers around the world each keep a copy of the same ledger. When a new transaction arrives, these computers follow a set of rules—a **consensus mechanism**—to agree on whether the transaction is valid and in what order it should appear. In one design, computers compete to solve a computational puzzle before earning the right to add a block of transactions (**Proof of Work**). In another, participants lock up their own assets as collateral, and the network randomly selects one to propose the next block (**Proof of Stake**). Either way, no single party controls the ledger.

The fourth problem is **usability**. Our two strangers now have the cryptographic tools and the shared ledger, but actually using them is another matter entirely. The seller must safeguard a private key—a long string of random data—because losing it means losing the asset forever. The buyer must pay a transaction fee, quoted in an unfamiliar unit, that fluctuates with network demand. Sending funds to a wrong address is irreversible. There is no "forgot password" link, no customer support hotline. The system that eliminates the middleman also eliminates the safety net.

This is the **UX gap**: the distance between the elegant mathematics under the hood and the bewildering experience of an ordinary person trying to use it. Wallets, seed phrases, gas fees, confirmation times—these are not flaws in the design, but consequences of the design's

core principle: you are your own bank.

The fifth problem is **design philosophy**. Once you accept that a decentralised ledger is possible, you still have to decide what it should do. Should it do one thing extremely well—record transfers of a scarce digital asset—and resist all pressure to change? Or should it be a general-purpose platform where anyone can deploy programs that execute automatically?

The first path leads to something that looks like **digital gold**: conservative, minimal, optimised for security and scarcity. The second path leads to something that looks like a **world computer**: flexible, ambitious, optimised for programmability and innovation. Neither is right or wrong. Each makes a different bet about what matters most.

By the end of their journey, our two strangers have completed their trade. No bank verified their identities—cryptographic keys did. No court enforced the deal—a digital signature did. No company maintained the ledger—a global network of nodes did. And the design of the system they used—whether conservative or ambitious—shaped every aspect of their experience.

Why This Matters

If you work in finance, technology, regulation, or any field that touches money, the ideas in this module are not optional background reading. They are the foundation on which an entire parallel financial system is being built.

Hash functions are the reason blockchain history is tamper-evident. If you understand why changing one bit of data produces a completely different fingerprint, you understand why a blockchain ledger is credible even without a central bookkeeper.

Public-key cryptography is the reason digital ownership is possible without identity documents. If you understand the one-way relationship between a private key and a public key, you understand why "not your keys, not your coins" is not a slogan but a mathematical fact.

Consensus mechanisms are the reason thousands of strangers can agree on a single version of the truth. If you understand the trade-offs between Proof of Work and Proof of Stake, you can evaluate any blockchain's security claims with a critical eye rather than relying on marketing.

Wallets and transactions are the interface between users and all of the above. If you understand gas fees, seed phrases, and the UTXO model, you can assess whether a blockchain product is genuinely usable—or whether it hides complexity that will cost users dearly when something goes wrong.

Finally, the **Bitcoin-versus-Ethereum comparison** is the clearest illustration of how design philosophy shapes a technology's strengths and weaknesses. If you understand why one project chose limited scripting and a fixed supply while the other chose a Turing-complete virtual machine and dynamic issuance, you can reason about any new blockchain project that comes along—because they all sit somewhere on the same spectrum.

> Cryptography does not eliminate trust. It shifts trust from institutions you must believe to mathematics you can verify. That shift has consequences—both liberating and dangerous—

that every participant in digital finance needs to understand.

Cryptographic Building Blocks

Topic 3.1 introduces the three atomic primitives on which every blockchain is built: hash functions, public-key cryptography, and digital signatures. The slides deliberately focus on what these tools *guarantee* rather than on the mathematics behind them—a design choice that reflects the course's promise of no prerequisites in mathematics.

Hash Functions: Digital Fingerprints

A **hash function** takes any input—a single word, a multi-gigabyte file—and produces a fixed-size output called a *hash* or *digest*. The standard used in most blockchains, SHA-256, always produces a string of fixed length regardless of how large or small the input is.

Five properties make hash functions useful for building trust:

1. **Deterministic.** The same input always produces the same output.
2. **Fixed output size.** The hash is always the same length.
3. **One-way.** Given a hash, you cannot compute the original input.
4. **Collision-resistant.** It is practically impossible to find two different inputs that produce the same hash.
5. **Avalanche effect.** Changing even one character of the input produces a completely different output.

These properties combine to create a reliable "digital fingerprint" system. In a blockchain context, hash functions are used to link blocks together (each block contains the hash of its predecessor), to create unique transaction identifiers, and to build **Merkle trees**—hierarchical structures that let you verify any single transaction without downloading the entire ledger.

