Topic 3.1: Cryptographic Building Blocks

Hashing, Keys, and Digital Signatures

Joerg Osterrieder

Digital Finance

2025

Learning Objectives

By the end of this topic, you will be able to:

1. **Explain** what cryptographic hash functions are and their essential properties
2. **Describe** how public-key cryptography enables secure identity without a central authority
3. **Understand** how digital signatures provide authentication, integrity, and non-repudiation
4. **Connect** these three primitives to how blockchain systems establish trust
5. **Apply** these concepts in hands-on exercises using Python

Why This Matters

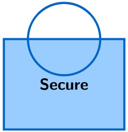
Cryptography is the foundation that allows blockchain to replace institutional trust with mathematical proof.

Definition: The science of secure communication in the presence of adversaries.

Everyday Examples:

- **HTTPS** – Secure websites (the padlock icon)
- **WhatsApp** – End-to-end encrypted messages
- **ATM PINs** – Encrypted card data
- **Passwords** – Stored as hashes, not plaintext

Key Insight: You already use cryptography daily!



Cryptography protects your data

In This Topic:
We focus on *what* cryptographic tools guarantee, not the complex math behind them.

The Trust Problem: Why Cryptography Matters for Finance

Traditional Trust Model

- Banks verify your identity
- Courts enforce contracts
- Governments back currency
- Intermediaries everywhere

Problem: Single points of failure

Cryptographic Trust Model

- Mathematics verifies identity
- Code enforces agreements
- Network backs value
- Trust is distributed

Solution: Trust through verification

Key Insight: Cryptography lets us replace "trust me" with "verify this"

Three Cryptographic Primitives

Hash Functions	**Public-Key Crypto**	**Digital Signat**
Digital fingerprints	Identity without authority	Unforgeable pro
Integrity	Identity	Non-repudiatio
Data hasn't changed	You are who you claim	You can't deny sig

Focus: What these tools *guarantee*, not how the math works

These three primitives are the atoms of decentralized trust

Think of it like: A fingerprint machine for data

- Any person $\rightarrow$ unique fingerprint (fixed size)
- Any data $\rightarrow$ unique hash (fixed size: 256 bits)

SHA-256 = Secure Hash Algorithm producing a 256-bit output. Hexadecimal = a number system using 0–9 and a–f, common in computing.

Definition: A hash function takes *any* input and produces a fixed-size output called a **hash** (or digest).

Hash Functions: Digital Fingerprints

Any input data → HASH (SHA-256) → `a7f3...c9d2` (256 bits)

Deterministic: Same input → same output, always

One-way: Cannot reverse to find input

Collision-resistant: Practically impossible to find two inputs with same hash

Avalanche effect: Tiny change → completely different output

Five Essential Properties of Hash Functions

Don't memorize all five – the key insight is that hash functions are **one-way** *and produce* **unique** *outputs.*

1. **Deterministic**
   Hash("Bitcoin") today = Hash("Bitcoin") tomorrow = Hash("Bitcoin") forever

2. **Fixed Output Size**
   SHA-256 always produces 256 bits (64 hex characters), regardless of input size

3. **One-Way (Preimage Resistant)**
   Given a hash, you cannot compute the original input

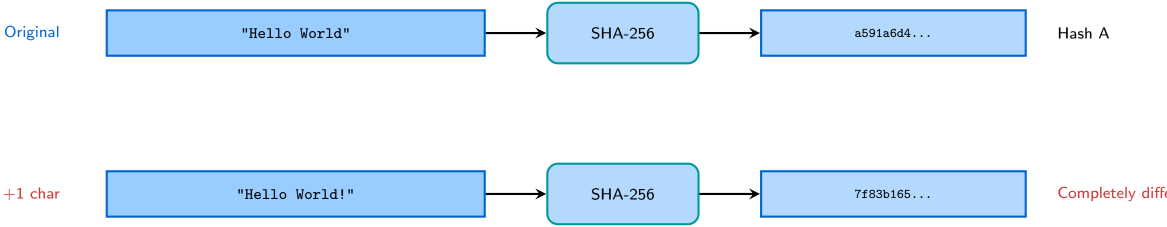
4. **Collision Resistant**
   Practically impossible to find two different inputs with the same hash

5. **Avalanche Effect**
   Change 1 bit of input → approximately 50% of output bits change

Why These Matter

Together, these properties make hashes reliable "digital fingerprints" for data integrity.

