BITCOIN ACADEMY

ADVANCED

Bitcoin Mastery

For those who want to truly understand the protocol — cryptography, scripting, node operation, and monetary theory.

---

8 Modules | ~6 Hours | Course Module 03
bitcoinacademy.online

Contents

MODULE 01

Elliptic Curve Cryptography

At the heart of every Bitcoin transaction is **elliptic curve cryptography (ECC)**. Bitcoin uses a specific curve called **secp256k1**, defined by the equation $y^2 = x^3 + 7$ over a finite field. This mathematical structure makes it easy to compute a public key from a private key, but computationally infeasible to reverse the process.

Your **private key** is simply a random 256-bit number — one of roughly 10^{77} possible values (more than the number of atoms in the observable universe). Your **public key** is derived by multiplying the private key by the curve's generator point G using elliptic curve point multiplication.

THE TRAPDOOR FUNCTION

ECC is a "trapdoor function" — easy in one direction, virtually impossible in reverse. Given private key k, computing public key K = k x G takes microseconds. But given K and G, finding k would take longer than the age of the universe with current computing. This asymmetry is the foundation of all Bitcoin security.

From public key to address: Bitcoin doesn't directly use the raw public key as an address. Instead, it applies SHA-256 hashing followed by RIPEMD-160 (collectively called Hash160), then adds a version byte and checksum to create a Base58Check-encoded address. This additional hashing provides a second layer of protection — even if ECC were somehow broken, your funds would still be protected by the hash functions.

QUANTUM RESISTANCE

Shor's algorithm running on a sufficiently powerful quantum computer could theoretically break ECC. However, Bitcoin addresses that have never spent (only received) are protected by the hash layer — an attacker would need to break both ECC and SHA-256. The Bitcoin community is actively researching post-quantum signature schemes for future soft-fork integration.

MODULE 02

Bitcoin Script & Transaction Types

Bitcoin transactions aren't just "send X to Y." They're programmable using **Bitcoin Script** — a stack-based, Forth-like language that defines the conditions under which Bitcoin can be spent. Every UTXO is locked with a script (scriptPubKey), and spending it requires providing a matching script (scriptSig).

Script is intentionally limited: it's **not Turing-complete** (no loops), which prevents infinite execution and makes transaction validation predictable and safe. This is a deliberate design choice — security over flexibility.

Type	Script	Use Case
P2PKH	Pay-to-Public-Key-Hash	Classic addresses (1...)
P2SH	Pay-to-Script-Hash	MultiSig, complex (3...)
P2WPKH	Pay-to-Witness-PKH	Native SegWit (bc1q...)
P2WSH	Pay-to-Witness-Script-Hash	SegWit script hashes
P2TR	Pay-to-Taproot	Taproot (bc1p...)

HOW SCRIPT EXECUTES

When you spend Bitcoin, the network combines your scriptSig (the "key") with the UTXO's scriptPubKey (the "lock") and executes them on a stack machine. If the final result is TRUE, the spend is valid. For a simple P2PKH transaction: push signature, push public key, duplicate, hash, compare, verify. Six opcodes, bulletproof security.

MODULE 03

Running & Configuring a Full Node

A full node is the ultimate expression of **Bitcoin sovereignty**. When you run a full node, you independently validate every transaction and every block against the consensus rules — you trust no one. You become an equal peer in the Bitcoin network.

What a full node does: it downloads and verifies the entire blockchain (~600GB as of 2026), validates every transaction against the consensus rules, relays valid transactions and blocks to other nodes, and rejects anything that violates the rules — regardless of who sent it or how much hash power supports it.

HARDWARE REQUIREMENTS

Minimum: Any modern computer with 2GB RAM, 1TB storage (SSD recommended), and a stable internet connection. A Raspberry Pi 4 with an external SSD is a popular low-cost option (~$150 total). Software: Bitcoin Core is the reference implementation. Download from bitcoincore.org, verify the PGP signatures, and let it sync (initial sync takes 1-3 days).

Why it matters for the network: Every full node is a vote for the consensus rules. During the 2017 block size wars, it was the full nodes — not the miners — that ultimately decided which chain was "Bitcoin." Miners proposed SegWit2x, but the economic majority of nodes rejected it. This was proof that Bitcoin's governance is decentralised.

PRUNED VS ARCHIVAL

If storage is a concern, you can run a "pruned" node that only keeps the most recent blocks (as little as 5GB) while still fully validating everything. You lose the ability to serve historical blocks to other nodes, but you gain the same security guarantees as an archival node. There's no excuse not to run one.