Public-Key Cryptography: Identity Without Authority

Public-key cryptography solves a deceptively simple problem: how can two strangers communicate securely without first meeting to exchange a secret password? The answer is a pair of mathematically linked keys. One key is kept private; the other is shared publicly. Data encrypted with the public key can only be decrypted with the private key, and vice versa.

In blockchain, the private key is your identity. Whoever holds it controls the associated funds. The public key (or an address derived from it) is your "account number," safe to share with anyone. The relationship is one-way: deriving a public key from a private key is trivial, but reversing the process is computationally infeasible.

Blockchains use a specific flavour of public-key cryptography called ECDSA (Elliptic Curve Digital Signature Algorithm), which provides the same security as older schemes but with much smaller key sizes—an important advantage when every byte costs transaction fees.

Digital Signatures: Unforgeable Proof

A **digital signature** combines the previous two primitives. To sign a message, you first hash it (producing a fixed-size digest) and then encrypt that digest with your private key. Anyone can verify the signature by decrypting it with your public key and comparing the result to a fresh hash of the message.

Digital signatures provide three guarantees simultaneously:

- **Authentication.** Only the holder of the private key could have created the signature.
- **Integrity.** Any change to the message invalidates the signature.
- **Non-repudiation.** The signer cannot later deny having signed.

Every blockchain transaction carries a digital signature. It is what proves that you—and only you—authorised a transfer.

> Think of these three primitives as a stack. Hash functions guarantee that data has not changed. Public-key cryptography guarantees identity. Digital signatures combine both to guarantee authorisation. Together, they replace "trust the institution" with "verify the math."

Blockchain Mechanics

Topic 3.2 takes the cryptographic primitives from the previous topic and assembles them into a working system. A blockchain is, at its core, a distributed ledger that is append-only, cryptographically linked, and maintained by consensus. The slides unpack each of those adjectives.

Block Structure and Hash Linking

Every block contains a set of transactions, a timestamp, a reference to the previous block's hash, a nonce (a number used in mining), and its own hash—which is computed from all the other fields. Because the hash of one block is embedded in the next, changing any data in a past block would invalidate every subsequent block in the chain. This cascading invalidity is what makes blockchain history effectively immutable: an attacker would have to recalculate not just the target block, but every block that follows, faster than the rest of the network adds new ones.

The very first block in any chain—the **genesis block**—has no predecessor, so its "previous hash" field is typically all zeros.

Consensus Mechanisms

The deepest challenge in decentralised systems is not data structures but agreement. How do thousands of computers, some of which may be faulty or malicious, converge on a single version of the truth?

Proof of Work answers this question with a computational lottery. Miners race to find a nonce that produces a block hash meeting a difficulty target (typically a hash that starts with a certain number of zeros). The process is deliberately expensive—requiring vast amounts of electricity—so that attacking the network is economically irrational.

Proof of Stake answers the same question with economic collateral. Validators lock up cryptocurrency as a deposit. If they behave honestly, they earn rewards. If they cheat, their deposit is "slashed." The result is similar security at a fraction of the energy cost.

Neither mechanism is universally superior. Proof of Work is battle-tested and truly decentralised but energy-intensive. Proof of Stake is efficient and enables faster finality but introduces different centralisation risks. The choice between them is one of the defining design decisions any blockchain project must make.

The Blockchain Trilemma

Every blockchain must balance three properties: **decentralisation**, **security**, and **scalability**. The trilemma states that you can optimise for at most two of these at the expense of the third. One blockchain may be highly decentralised and secure but process only a handful of transactions per second. Another may be fast and secure but rely on a small set of validators. A traditional database may be fast and centralised but offer no decentralisation at all.

Layer 2 solutions—protocols built on top of a base blockchain—attempt to break the trilemma by handling transactions off-chain and settling periodically on the main chain. The Lightning Network (for Bitcoin) and rollups like Optimism and Arbitrum (for Ethereum) are prominent examples.

> The trilemma is not a temporary engineering bottleneck. It reflects a fundamental trade-off: the more computers that must verify every transaction, the slower and more expensive the system becomes. Understanding this trade-off is essential for evaluating any blockchain's claims about speed, security, or decentralisation.

Wallets and Transactions

Topic 3.3 shifts from the system's internal machinery to the experience of the people who use it. The transition is jarring. The elegant mathematics described in earlier topics translates into an interface that routinely punishes small mistakes with permanent, irreversible losses.