The Avalanche Effect Visualized



Why this matters for blockchain:

- Change one transaction → entire block hash changes
- This change cascades through all subsequent blocks
- Tampering becomes immediately detectable

How Secure is SHA-256?

SHA-256 produces 2^{256} possible outputs.

How big is that number?

$$2^{256} \approx 10^{77}$$

This is close to the estimated number of
atoms in the observable universe ($\approx 10^{80}$)

To find a collision by brute force:

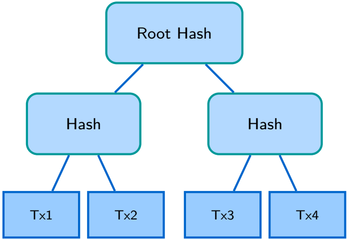
- Even trying 1 billion hashes per second
- Using every computer on Earth
- Would take longer than the age of the universe

Practical Security

SHA-256 is considered cryptographically secure for all practical purposes.

Three Key Uses in Blockchain

- **Data integrity:** Verify nothing changed – if the hash matches, the data is untouched
- **Block linking:** Each block contains the hash of the previous block, creating a tamper-evident chain
- **Efficient verification:** Hash-based structures (covered next) let you check any single transaction with only a handful of checks, even among thousands



Hash-Based Verification

Like a family tree where you can verify any member by checking their parents

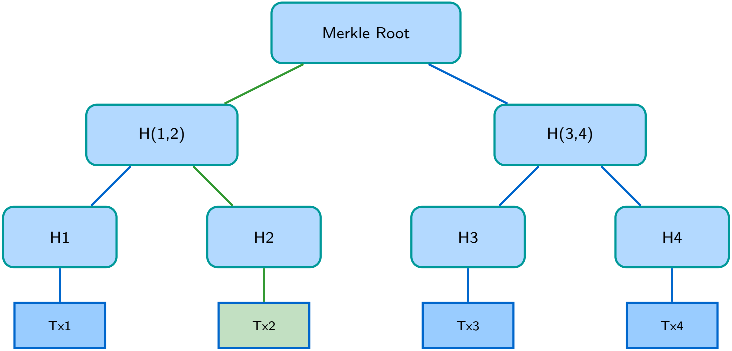
Only a handful of checks needed

The Guarantee

If two hashes match, the data is identical (with overwhelming probability).

Problem: How do you verify one transaction out of thousands without downloading everything?

Like verifying one page of a book by checking chapter summaries – you don't need to read every page



To verify Tx2: Only need H1, H(3,4), and Merkle Root (3 items, not 4 transactions)

With 1 million transactions, you only need about 20 hashes to verify any single one

Introduction to Public-Key Cryptography

The Problem: How can two strangers communicate securely without meeting first to exchange a secret password?

Traditional (Symmetric) Encryption:

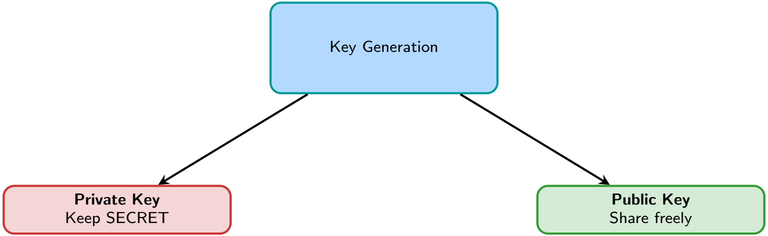
- Same key encrypts and decrypts
- Problem: How do you safely share the key?

Public-Key (Asymmetric) Solution:

- Two mathematically related keys
- One key encrypts $\rightarrow$ only the other decrypts
- Share one key publicly, keep the other secret

Breakthrough (1976): Diffie, Hellman, and later RSA showed this was mathematically possible

Public-Key Cryptography: Identity Without Authority



Mathematical relationship: Public key is *derived* from private key

Easy: Private $\rightarrow$ Public   Impossible: Public $\rightarrow$ Private

Public-Key Cryptography: How It Works

To send encrypted message to Bob:

1. Alice encrypts with Bob's public key
2. Only Bob can decrypt with his private key

Key insight: Only the person with the private key can decrypt

The Mathematical Intuition (No Math Required!)

