The Hitchhiker's Guide to Blockchain

A (non-formal) technical companion for that thing you hear every day

Stochastic Batman

Winter 2025

Abstract

This document puts together the material taught by Professor Dr. David Kipshidze in *Blockchain & Cryptography Introduction* at Kutaisi International University (Winter 2025). It formalizes parts of the lectures to clarify certain topics while remaining comprehensible to non-technical readers. This document does not aim to explain why we need digital money. Search online (do not use an LLM!) for "Financial Crisis 2008" and try to figure out why it served as "an inspiration" for Satoshi Nakamoto to create Bitcoin, the first cryptocurrency. Unless otherwise noted, diagrams were created by the author using Excalidraw. Grammar corrections were assisted by DeepSeek V3.

Contents

1 Fundamentals

Centralized Systems

Most of the internet's systems today are centralized. This means there is a single unit (or a small group of units) controlling the system, and only this unit is responsible for decision-making. In a centralized system, all data, resources, and functions are managed from a central control point, often called a central server, administrator, or governing body. These systems can be fast, as all data is processed by a single unit, but that also means there is a single point of failure. A simple malfunction can make the entire system unavailable.

Think of a bank. You open a bank account and store your money in that bank. If the bank's system is disrupted or if all clients decide to withdraw their money simultaneously (which has happened in the past), the bank system cannot respond, and you (temporarily or permanently) lose access to your money. Not exactly the best day of your life.

Decentralized Systems

In a decentralized system, decision-making is distributed across multiple units, and unless they agree upon the decision (consensus), the decision is not carried out. "They" refers to the majority, all of them, or some fixed number of them - depending on the consensus algorithm, which we will cover later. Because such a system does not have a single point of failure, decentralized systems continue to operate even if any participant in the system is damaged or unavailable. However, there might be a speed drawback. These kinds of distributed systems are scalable, meaning they can add units easily and withstand large user demands.

So, What is a Blockchain?

A blockchain is a decentralized system where data is stored through a chain of chronologically and cryptographically linked blocks.

Blockchain operates through a decentralized network where decisions are made through consensus among network participants. Once data is recorded on the blockchain, it cannot be changed - this is called immutability. Transparency means records in the blockchain can be verified from anywhere in the world at any time. No personal information needs to be disclosed to communicate with the blockchain network - this is anonymity.

Blockchain is a key cornerstone of the modern, decentralized internet, the so-called "Web 3.0".

Distributed Ledger Technology (DLT)

DLT is a decentralized system for recording and managing all sorts of transactions. Unlike traditional centralized databases, DLT allows a network of computers (nodes) to maintain a shared and synchronized ledger.

All network participants have a full copy of the ledger for complete transparency. The identity of participants is either pseudonymous or anonymous. Each transaction is recorded with a precise timestamp. All network participants agree on the validity of each record (unanimous consensus). Any validated records are irreversible and cannot be changed (immutability). All records are individually encrypted.

Additionally, DLT is programmable: "Smart Contracts" enable the execution of predefined rules and agreements without intermediaries.

