Cryptographic Foundations

L02: A Standalone Mini-Course on Blockchain's Mathematical Backbone

BSc Blockchain Course

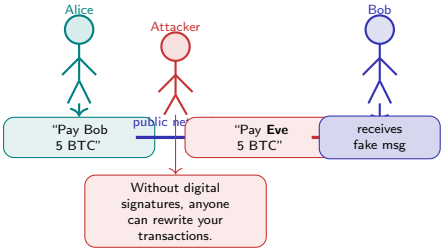
What If Someone Could Forge Your Digital Signature on the Blockchain?

Every blockchain transaction is just a message: "Pay Bob 5 BTC." The message travels across a public network where anyone can read it, copy it, or – without protection – change it. There is no bank to verify your identity, no customer service number to call.

The core tension:

- How does the network know the message really came from Alice and not from an attacker pretending to be Alice?
- How does anyone confirm the message was not altered in transit – that "5 BTC to Bob" did not become "5 BTC to Eve"?
- How can all of this work without any central authority to ask?

The answer is cryptography: mathematical operations that are easy to perform but practically impossible to reverse. Every blockchain transaction trusts mathematics instead of banks – but what if someone breaks the math?



Digital signatures solve the forgery problem: they bind a message to a specific sender using mathematics that no impostor can replicate.

Core tension: every blockchain transaction trusts mathematics instead of banks – but what if someone breaks the math?

Think About the Last Time You Proved Your Identity Online – How Did You Do It?

You have done this hundreds of times without thinking about it. You typed a password into a website, and the website checked it against a stored copy. The system worked because both you and the bank shared a secret – and because you trusted the bank to store that secret safely.

Now imagine removing the bank from the equation. No central server stores your credentials. No customer service can reset your password. No institution vouches for your identity. You must prove to a network of complete strangers that you – and only you – authorised a transaction. How?

Three Prompts – Reflect Before We Continue

1. **List every step that happens when you log into your bank account.** Start with your fingers on the keyboard. Where does the password travel? Who checks it? What could go wrong at each step? How many institutions are involved before you see your balance?
2. **What happens if the bank's password database is breached?** In 2012, LinkedIn lost 117 million hashed passwords. In 2019, Capital One exposed 100 million customer records. What is the risk when a central authority holds millions of secrets in one place?
3. **What if you could prove your identity using mathematics alone – no shared secret, no central server, no trust required?** This is the promise of public-key cryptography: you hold a private key that never leaves your device, and anyone can verify your signature using only a public key. No one needs to store your secret. No one can reset it. No one can steal it from a server. But you can never recover it if you lose it.

The shift from passwords to cryptographic keys is the shift from trusting institutions to trusting mathematics.

What Makes a Hash Different from Encryption, and Why Does Blockchain Need Both?

Four cryptographic tools – same math family, very different jobs:

	Hash Function	Symmetric Encryption	Asymmetric Encryption	Digital Signature
Purpose	Fingerprint	Confidentiality	Key exchange	Authenticity
Reversible?	No (one-way)	Yes (same key)	Yes (priv. key)	Verifiable
Output size	Fixed (256 bit)	Same as input	Same as input	Fixed (512 bit)
Keys needed	None	1 shared	2 (pub + priv)	2 (pub + priv)
Blockchain use	Block headers	Off-chain data	Rarely used	Every transaction
Example	SHA-256	AES-256	RSA, ECIES	ECDSA

The pattern to notice: Only hashing is truly one-way – everything else can be reversed or verified with the right key. Blockchain relies most heavily on hashing (for integrity) and digital signatures (for ownership). Symmetric encryption is rarely used on-chain because the ledger is public by design.

Why blockchain needs both:

- **Hashing** creates a tamper-evident seal. Change one bit in a block and the hash changes completely. Every subsequent block in the chain breaks.
- **Digital signatures** prove who sent a transaction. Only the private key holder can create a valid signature, but anyone with the public key can verify it.

Quick test – which tool?

- "Prove this block has not been modified" – **Hash**
- "Prove Alice authorised this transfer" – **Signature**
- "Send a secret message to Bob" – **Encryption**
- "Summarise 2000 transactions in 32 bytes" – **Hash** (Merkle root)

Hashing guarantees integrity (nothing was changed); digital signatures guarantee authenticity (the right person sent it). Together they make trustless transactions possible.

Hash = fingerprint (one-way, no key). Signature = proof of authorship (two keys, verifiable). Blockchain needs both.

Follow One Bitcoin Transaction from Signing to Verification

Five steps – from Alice's wallet to the blockchain:

1. Create transaction
Alice's wallet builds a message: "Send 0.5 BTC from address A to address B."

↓

2. Sign with private key
The wallet hashes the message (SHA-256), then signs the hash using Alice's private key (ECDSA).

↓

3. Broadcast to network
The signed transaction is sent to all connected nodes. Anyone can see it; no one can modify it.

↓

4. Verify with public key
Each node checks: does Alice's public key confirm the signature? Does address A have enough funds?