Analogy: The Padlock and Key

Traditional Lock

- Same key locks and unlocks
- Must physically give key to someone
- If copied, security is broken

Public-Key "Lock"

- Public key = open padlock (share freely)
- Private key = the only key that opens it
- Anyone can lock (encrypt), only you can open (decrypt)

Message $+$ Public Key $\rightarrow$ Encrypted $+$ Private Key $\rightarrow$

From Public Key to Blockchain Address

What You Control

- Private key = your identity
- Whoever has it controls the funds
- Lose it = lose everything
- Share it = share everything

What You Share

- Address = your "account number"
- Safe to share publicly
- Used to receive payments
- Cannot derive private key from it

"Not your keys, not your coins" – a fundamental principle of crypto

Elliptic Curve Cryptography in Blockchain

You don't need to understand the math. The key point is: blockchain uses a specific type of digital signature that is compact and secure.

Why do blockchains use elliptic curve signatures?

There are different mathematical approaches to creating digital signatures. Blockchains chose **Elliptic Curve Digital Signature Algorithm (ECDSA)** because it provides strong security with much smaller keys:

Property	Older method (RSA)	Blockchain method (ECDSA)
Key Size (same security)	3072 bits	256 bits
Signature Size	Large	Small
Transaction Fee Impact	Higher	Lower

Bitcoin and Ethereum use a specific curve called secp256k1 – you don't need to remember this name. What matters is that it provides extremely strong security (more combinations than atoms in the universe) with small, efficient signatures.

In NB05, we use RSA for simplicity, but real blockchains use ECDSA for efficiency

What is a Digital Signature?

Definition: A mathematical scheme that proves:

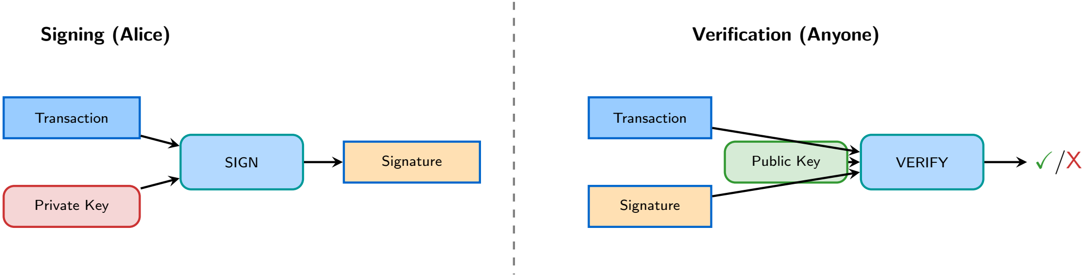
1. **Who** created or approved a message (authentication)
2. That the message **hasn't been changed** (integrity)
3. The signer **can't deny** signing (non-repudiation)

Physical vs Digital Signatures:

Property	Handwritten	Digital
Can be forged?	Yes	Practically no
Tied to document?	No	Yes (any change invalidates)
Remotely verifiable?	No	Yes
Provably unique?	No	Yes

Key Point: Digital signatures are *stronger* than handwritten ones!

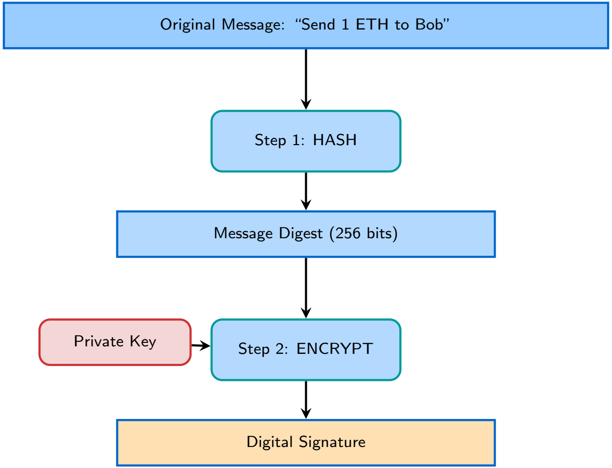
Digital Signatures: Unforgeable Proof



What Digital Signatures Guarantee:

- **Authentication:** Only private key holder could create this signature
- **Integrity:** The message hasn't been altered
- **Non-repudiation:** Signer cannot deny having signed