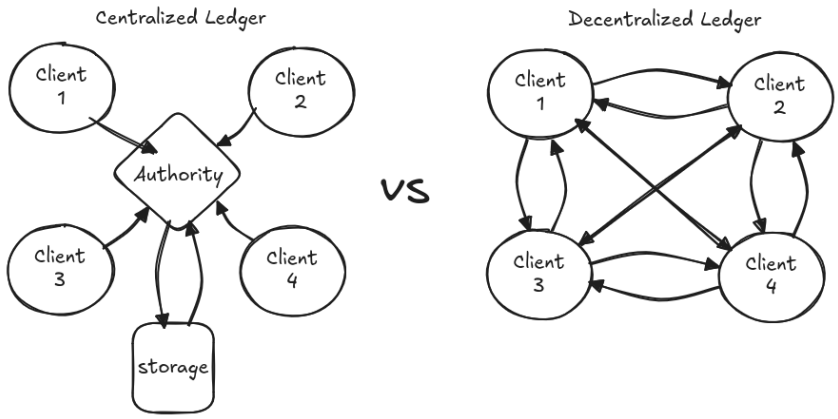

Figure 1: Centralized vs Decentralized Ledgers

2 Cryptography and Data Security

Cryptography is the practice of developing and using coded algorithms to protect and obscure transmitted information. Without cryptography, every message you send online would be readable by anyone, not just the intended recipient. Cryptography enables systems to protect digital identities and ensure communications are not intercepted - or if intercepted, the data obtained by third parties remains unreadable. The goal of cryptography is to ensure the confidentiality, integrity, authenticity, and non-repudiation of information.

All cryptographic algorithms involve two fundamental steps:

- Encryption - The process of converting plaintext into ciphertext using various algorithms and cryptographic keys. Cryptographic keys are secret values that control the encryption and decryption processes, functioning like digital passwords. Only those who possess the correct key can decrypt the ciphertext back into plaintext.
- Decryption - The reverse process of encryption, where ciphertext is converted back into plaintext using a decryption key.

There are two primary types of encryption:

- Symmetric - A single key is used for both encryption and decryption of messages. These algorithms are fast but require a secure method to share the key between communicating parties. Examples include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).
- Asymmetric - This approach generates two keys: a public key and a private key. These keys are used in combination to encrypt and decrypt messages. As the name suggests, the public key is visible to anyone, while the private key must be kept secret. This type of encryption solves the key distribution problem. Examples include RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography).

Some of the most common non-functional components of cryptography include:

- Confidentiality - Protects sensitive data by ensuring only authorized users can access it, primarily through encryption.
- Integrity - Ensures data has not been altered during transmission or storage, typically using hash function verification.

- Non-repudiation - Prevents parties from denying their actions by providing proof of origin, mainly using digital signatures.
- Authentication - Confirms the identity of users or systems, commonly using passwords or digital certificates.
- Secure Communication - Ensures data security during transmission with encryption protocols like SSL/TLS, maintaining privacy and protection from tampering.
- Access Control - Restricts access to resources or data based on permissions, often using different keys for different parties.

NOTE: From this point until the end of this section, the content becomes more technical, so you can skip it if you prefer not to see the math.

Diffie-Hellman

The Diffie-Hellman key exchange is a cryptographic protocol that allows two parties to securely establish a shared secret over an insecure channel. It is useful for the initial key exchange for symmetric encryption algorithms.

Algorithm 1 Diffie-Hellman Key Exchange Protocol

Parameters:
Large prime modulus: p
Primitive root modulo p: g {Primitive root ensures $g^k \mod p$ generates all values $1, \ldots, p-1$}

Key Generation:
Party A: Choose private integer a where $1 < a < p - 1$
Party B: Choose private integer b where $1 < b < p - 1$

Public Key Exchange:
Party A computes: $x_a = g^a \mod p$
Party B computes: $x_b = g^b \mod p$
Party A sends x_a to Party B
Party B sends x_b to Party A {Intercepted values are useless due to discrete logarithm problem [1]}

Shared Secret Computation:
Party A computes: $X = x_b^a \mod p = (g^b)^a \mod p = g^{ab} \mod p$
Party B computes: $X = x_a^b \mod p = (g^a)^b \mod p = g^{ab} \mod p$

Output: Shared private key X known only to A and B {Security proven by [2]}

RSA

RSA is probably the most famous (asymmetric) cryptographic algorithm. "Reverse-engineering" RSA is computationally infeasible because, at present, there is no efficient prime-factorization algorithm for large numbers.

Note: For the algorithm below, even when $\gcd(m, N) \neq 1$ (which is extremely rare for random m), RSA still works correctly through the Chinese Remainder Theorem, though this detail is often omitted in introductory explanations.