↓

5. Include in block
A miner bundles verified transactions, solves a proof-of-work puzzle, and adds the block to the chain.

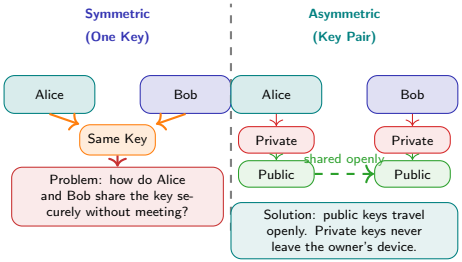
What cryptography does at each step:

1. The transaction is just data – unprotected until signed.
2. The private key is a 256-bit secret number. ECDSA produces a signature pair (r, s) that is unique to this exact message and this exact key. Change one bit and the signature is invalid.
3. The signature travels with the message. Even though the network is public, no attacker can modify the message without invalidating the signature.
4. Verification uses only Alice's public key – her private key is never transmitted. The math confirms: "Yes, whoever holds the private key behind this public key created this signature."
5. Once in a block, the transaction is hashed into a Merkle tree. Changing it would break every hash above it and invalidate the block header.

No bank, no identity check, no intermediary. Cryptography alone proves who sent the transaction and that it was not altered.

Five steps: create, sign, broadcast, verify, include. Every step is enforced by math, not by institutions.

How Does Asymmetric Cryptography Let Strangers Trust Each Other?



The key distribution problem:

- **Symmetric:** One shared key encrypts and decrypts. Fast and efficient – but both parties must exchange the key secretly first. With millions of strangers on a public network, this is impossible.
- **Asymmetric:** Each person generates a key pair – one private (secret), one public (shareable). The public key can be posted anywhere. The private key never moves.

Why blockchain uses asymmetric:

- You sign transactions with your private key.
- The network verifies with your public key.
- No pre-arranged secret exchange needed.
- Strangers can verify you without knowing you.

Think of it as a mailbox with a slot: anyone can drop a letter in (public key), but only you have the key to open the box (private key).

Asymmetric cryptography eliminates the need to share secrets. This is why millions of strangers on a blockchain can verify each other's transactions without ever meeting.

Asymmetric = two keys solve the key-distribution problem. No shared secret needed between strangers.

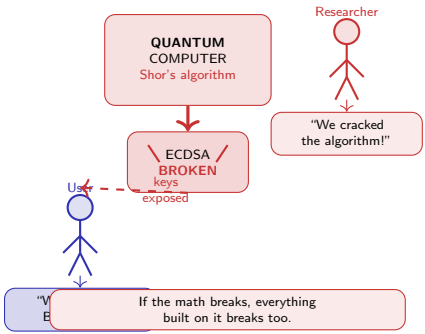
What Happens When a Cryptographic Algorithm Gets Broken?

Cryptography does not promise permanent security – it promises that breaking it would take longer than anyone can wait. But "longer than anyone can wait" has a shelf life. Algorithms that were unbreakable ten years ago have been cracked by faster hardware and smarter attacks.

Real-world breaks:

- **DES (56-bit):** US government standard from 1977. Cracked in 22 hours by a custom machine in 1999. Cost: roughly 250,000 dollars.
- **MD5 (128-bit hash):** Collision found in 2004. By 2008, researchers created a rogue certificate authority. Still used in legacy systems today.
- **SHA-1 (160-bit hash):** Google produced a practical collision in 2017 (SHAttered). Cost: roughly 110,000 dollars in cloud compute.

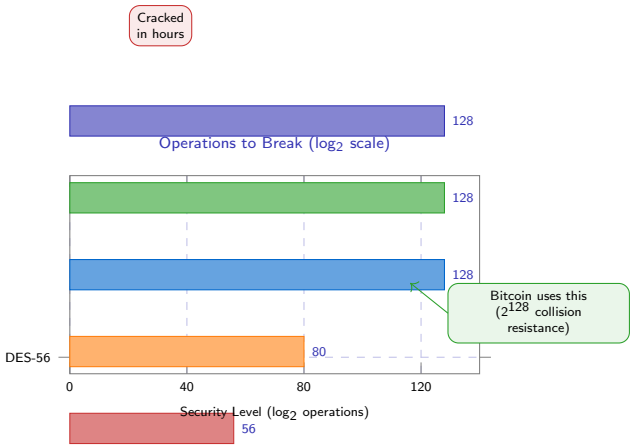
The quantum threat: A sufficiently powerful quantum computer running Shor's algorithm could break the elliptic-curve math that protects every Bitcoin private key. No such computer exists yet – but the race to build one is real.



Cryptographic security is not permanent – it is a race between the defenders who increase key sizes and the attackers who find faster ways to break them.

Every algorithm has a shelf life. Post-quantum cryptography (lattice-based, hash-based) is being standardized now for the transition ahead.