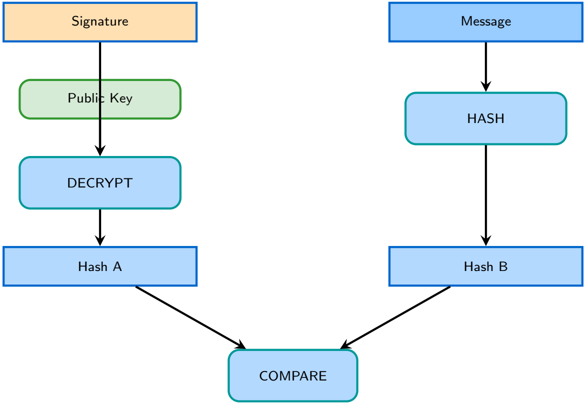
The Signing Process: Step by Step



Why hash first?

- Messages can be any size; hashes are always fixed (256 bits)
- Encrypting a small hash is much faster than encrypting a large message

The Verification Process: Step by Step



Hash A = Hash B?

✓Valid X Invalid

If hashes match: Signature is valid – message is authentic and unaltered

Digital Signatures in Blockchain Transactions

Every blockchain transaction includes a digital signature proving authorization.

A blockchain transaction contains four essential pieces:

Component	What It Does
Who sends	Sender's address (e.g., 0x7a2b...)
Who receives	Recipient's address (e.g., 0x9c4d...)
How much	The amount to transfer
Digital signature	Proves the sender authorized this transfer

How it works in practice:

1. Alice creates transaction: "Send 1.5 ETH to Bob"
2. Alice signs with her private key
3. Network verifies signature using Alice's public key
4. If valid: transaction is processed. If invalid: rejected

Technical details like gas fees and transaction ordering are covered in Topic 3.3.

Summary: The Cryptographic Trust Stack

Blockchain = All three combined at scale

$\uparrow$

Digital Signatures $\rightarrow$ Authorization & Non-repudiation

$\uparrow$

Public-Key Crypto $\rightarrow$ Identity & Ownership

$\uparrow$

Hash Functions $\rightarrow$ Data Integrity

Key Takeaway

Cryptography transforms trust from "believe the institution" to "verify the math." No bank, government, or third party needed – just mathematics.

Hands-On Exercise: NB05 Cryptographic Operations

What you'll do in the Colab notebook:

1. **Hash Functions**
   - Compute SHA-256 hashes of different inputs
   - Observe the avalanche effect firsthand
   - Verify that same input = same output
2. **Key Generation**
   - Generate a public-private key pair
   - Derive a wallet address from the public key
   - Understand the one-way relationship
3. **Digital Signatures**
   - Sign a message with your private key
   - Verify the signature with the public key
   - See what happens when verification fails

Access: NB05 – Cryptographic Operations
No installation required (runs in browser)

Hashing in Python:

```python
import hashlib
message = "Hello, Blockchain!"
hash_result = hashlib.sha256(message.encode()).hexdigest()
print(hash_result)  # 64 hex characters
```

Creating a signature:

```python
# Sign with private key
signature = private_key.sign(
    message_hash,
    algorithm=hashes.SHA256()
)
```

Verifying a signature:

```python
# Verify with public key
public_key.verify(signature, message_hash)
# Raises exception if invalid!
```

Complete code and explanations in NB05 notebook

Discussion: Where Do You See These Concepts?

Think-Pair-Share: Where else are these cryptographic primitives used?

Hash Functions

- Password storage
- File integrity (checksums)
- Git version control
- Digital forensics
- Deduplication systems

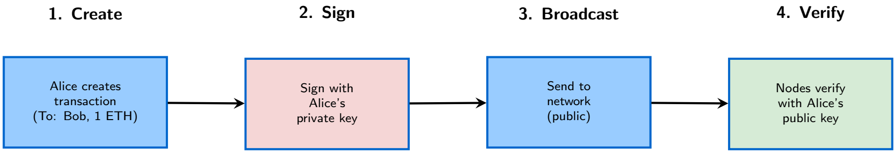
Digital Signatures

- Software updates
- Email (S/MIME, PGP)
- PDF documents
- Code signing
- SSL/TLS certificates

Discussion Questions

1. Why might a company hash passwords instead of encrypting them?
2. What happens to digital signatures if quantum computers become powerful?