Algorithm 2 RSA Encryption Algorithm

Key Generation:
Choose two distinct large prime numbers: P and Q
Compute modulus: $N = P \times Q$
Compute Euler's totient function: $\Phi(N) = (P-1) \times (Q-1)$
Choose public exponent e where $1 < e < \Phi(N)$ and $\gcd(e, \Phi(N)) = 1$
Compute private exponent d such that $d \equiv e^{-1} \pmod{\Phi(N)}$
    (i.e., with extended euclidean algorithm find d where $e \times d = k \times \Phi(N) + 1$ for some integer k)

Output: Public key: (N, e), Private key: d {Factoring N into P and Q is computationally infeasible [3]}

Encryption (Sender):
Convert message M to integer m where $0 \leq m < N$
Compute ciphertext: $c = m^e \mod N$

Decryption (Recipient):
Compute original message: $m = c^d \mod N$
Convert integer m back to message M

Verification (Using Euler's Theorem): {For $\gcd(m, N) = 1$, Euler's theorem states $m^{\Phi(N)} \equiv 1 \pmod{N}$}
Since $e \times d = k \times \Phi(N) + 1$, we have:
$c^d \mod N = (m^e)^d \mod N = m^{ed} \mod N = m^{k \times \Phi(N)+1} \mod N$
$= (m^{\Phi(N)})^k \times m \mod N = 1^k \times m \mod N = m$

Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is a public-key cryptography approach based on the algebraic structure of elliptic curves over finite fields. Compared to traditional systems like RSA, ECC offers equivalent security with much smaller key sizes, making it more efficient for constrained environments.

Elliptic Curve Definition

An elliptic curve over real numbers is defined by the equation:

$$y^2 = x^3 + ax + b \tag{1}$$

where a and b are constants with $4a^3 + 27b^2 \neq 0$ (this discriminant condition ensures the curve is smooth, without cusps or self-intersections).

Real Geometric Shapes

The discriminant $\Delta = -16(4a^3 + 27b^2)$ determines the curve's topology over real numbers. Two possible scenarios:

- **Three real roots** ($\Delta > 0$): The curve has two components: a compact "droplet" shape (oval) and an unbounded arc.
- **One real root** ($\Delta < 0$): The curve is a single connected unbounded curve (topologically $\mathbb{R}$).

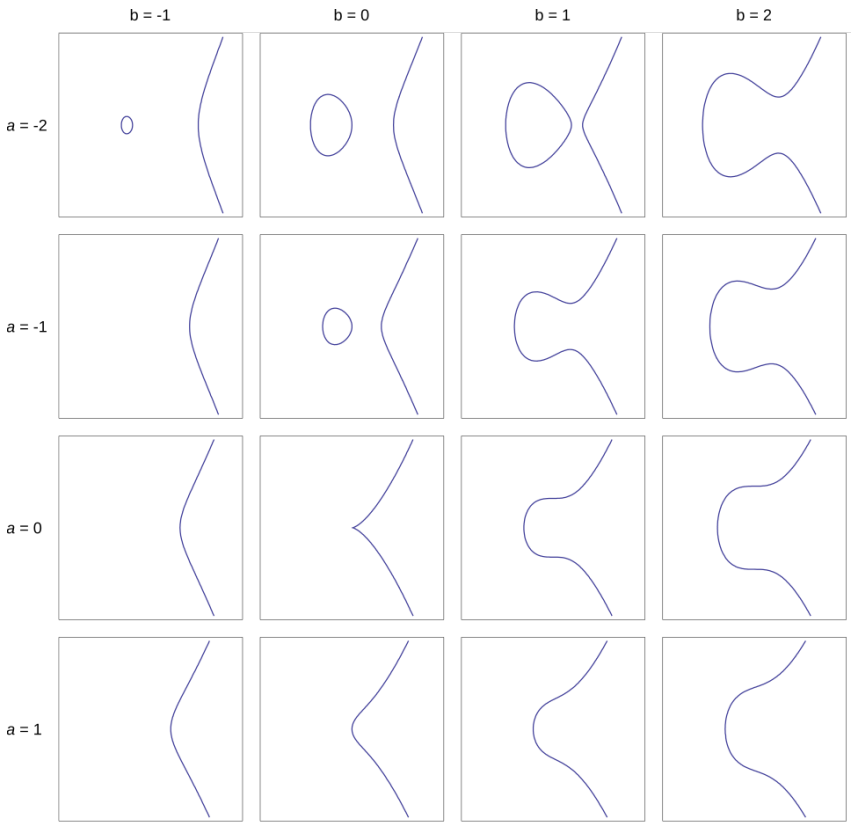

Figure 2: From Wikipedia: A catalog of elliptic curves for $x, y \in [-3, 3]$. For $(a, b) = (0, 0)$ the function is not smooth and therefore not an elliptic curve.