MODULE 04

Multi-Sig & Timelock Contracts

Bitcoin Script enables programmable money far beyond simple transfers. Two of the most powerful primitives are **multi-signature (multisig)** transactions and **timelocks** — and combining them unlocks sophisticated financial arrangements without intermediaries.

Multisig requires M-of-N signatures to spend. A 2-of-3 multisig means three keys exist, and any two must sign to authorise a transaction. This eliminates single points of failure — no single lost or stolen key can compromise your funds.

Timelocks add a time dimension to Bitcoin Script. CLTV (CheckLockTimeVerify) prevents a UTXO from being spent before a specific block height or timestamp. CSV (CheckSequenceVerify) enforces a relative delay from when the UTXO was created.

Together, these enable: inheritance planning (coins unlock after X years of inactivity), escrow arrangements, payment channels (the foundation of Lightning), and vesting schedules for organisational treasuries.

REAL-WORLD APPLICATION

Consider a Bitcoin inheritance plan: you create a 2-of-3 multisig where Key 1 is yours, Key 2 is your heir's, and Key 3 is a time-locked recovery key that activates after 1 year of inactivity. Normally, only you can spend (Key 1 + Key 2 with your cooperation). But if something happens to you, after one year your heir can use Key 2 + the now-unlocked Key 3. No lawyers, no probate, no trust required.

MODULE 05

Taproot, Schnorr & Future Upgrades

The **Taproot upgrade** (activated November 2021) was Bitcoin's most significant protocol change since SegWit. It introduced Schnorr signatures, Merkelized Alternative Script Trees (MAST), and a new address format (P2TR / bc1p...) — collectively enhancing privacy, efficiency, and smart contract capability.

Schnorr signatures replace ECDSA for Taproot transactions. They're mathematically simpler, provably secure, and critically support **key aggregation**: multiple public keys can be combined into a single key, and multiple signatures into a single signature. A 10-of-10 multisig becomes indistinguishable from a single-key spend on-chain — same size, same cost, complete privacy.

MAST (Merkelized Alternative Script Trees) allows complex spending conditions to be organised in a Merkle tree. Only the branch actually used is revealed on-chain. If you have a contract with 10 possible spending conditions, the blockchain only ever sees the one that was executed — the other 9 remain hidden.

KEY/SCRIPT PATH DUALITY

Every Taproot output has two spending paths: a key path (simple Schnorr signature, looks like any normal transaction) and a script path (reveals a MAST branch for complex conditions). In the cooperative case, parties use the key path — fast, cheap, and private. The script path is the fallback. This "optimistic execution" model means the common case is always the most efficient.

Future upgrades being researched: OP_CTV (covenant-style restrictions on how coins can be spent), cross-input signature aggregation (further reducing transaction sizes), and SIGHASH_ANYPREVOUT (enabling Eltoo, a next-generation Lightning channel design). Bitcoin development is slow and deliberate by design — each change undergoes years of peer review.

MODULE 06

Austrian Economics & Bitcoin

Bitcoin didn't emerge from a vacuum. Its design philosophy is deeply rooted in **Austrian economics** — a school of economic thought that emphasises individual action, subjective value, sound money, and skepticism of central planning. Understanding this intellectual foundation explains **why** Bitcoin works the way it does.

Sound money theory argues that money should be difficult to produce (scarcity), resistant to debasement (hard cap), and chosen freely by the market rather than imposed by decree. Gold served this role for millennia. Fiat currency abandoned these principles entirely. Bitcoin re-implements them digitally.

The Cantillon Effect: When new money is created (printed), those closest to the source (governments, banks, large institutions) benefit first by spending it before prices adjust. By the time new money reaches ordinary people, prices have already risen. This systematic wealth transfer from savers to those closest to the money printer is invisible but devastating over decades. Bitcoin's fixed supply eliminates this entirely.

TIME PREFERENCE

Austrian economics emphasises time preference — the degree to which people value present consumption over future consumption. Sound money (which holds or increases in value) encourages low time preference: saving, investing, thinking long-term. Inflationary money encourages high time preference: spending now because your money loses value tomorrow. Bitcoin, as a deflationary asset, structurally incentivises low time preference — arguably one of its most profound societal effects.

Hayek's denationalisation of money: In 1976, economist Friedrich Hayek proposed allowing private currencies to compete with government money, arguing competition would produce superior monetary systems. Bitcoin is the most successful implementation of this idea — a non-state money competing purely on its merits in a global market.