Application: How a Blockchain Transaction Stays Secure



Security guarantees at each step:

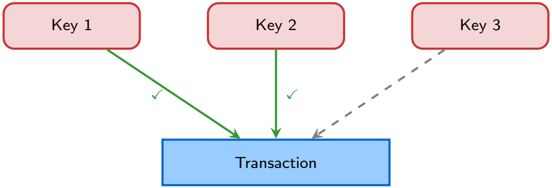
- **Step 2:** Only Alice can sign (private key)
- **Step 3:** Transaction is public but tamper-evident (hash)
- **Step 4:** Anyone can verify Alice authorized it (public key)

Result: No one can forge, alter, or deny the transaction

Application: Multi-Signature Wallets

Problem: What if one private key is stolen or lost?

Solution: Require multiple signatures (e.g., 2-of-3 multisig)



2 of 3 = Valid!

Use Cases:

- Corporate treasury (CFO + CEO approval)
- Family inheritance (multiple heirs)
- Exchange cold storage (security team)

Multi-sig combines multiple digital signatures for enhanced security

Application: Trust Implications for Finance

How cryptography changes financial trust:

Aspect	Traditional	Cryptographic
Identity verification	Bank/Government	Public key
Authorization	Signature card	Digital signature
Transaction integrity	Bank records	Hash chains
Dispute resolution	Courts	Mathematical proof
Account recovery	ID documents	Seed phrase

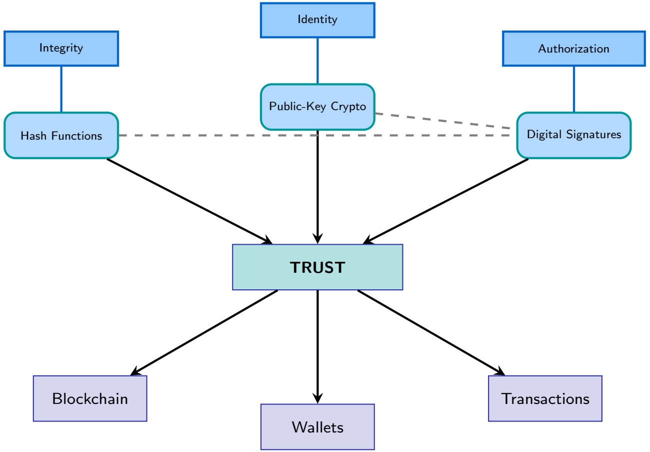
Trade-off

Benefit: No intermediaries, censorship-resistant, 24/7 operation
Risk: "With great power comes great responsibility" – lose your keys, lose your funds

Executive Summary: 5 Key Takeaways

1. **Hash functions** create unique digital fingerprints – any change to data produces a completely different hash, enabling tamper detection
2. **Public-key cryptography** allows secure identity without central authorities – you control your identity through your private key
3. **Digital signatures** provide three guarantees: authentication (who signed), integrity (not altered), and non-repudiation (can't deny signing)
4. **These primitives combine** in blockchain to enable trustless transactions – math replaces institutional trust
5. **Security responsibility shifts** from institutions to individuals – "not your keys, not your coins"

Concept Map: How It All Connects



Key relationship: Digital signatures combine hash functions (for efficiency) with public-key crypto (for authentication)

Key Terms & Definitions (Part 1)

Hash Function A mathematical function that converts any input into a fixed-size output (digest). One-way and deterministic.

SHA-256 Secure Hash Algorithm producing 256-bit output. Used in Bitcoin and many blockchains.

Avalanche Effect Property where a tiny input change causes a dramatically different output hash.

Collision Resistance Property that makes it computationally infeasible to find two different inputs with the same hash.

Merkle Tree A tree structure where each leaf is a hash of data, and each non-leaf is a hash of its children. Enables efficient verification.

Key Terms & Definitions (Part 2)

Public Key The shareable part of a key pair, used to verify signatures or encrypt messages.

Private Key The secret part of a key pair, used to create signatures or decrypt messages. Must never be shared.

Digital Signature Cryptographic proof that a message was approved by the holder of a specific private key.

ECDSA Elliptic Curve Digital Signature Algorithm. Used by Bitcoin and Ethereum for smaller, faster signatures.

Non-repudiation The property that a signer cannot deny having signed a message, as only their private key could have created the signature.