Common Curve Types

1. **Prime field curves**: $y^2 = x^3 + ax + b \mod p$ over $\mathbb{F}_p$ (e.g., secp256k1 used in Bitcoin)
2. **Binary field curves**: $y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_{2^m}$

Group Structure and Operations

Elliptic curves form an abelian group under point addition:

- **Identity**: The point at infinity $\mathcal{O}$
- **Point addition**: For $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, draw line through P and Q; it intersects curve at third point R'; reflect over x-axis to get $P + Q$.
- **Point doubling**: For $P = Q$, use tangent line.

The formulas for $P \neq Q$:

$$s = \frac{y_2 - y_1}{x_2 - x_1}, \quad x_3 = s^2 - x_1 - x_2, \quad y_3 = s(x_1 - x_3) - y_1 \tag{2}$$

For $P = Q$:

$$s = \frac{3x_1^2 + a}{2y_1}, \quad x_3 = s^2 - 2x_1, \quad y_3 = s(x_1 - x_3) - y_1 \tag{3}$$

Scalar Multiplication and Security

The fundamental operation in ECC is *scalar multiplication*:

$$kP = \underbrace{P + P + \cdots + P}_{k \text{ times}} \tag{4}$$

where k is an integer and P is a point on the curve.

Double-and-Add Algorithm: This efficient algorithm computes kP by exploiting the binary representation of k. Given $k = \sum_{i=0}^{n-1} k_i 2^i$ where $k_i \in \{0, 1\}$:

1: $Q \leftarrow \mathcal{O}$ {Initialize with point at infinity}
2: $R \leftarrow P$ {Temporary point}
3: **for** $i = 0$ to $n - 1$ **do**
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + R$ {Add operation}
6: **end if**
7: $R \leftarrow R + R$ {Double operation}
8: **end for**
9: **return** Q

The algorithm requires approximately $\log_2(k)$ point doublings and $h(k)$ point additions, where $h(k)$ is the Hamming weight of k's binary representation [4]. This achieves time complexity $O(\log k)$, making it practical for cryptographic applications with large k.

Security Foundation: ECC's security relies on the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**:

> *Given points P and $Q = kP$ on an elliptic curve, find the integer k.*

This problem is believed to be significantly harder than the classic discrete logarithm problem over finite fields [5]. The best known algorithms for solving ECDLP have exponential time complexity $O(2^{n/2})$ for n-bit keys [6], compared to sub-exponential algorithms for integer factorization (RSA) and classic discrete logarithms. This exponential hardness enables **smaller key sizes** while maintaining strong security:

ECC Key Size	RSA Equivalent
160 bits	1024 bits
256 bits	3072 bits
384 bits	7680 bits

These key size equivalences are based on current best-known attack complexities [7].

Fun Fact: Millennium Prize Problem

The **Birch and Swinnerton-Dyer Conjecture**, one of the seven Millennium Prize Problems, concerns elliptic curves. It relates the rank of an elliptic curve (the number of independent rational points of infinite order) to the behavior of its associated L-function at $s = 1$. While primarily a deep number theory problem, its connection to elliptic curves underscores their mathematical richness and importance beyond cryptography.

Advantages Over Traditional Cryptography

- **Smaller keys**: 256-bit ECC $\approx$ 3072-bit RSA security
- **Faster computations**: Especially for key generation and verification
- **Lower power consumption**: Ideal for IoT and mobile devices
- **Stronger security**: Apparent exponential hardness of ECDLP

3 Hashing and Blockchain Security

What is hashing?

Sometimes people confuse hash functions with encryption because both are often used in similar contexts.