What a Wallet Actually Is

A common misconception is that a cryptocurrency wallet "stores" your coins. In reality, a **wallet** is a key-management tool. It stores your private keys, uses them to sign transactions, and broadcasts those transactions to the network. Your balance is not inside the wallet; it is recorded on the blockchain. The wallet simply proves you have the right to spend it.

Wallets sit on a spectrum of security versus convenience:

- **Custodial wallets** (exchanges) hold keys on your behalf. They are easy to use—password resets, customer support, familiar interfaces—but introduce counterparty risk. If the exchange is hacked or goes bankrupt, your funds may be lost.
- **Hot wallets** (browser extensions, mobile apps) give you control of your keys but keep them on an internet-connected device, making them vulnerable to malware and phishing.
- **Cold wallets** (paper, air-gapped computers) store keys offline, removing remote attack vectors but adding friction to every transaction.

- **Hardware wallets** (dedicated devices) strike a balance: keys are generated and stored on a secure chip that never exposes them to the internet, while a small screen lets you verify transactions before signing.

The **seed phrase**—a list of twelve to twenty-four words generated when you create a wallet—is the master backup from which all private keys can be regenerated. Lose it, and there is no recovery. Share it, and your funds are gone.

Transaction Anatomy and Gas

Every blockchain transaction includes a sender, a recipient, an amount, and a digital signature. On Ethereum, transactions also carry a **gas** budget—a measure of the computational work required to process them.

Gas exists for a practical reason: because Ethereum's virtual machine is Turing-complete, code could theoretically run forever. Gas limits prevent that. You specify a maximum amount of gas you are willing to pay, and validators stop executing once that limit is reached. The total fee is the gas consumed multiplied by the gas price, which fluctuates with network demand.

A transaction's lifecycle runs from creation through signing, broadcasting to the peer-to-peer network, waiting in the **mempool** (a queue of unconfirmed transactions), inclusion in a block by a validator, and finally confirmation as additional blocks are added on top.

The UX Gap

Traditional banking is forgiving. You can reset a password, dispute a fraudulent charge, and call customer support at any hour. Cryptocurrency is unforgiving. There is no "forgot password" link for a seed phrase. Sending funds to the wrong address is irreversible. Approving a malicious smart contract can drain your entire wallet.

This gap between the system's mathematical elegance and its practical usability is one of the largest barriers to mainstream adoption. Efforts to close it—account abstraction, social recovery, human-readable addresses, transaction simulation—are active areas of development, but the fundamental tension remains: the more you protect users from mistakes, the more you reintroduce intermediaries and trust assumptions.

> "Not your keys, not your coins" is the defining principle of self-custody. It means that if someone else holds your private keys—an exchange, a custodian, a friend—they control your funds, no matter what your account screen displays. This is not a theoretical warning; high-profile exchange failures have demonstrated it in practice.

Bitcoin vs. Ethereum

Topic 3.4 presents the two most influential blockchain designs as embodiments of different philosophies rather than as competitors fighting for the same market.

Digital Gold vs. World Computer

Bitcoin's design philosophy is conservative and minimalist. Its scripting language is deliberately limited—intentionally *not* Turing-complete—because every additional capability is an additional

attack surface. Its supply is capped at a fixed number of coins, enforced by a **halving** mechanism that cuts the block reward at regular intervals. Changes to the protocol require years of debate and broad consensus. The result is a system optimised for one thing: being a reliable, censorship-resistant store of value.

Ethereum's design philosophy is progressive and ambitious. Its virtual machine (the **EVM**) is Turing-complete, meaning it can run arbitrary programs—called **smart contracts**—that execute automatically when conditions are met. Its supply model is dynamic, and the protocol undergoes regular upgrades. The result is a general-purpose platform that enables decentralised finance, non-fungible tokens, decentralised autonomous organisations, and applications that have not yet been imagined.

Key Technical Differences

Several technical contrasts follow from these philosophies:

- **Consensus.** Bitcoin uses Proof of Work; Ethereum transitioned to Proof of Stake, reducing energy consumption dramatically.
- **Block time.** Bitcoin produces a block roughly every ten minutes; Ethereum produces one roughly every twelve seconds.
- **Transaction model.** Bitcoin uses a UTXO (Unspent Transaction Output) model, which is analogous to cash bills that must be spent whole with change returned. Ethereum uses an account model, which is analogous to a bank balance that can be debited by any amount.
- **Programmability.** Bitcoin Script can answer simple yes-or-no questions ("Is this signature valid?"). The EVM can execute arbitrary logic ("Run this lending protocol, check the collateral ratio, and liquidate if necessary").
- **Development pace.** Bitcoin's roadmap is measured in years per change. Ethereum's roadmap includes regular hard forks and major protocol upgrades.