Common Misconceptions: Myth vs Reality

Myth 1:
"Hashing encrypts data"
Reality:
Hashing is one-way – you cannot "decrypt" a hash. Encryption is reversible; hashing is not.

Myth 2:
"If my public key is exposed, I'm hacked"
Reality:
Public keys are *meant* to be public! Only exposure of your private key is dangerous.

Myth 3:
"Longer passwords = stronger hashes"
Reality:
Hash output size is fixed regardless of input. A 256-bit hash is 256 bits whether input is "a" or a novel.

Myth 4:
"Quantum computers will break all crypto"
Reality:
Some algorithms are vulnerable, but post-quantum cryptography already exists. Hash functions remain largely secure.

Question 1: What is the primary purpose of a cryptographic hash function?

A. To encrypt data so it can be decrypted later
B. To create a fixed-size unique fingerprint of any input data
C. To generate random numbers for cryptographic operations
D. To compress large files into smaller ones

Self-Assessment: Test Your Understanding

Question 1: What is the primary purpose of a cryptographic hash function?

A. To encrypt data so it can be decrypted later
B. To create a fixed-size unique fingerprint of any input data
C. To generate random numbers for cryptographic operations
D. To compress large files into smaller ones

Answer: B
A cryptographic hash function takes any input and produces a fixed-size output called a hash or digest. This serves as a unique "fingerprint" of the data. Unlike encryption, hashing is one-way and cannot be reversed.

Self-Assessment: Test Your Understanding

Question 2: What three properties does a digital signature provide?

A. Encryption, compression, and speed

B. Authentication, integrity, and non-repudiation

C. Confidentiality, availability, and scalability

D. Hashing, signing, and verification

Self-Assessment: Test Your Understanding

Question 2: What three properties does a digital signature provide?

A. Encryption, compression, and speed
B. Authentication, integrity, and non-repudiation
C. Confidentiality, availability, and scalability
D. Hashing, signing, and verification

Answer: B
A digital signature provides: (1) **Authentication** – proves who signed, (2) **Integrity** – proves the message wasn't altered, and (3) **Non-repudiation** – the signer cannot deny having signed.

Question 3: Why is it important that hash functions are one-way?

Self-Assessment: Test Your Understanding

Question 2: What three properties does a digital signature provide?

A. Encryption, compression, and speed
B. Authentication, integrity, and non-repudiation
C. Confidentiality, availability, and scalability
D. Hashing, signing, and verification

Answer: B
A digital signature provides: (1) **Authentication** – proves who signed, (2) **Integrity** – proves the message wasn't altered, and (3) **Non-repudiation** – the signer cannot deny having signed.

Question 3: Why is it important that hash functions are one-way?
Answer: Because someone who sees a hash cannot reverse-engineer the original data. This protects transaction details, passwords, and other sensitive information on the blockchain.

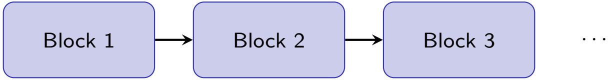
Now that you understand the building blocks, we'll see how they assemble:

Topics Covered:

- What is a blockchain?
- How blocks link together
- The block structure
- Consensus mechanisms
- The blockchain trilemma

You'll Learn:

- Why tampering is detectable
- How networks agree on truth
- Trade-offs in blockchain design
- Proof of Work vs Proof of Stake



Preview: The hash of each block is included in the next – creating an unbreakable chain

Resources for Further Learning

Hands-On:

- **NB05:** Cryptographic Operations (Colab notebook)
- Online SHA-256 calculator: `https://emn178.github.io/online-tools/sha256.html`

Reading:

- Antonopoulos, A. (2017). *Mastering Bitcoin*, Chapter 4: Keys & Addresses
- Narayanan et al. (2016). *Bitcoin and Cryptocurrency Technologies*, Chapter 1

Video:

- 3Blue1Brown: "How secure is 256-bit security?" (YouTube)
- Computerphile: "Hashing Algorithms and Security" (YouTube)

Interactive:

- Anders Brownworth's Blockchain Demo: `https://andersbrownworth.com/blockchain/`

Questions?

Topic 3.1: Cryptographic Building Blocks

Hashing, Keys, and Digital Signatures

Next: Topic 3.2 – Blockchain Mechanics

Joerg Osterrieder — Digital Finance — 2025