Unlike encryption algorithms, where some computed values are kept secret, with hash functions we keep the input values secret.

A hash function is a (typically non-invertible) one-way mathematical algorithm that takes digital data as input and returns a fixed-size string of characters (often expressed in hexadecimal) called a hash.

Formally, for inputs $x \in D_1$, where D_1 is some domain (numbers, images, etc.), a hash function $h : D_1 \mapsto D_2^n$, where D_2 is the output alphabet and n is the output length, so $h(x) = y \in D_2^n$.

Example: $D_1 = \{APPLE, NVIDIA\}$, $D_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ is the hexadecimal system and $n = 6$, one possible hash function is $h(APPLE) = 1005EF$ and $h(NVIDIA) = C00135$.

A one-way function means you can compute $h(x) = y$, but there is no known efficient way to recover x from y except by brute force.

The existence of true one-way functions remains an open question in theoretical computer science, but modern cryptographic hash functions are strong enough that no practical method better than brute force is known for finding a preimage $x \in D_1$ for a given $y \in D_2^n$.

Hash functions are deterministic algorithms that always produce the same output for a given input, are designed to make collisions extremely unlikely (different inputs should not yield the same hash: $x_1, x_2 \in D_1, x_1 \neq x_2 \implies h(x_1) \neq h(x_2)$), and exhibit the avalanche effect, where even a tiny change to the input causes a large, unpredictable change in the output.

Merkle Trees

A Merkle tree, or hash tree, is a simple data structure for checking that data hasn't been altered. It's a binary tree where each leaf holds the hash of one data block, and each internal node holds the hash of the two child hashes joined together. Blockchains use Merkle trees to efficiently verify transactions within blocks.

Formal Definition

Given a cryptographic hash function $h : D_1 \to D_2^n$ and data blocks $x_1, x_2, \ldots, x_m \in D_1$, we construct a Merkle tree as follows:

1. **Leaf nodes** (hashes of the original data blocks): For each data block x_i, compute its hash:

$$L_i = h(x_i) \quad \forall i \in \{1, \ldots, m\}$$

2. **Internal nodes**: For each pair of child nodes $C_{\text{left}}, C_{\text{right}}$, compute:

$$N_{\text{parent}} = h(C_{\text{left}} \circ C_{\text{right}})$$

   where $\circ$ denotes concatenation. For a complete binary tree with $m = 2^k$ leaves, there are $2^k - 1$ total nodes.

3. **Merkle root**: The topmost node R represents the whole dataset; changing any single data block changes the Merkle root, so it reveals tampering with the original data. which satisfies:

$$R = h(h(h(x_1) \circ h(x_2)) \circ h(h(x_3) \circ h(x_4)) \circ \cdots)$$

Tree Structure

For m data blocks arranged in a complete binary tree:

- **Height**: $H = \lceil \log_2 m \rceil$
- **Level** l $(0 \leq l \leq H)$: Contains 2^{H-l} nodes
- **Total nodes**: $N_{\text{total}} = 2^{H+1} - 1$
- **Merkle path**: For any leaf x_i, the path to root has H hashes

Properties and Applications

Merkle trees provide several important properties:

- **Efficient verification**: To verify a data block x_i, only $O(\log m)$ hashes (the Merkle path) need to be recomputed and compared to the stored Merkle root.
- **Data integrity**: Any change in x_i propagates up the tree, changing the Merkle root:

$$x_i' \neq x_i \implies h(x_i') \neq h(x_i) \implies R' \neq R$$

- **Space efficiency**: Storing only the root hash (n bits) allows verification of any data block's integrity.
- **Incremental updates**: Changing one data block requires only $O(\log m)$ hash recalculations.