The Flexibility-Security Trade-off

More programmability means more capability—but also more things that can go wrong. Bitcoin's protocol has never been successfully exploited. Ethereum's smart contracts have suffered numerous high-profile hacks, the most famous being the DAO exploit that led to a controversial hard fork splitting the chain into two.

This trade-off is not a flaw in either design. It is a fundamental tension: the more expressive a system's programming language, the larger its attack surface. Bitcoin accepts limited functionality in exchange for a smaller target. Ethereum accepts greater risk in exchange for the power to build entirely new financial instruments in code.

Layer 2 and Convergence

Both blockchains face the same scaling bottleneck—neither can match the throughput of traditional payment networks. Both address this through Layer 2 solutions: the Lightning Network for Bitcoin (focused on payments) and rollups for Ethereum (capable of handling any application). Cross-chain bridges like Wrapped Bitcoin attempt to combine Bitcoin's store-of-value

properties with Ethereum's programmability, though they introduce their own trust assumptions and risks.

> Bitcoin and Ethereum are not competing for the same role. Bitcoin is designed to be money you can trust because it does not change. Ethereum is designed to be a platform you can build on because it does. Understanding when to use which—and why—is one of the most practical skills in digital finance.

Hands-On Highlights

Three Jupyter notebooks accompany Module 3, each building on the previous one:

- `NB05` — **Cryptographic Operations.** You compute SHA-256 hashes and observe the avalanche effect firsthand. You generate a public-private key pair, derive an address, and then sign and verify a message. By the end, you have a concrete, hands-on understanding of the three cryptographic primitives.
- `NB06` — **Blockchain Simulation.** You build a simple blockchain from scratch: a block data structure, a chain that links blocks via hash pointers, and a mining function that searches for a valid nonce. You then tamper with a block and watch the chain validation fail, making the concept of immutability viscerally real.
- `NB07` — **Blockchain Transactions.** You construct a transaction with all required fields, calculate gas fees under different network conditions, compare the UTXO and account models through worked examples, and simulate a fee market to see how supply and demand determine the cost of block space.

All three notebooks run in the browser via Google Colab—no installation required.

Note: An optional advanced notebook (`NB15` — Layer 2 Simulation) and an additional slide set (T3.5 — Layer 2 Scaling) are available for students who want to go deeper into scaling solutions.

Key Takeaways

1. **Hash functions are the foundation of data integrity.** They create unique digital fingerprints that make any tampering instantly detectable. In a blockchain, they link blocks into an immutable chain and organise transactions into efficiently verifiable Merkle trees.
2. **Public-key cryptography enables ownership without intermediaries.** A private key is your identity; a public key (or its derived address) is your account number. The one-way mathematical relationship between them means anyone can verify your identity, but nobody can impersonate you.
3. **Digital signatures provide authentication, integrity, and non-repudiation simultaneously.** Every blockchain transaction carries a signature that proves who authorised it, that it has not been altered, and that the signer cannot deny having signed.
4. **Consensus mechanisms solve the agreement problem in different ways.** Proof of Work spends energy; Proof of Stake stakes capital. Both create economic costs for attackers,

but they make different trade-offs on efficiency, speed, and centralisation risk.

5. **The blockchain trilemma is a real constraint, not a temporary limitation.** No blockchain can simultaneously maximise decentralisation, security, and scalability. Layer 2 solutions mitigate but do not eliminate this tension.
6. **Wallets are key-management tools, not coin storage.** Understanding the difference between custodial and non-custodial wallets—and the responsibilities that come with self-custody—is essential for anyone handling digital assets.
7. **Bitcoin and Ethereum embody different design philosophies.** Bitcoin prioritises simplicity, scarcity, and immutability. Ethereum prioritises programmability, flexibility, and rapid innovation. Neither is universally superior; each is optimised for different use cases.

Looking Ahead

Module 3 gave you the *infrastructure*—the cryptographic primitives, the consensus layer, the wallet interface, and the two dominant designs. Module 4 asks: *what can you build on top of it?*

The answer is **programmable finance**. In Module 4, you will explore smart contracts in depth: how they work, what they enable, and what can go wrong. You will study the core building blocks of decentralised finance—automated market makers, lending protocols, and decentralised exchanges—and see how they compose into complex financial instruments without any central operator. You will examine **stablecoins**, the bridge between volatile crypto assets and the stable values that everyday commerce requires. And you will look at **tokenisation** and **central bank digital currencies**—the points where the decentralised world of blockchain begins to intersect with the regulated world of traditional finance.

If Module 3 answered "How do you build trust without institutions?", Module 4 answers "Now that you have trust, what financial services can you create from code alone?"