How Many Operations Would It Take to Crack Bitcoin's Cryptography?



$\log_2$ scale: each increment doubles difficulty. 128 means $2^{128} \approx 3.4 \times 10^{38}$ operations.

Putting the numbers in perspective:

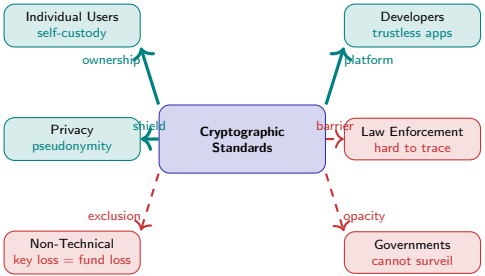
- **DES (56-bit):** Cracked in 22 hours by a dedicated machine in 1999. Hardware improvements consumed the entire margin.
- **RSA 1024-bit:** Insecure since 2010. Equivalent to roughly 80-bit symmetric security.
- **AES 128-bit:** At 10^{18} operations/second, breaking this takes roughly 10 billion years – longer than the age of the universe.
- **Bitcoin's choice:** SHA-256 and secp256k1 both provide 128-bit security. Cracking either by brute force requires more energy than the sun produces.

Caveat: assumes no math breakthrough and no quantum computer.

Bitcoin's cryptography is not unbreakable in theory – it is unbreakable in practice, given current and foreseeable technology.

128-bit security = 3.4×10^{38} operations. Even a billion supercomputers would need longer than the age of the universe.

Who Benefits from Unbreakable Math – and Who Gets Locked Out?



The same math creates winners and losers:

Individual users: Self-custody means no bank can freeze your funds, no government can seize your assets without your private key. True financial sovereignty.

Developers: Cryptographic guarantees let developers build applications where users do not need to trust the developer – only the math.

Privacy: Pseudonymous addresses allow transactions without revealing real-world identity.

Non-technical users: Lose your private key and your funds are gone permanently. There is no "forgot password" link. Roughly 20 percent of all Bitcoin is estimated to be permanently inaccessible.

Governments: Strong cryptography limits surveillance and makes tax evasion, sanctions evasion, and money laundering harder to detect.

Unbreakable math is a double-edged sword: it protects individual sovereignty but eliminates the safety nets that institutions provide.

The same cryptographic strength that enables self-custody also means there is no recovery if you lose your keys.

Three Questions That Reveal Whether a Blockchain's Cryptography Is Sound

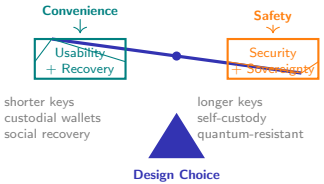
Use these three questions to evaluate any blockchain's cryptographic design:

Q1 **Can the keys be recovered if lost?** If yes – someone else can access them too. If no – lost keys mean lost funds forever. Most systems sit uncomfortably between these extremes. Custodial wallets offer recovery but reintroduce trust. Self-custody offers sovereignty but no safety net.

Q2 **Who controls the signing process?** Does the user sign locally on their device? Or does a service sign on their behalf? If a third party holds the signing key, you have reinvented the bank – with worse consumer protection.

Q3 **Is the algorithm quantum-resistant?** SHA-256 resists Grover's algorithm with a halved security margin (128-bit effective). ECDSA on secp256k1 does not resist Shor's algorithm. Post-quantum migration is not optional – it is a question of when.

Every design tips the balance.
Q1–Q3 tell you which way.



Quick diagnostic:

- All three answers favour security → maximum sovereignty, maximum risk
- All three favour usability → the system is convenient but depends on trust
- Best designs balance both – e.g. multi-signature wallets, social recovery, hybrid custody

There is no universally correct answer. The right cryptographic design depends on who the users are and what they are willing to risk.

Q1 (recovery), Q2 (signing control), Q3 (quantum readiness) – apply these to any wallet, exchange, or protocol.

Your Challenge

Read the case below, then apply the three evaluation questions from the previous slide.

Case: Evaluating a Startup's Crypto Wallet Design

Situation: You are advising a fintech startup building a consumer crypto wallet. Their current design:

- The wallet uses **128-bit keys** (half the length of Bitcoin's 256-bit keys) to "make the user experience simpler."
- Private keys are **stored on a central server** so users can recover accounts if they forget their password.
- The team plans to add **quantum-resistant algorithms "later,"** once the technology matures.

Apply Q1–Q3. Fill in the table:

Question	Assessment	Your reasoning (one sentence)
Q1: Can keys be recovered if lost?	.....	.....
Q2: Who controls the signing process?	.....	.....
Q3: Is the algorithm quantum-resistant?	.....	.....

Discuss (3 min): What would you recommend? Is there a design that keeps recovery possible without storing private keys on a central server? (Hint: multi-signature schemes, threshold cryptography, or social recovery.)

The hardest design problem in blockchain is balancing security with usability. There is no free lunch – every convenience has a trust cost.