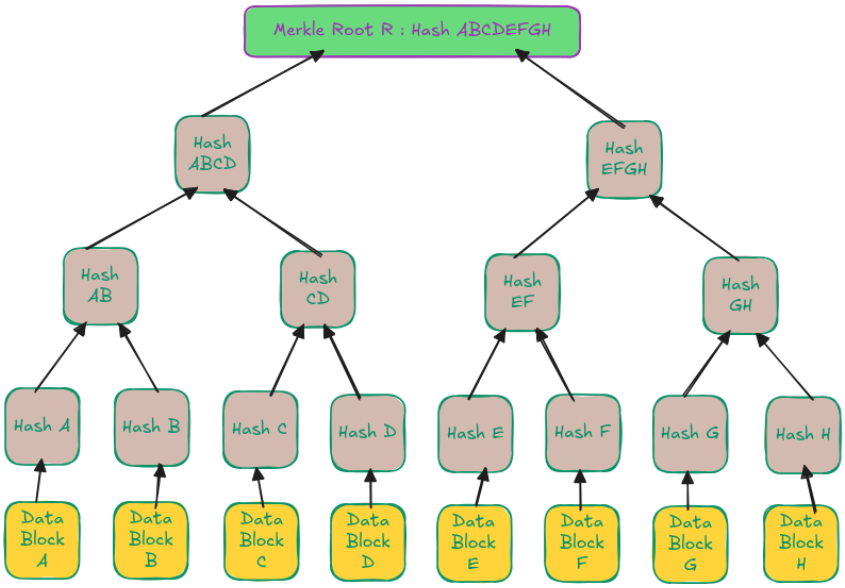

Figure 3: Merkle tree with eight leaves

Digital Signing & Verification

Digital signatures utilize asymmetric cryptography to provide authentication, integrity, and non-repudiation for digital messages. Unlike encryption which protects data confidentiality, signatures verify the origin and integrity of data.

Formal Definition

Let:

- M be the message to be signed, $M \in \mathcal{M}$ (message space)
- $h : \mathcal{M} \to \{0,1\}^n$ be a cryptographic hash function (output size n bits)
- (PK, SK) be a public-private key pair for the signer
- $\text{Sign}_{SK} : \{0,1\}^n \to \mathcal{S}$ be the signing function, where $\mathcal{S}$ is the signature space
- $\text{Verify}_{PK} : \mathcal{M} \times \mathcal{S} \to \{\text{true}, \text{false}\}$ be the verification function

The signature scheme must satisfy:

$$\text{Verify}_{PK}(M, \text{Sign}_{SK}(M)) = \text{true} \quad \forall M \in \mathcal{M}$$

The signature space $\mathcal{S}$ depends on the cryptographic scheme.
Example: for RSA signatures $\mathcal{S} = \mathbb{Z}_N$ where N is the RSA modulus.

Signing Process

1. **Hash computation**: Compute the message digest:
$$H = h(M)$$
2. **Signature generation**: Apply the signing function with the private key:
$$\sigma = \text{Sign}_{SK}(H)$$
For RSA-based signatures: $\sigma = H^d \mod N$, where d is the private key.
3. **Message transmission**: Send the pair (M, σ) to the recipient.

Verification Process

1. **Hash recomputation**: Compute the hash of the received message:
$$H' = h(M)$$
2. **Signature recovery**: Apply the verification function with the sender's public key:
$$H'' = \text{Verify}_{PK}(\sigma)$$
For RSA: $H'' = \sigma^e \mod N$, where (N, e) is the public key.
3. **Comparison**: Verify that:
$$H' \stackrel{?}{=} H''$$
If equal, the signature is valid; otherwise, it is rejected.

Security Properties

Digital signatures provide three fundamental security guarantees:

- **Authentication**: Confirms the identity of the signer through mathematical proof of private key possession. Formally:
$$\mathbb{P}(\text{Verify}_{PK}(M, \sigma') = \text{true} \mid \sigma' \neq \text{Sign}_{SK}(M)) \leq \epsilon$$
for negligible ϵ.
- **Integrity**: Ensures the message has not been altered. Any modification $M' \neq M$ results in:
$$h(M') \neq h(M) \implies \text{Verify}_{PK}(M', \sigma) = \text{false}$$
with overwhelming probability due to collision resistance of h.
- **Non-repudiation**: The signer cannot deny having signed the message, as only they possess SK. Provides legal evidence of:
$$\{\text{Verify}_{PK}(M, \sigma) = \text{true}\} \implies \{\text{Sign}_{SK}(M) = \sigma\}$$