MODULE 07

Game Theory of Decentralised Systems

Bitcoin doesn't rely on anyone being honest — it creates conditions where **acting honestly is the most profitable strategy**. This is mechanism design: engineering the rules of a game so that self-interested behaviour produces a desired collective outcome.

Miner incentives: Miners spend enormous amounts on hardware and electricity. Their only way to recoup costs is by finding valid blocks. An invalid block is rejected by nodes and earns nothing — wasting all that energy. Therefore, rational miners always produce valid blocks. The more they invest, the more they have to lose by cheating. This is **Nakamoto Consensus**: security through economic incentive.

The 51% attack paradox: An entity controlling >50% of hash power could theoretically double-spend. But doing so would instantly devalue Bitcoin, destroying the value of the attacker's own massive hardware investment and mining rewards. The cost of attack rises with Bitcoin's value, while the incentive to attack falls. It's a self-reinforcing security model.

SCHELLING POINTS

Bitcoin's 21 million supply cap is a Schelling point — a focal point that participants converge on without explicit coordination. Everyone expects it, so everyone enforces it. Changing it would require convincing the majority to abandon the very property that gives Bitcoin value. This social consensus is arguably stronger than any cryptographic guarantee — it's a Nash Equilibrium where no individual benefits from deviating.

Node game theory: Full nodes enforce rules for free (no block reward). Why do they run? Because node operators are typically Bitcoin holders whose wealth depends on the network's integrity. Running a node is rational self-interest disguised as altruism — you verify to protect your own assets.

THE BYZANTINE GENERALS

The Byzantine Generals Problem — how can distributed participants reach consensus when some may be dishonest? — was considered unsolvable for decades. Bitcoin's Proof of Work provides a practical solution: generals (miners) prove their commitment by expending real resources (energy), making dishonesty prohibitively expensive. Satoshi didn't just create a currency — he solved a 40-year-old computer science problem.

MODULE 08

Building on Bitcoin: Developer Introduction

You don't need to be a protocol developer to build on Bitcoin. A growing ecosystem of tools, libraries, and APIs makes it increasingly accessible to create applications that leverage Bitcoin's infrastructure.

Bitcoin Core RPC: The reference client exposes a JSON-RPC interface that lets you query blockchain data, create transactions, manage wallets, and interact with the network programmatically. This is the lowest-level interface — powerful but complex.

Language	Library	Best For
Python	python-bitcoinlib, bdk-python	Scripting, prototyping, data analysis
JavaScript	bitcoinjs-lib, bdk-wasm	Web apps, browser wallets
Rust	rust-bitcoin, BDK	Performance-critical apps, wallets
Go	btcd, btcutil	Infrastructure, node implementations

The **Lightning Development Kit (LDK)** lets you embed Lightning Network functionality into any application. Rather than running a full Lightning node, LDK provides modular components you can integrate into mobile apps, web services, or IoT devices.

WHERE TO START

Beginner: Start with Mempool.space's API — query blocks, transactions, and fee data via simple HTTP. Build a transaction explorer or fee estimator. Intermediate: Use bitcoinjs-lib or python-bitcoinlib to construct and sign transactions programmatically. Advanced: Set up Bitcoin Core in regtest mode (local test network), create custom scripts, and experiment with Taproot outputs.

THE BUILDER'S MINDSET

Bitcoin is open-source, permissionless infrastructure. Anyone can build on it without asking permission. The most impactful Bitcoin companies — exchanges, wallet providers, Lightning services, mining operations — were all started by people who learned the protocol and saw opportunities. Understanding Bitcoin at the level you've reached in this course puts you in a small percentage of the global population. The question isn't whether you can build — it's what you'll choose to build.

Mastery Achieved!

You've completed Course Module 03 — Bitcoin Mastery. You now have a deep, protocol-level understanding of Bitcoin — from the cryptography securing every transaction to the game theory keeping the network honest.

Head to the website to take the interactive quiz. Score 70% or higher to earn your official Bitcoin Academy Advanced Certificate.

Take the Quiz

Earn Your Certificate

You've completed the Bitcoin Mastery course material.

Now test your knowledge and earn your official Bitcoin Academy certificate.

Step 1 — Visit the link below

Step 2 — Scroll to the Final Quiz section

Step 3 — Score 70% or higher to pass

Step 4 — Enter your name and print your certificate

Your quiz is waiting at:

bitcoinacademy.online/course-advanced.html

Frame your certificate. Put it on the wall.
You've earned it.