Applications in Blockchain

In blockchain systems, digital signatures authenticate:

- **Transactions**: Each transaction is signed by the sender's private key
- **Blocks**: Block creators sign block headers (in some consensus mechanisms)
- **Smart contracts**: Executions can be authorized via signatures
- **Node communication**: Peers authenticate messages using signatures

Additional: Certificate Hierarchy (PKI)

Public Key Infrastructure (PKI) solves the fundamental trust problem: *How can Alice verify Bob's public key really belongs to Bob?* Without PKI, each user would need to personally exchange keys with every other user ($O(n^2)$ problem). PKI introduces trusted Certificate Authorities (CAs) that certify identities using digital signatures.

Certificate Structure A digital certificate binds a public key to an identity:

$$C = (\mathrm{ID}_E, PK_E, \text{metadata}, \sigma_{\text{issuer}})$$

where ID_E is the Distinguished Name (DN) or unique identifier of entity E, PK_E is the public key of E and $\sigma_{\text{issuer}} = \mathrm{Sign}_{SK_{\text{issuer}}}(h(\mathrm{ID}_E \circ PK_E \circ \text{metadata}))$.

Three-Level Trust Hierarchy

1. **Root CA**: Ultimate trust anchor. Self-signed:

   $$\sigma_{\text{root}} = \mathrm{Sign}_{SK_{\text{root}}}(h(\text{Root Certificate}))$$

   Public key PK_{root} pre-installed in browsers/OS. Typically offline.

2. **Intermediate CA**: Delegated by Root CA:

   $$\sigma_{\text{intermediate}} = \mathrm{Sign}_{SK_{\text{root}}}(h(\text{Intermediate Certificate}))$$

   Issues end-entity certificates; can be revoked if compromised.

3. **End Entity Certificate**: For users/servers:

   $$\sigma_{\text{end}} = \mathrm{Sign}_{SK_{\text{intermediate}}}(h(\text{End Entity Certificate}))$$

   Used for code signing & authentication.

Certificate Chain Verification Verify end entity certificate C_{end}:

1. Verify σ_{end} using $PK_{\text{intermediate}}$ from C_{end}

   $$\mathrm{Verify}_{PK_{\text{intermediate}}}(h(C_{\text{end}}), \sigma_{\text{end}}) \stackrel{?}{=} \text{true}$$

2. Verify intermediate certificate using pre-trusted PK_{root}

   $$\mathrm{Verify}_{PK_{\text{root}}}(h(C_{\text{intermediate}}), \sigma_{\text{intermediate}}) \stackrel{?}{=} \text{true}$$

PKI vs. Blockchain Trust

- **PKI**: Centralized hierarchy (trusted CAs)
- **Blockchain**: Decentralized trust (consensus + cryptography)

PKI enables scalable trust establishment but relies on central authorities, contrasting with blockchain's trustless model.

4 Blocks and Transactions

Non-Formal Definitions

Blocks

First, we define a **node**: a computer system that joins the network by keeping a copy of the shared ledger and often helping to verify and spread transactions and blocks. Nodes are vital for making the blockchain decentralized, secure, and able to reach agreement.

Now for the definition you were waiting for: A **block** is an object that must contain certain required information: data (about transactions), a timestamp, the hash of the current block (which is calculated), and the hash of the previous block. It may also include various other attributes depending on the specific blockchain implementation. The only block without a previous block's hash is the first one, called the **Genesis** block. This link between previous and current blocks creates the bond that forms the **Blockchain** system.

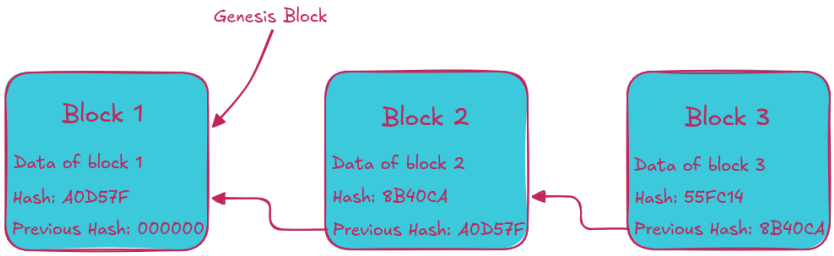

Figure 4: Toy Example: Blockchain with 3 blocks

In the blockchain network, blocks are added in order to keep a consistent and tamper-proof record. However, not every block is accepted into the blockchain.

A **valid block** follows all the rules and protocols of the blockchain network. It is checked and accepted by the network's nodes. A block is valid if it contains only valid transactions and has correct hashes for itself and the previous block.

An **invalid block** fails to meet the blockchain's requirements. These blocks are rejected by network nodes during checking. A block is invalid if it contains invalid or repeated transactions, or incorrect hashes for previous or current blocks.

Only valid blocks are added to the blockchain.

Transactions and Networks

A **transaction** is the basic unit of operation in a blockchain system. It represents the transfer of data or value between people on the network.

The process works like this: A user sends a transaction request to the blockchain network. Network participants check this transaction. After confirming it is legitimate, the data is added to a block. The new block is added to the chain and shared with everyone. The transaction is complete.

Public Networks Public blockchains are open systems where anyone can participate. They require a cryptocurrency to function and have high decentralization. Their design leads to low throughput (fewer transactions per second) and high energy consumption. They use consensus mechanisms like Proof-of-Work (PoW) and Proof-of-Stake (PoS).

Private Networks Private blockchains have participants that are pre-selected. No cryptocurrency is required for their operation. They feature low decentralization but offer high throughput and low energy consumption. They use consensus mechanisms like Practical Byzantine Fault

Tolerance (PBFT) and RAFT (yes, this is the same RAFT algorithm from non-blockchain distributed systems, which uses operation logs), plus custom mechanisms like Proof-of-Authority (PoA) used by VeChain.

Key Differences

Aspect	**Public Network**	**Private Network**
Primary purpose	Cryptocurrency-oriented	Business-oriented
Transparency level	Full transparency	Limited network transparency
Transaction cost	Typically high	Typically low
Participant identity	Anonymous or pseudonymous	Known participants
Consensus participation	Each node can participate	Only selected nodes participate
Joining requirements	Anyone can join without permission	Requires permissions to join
Decentralization	High	Low
Throughput	Low	High
Energy consumption	High	Low
Consensus mechanisms	PoW, PoS	PBFT, RAFT, PoA

A Bit of Formalism

Formal Block Definition

Let B_i denote the i-th block in a blockchain, where $i \geq 0$. Each block B_i is a tuple:

$$B_i = (H_{i-1}, T_i, \tau_i, \nu_i, \text{metadata}_i)$$

where:

- H_{i-1} is the cryptographic hash of the previous block B_{i-1}. For the Genesis block B_0, $H_{-1} = \emptyset$.
- $T_i = \{t_1, t_2, \ldots, t_k\}$ is the set of transactions contained in block i.
- τ_i is the timestamp when the block was created.
- ν_i is called "nonce", a value used in the consensus mechanism (e.g., for Proof-of-Work).
- metadata_i includes protocol-specific attributes (version, difficulty target, etc.).

The block's hash H_i is computed as:

$$H_i = h(B_i)$$

where h is a cryptographic hash function (e.g., SHA-256).

Formal Transaction Definition

A transaction $t \in T_i$ is a data structure representing a state transition:

$$t = (\text{sender}_{\text{addr}}, \text{receiver}_{\text{addr}}, \text{value}, \text{signature}, \text{data})$$

where:

- $\text{sender}_{\text{addr}}, \text{receiver}_{\text{addr}}$ are blockchain addresses.
- value $\in \mathbb{R}^+$ is the amount transferred.
- signature $= \text{Sign}_{SK_{\text{sender}}}(h(\text{transaction data}))$ authorizes the transfer.
- data contains additional information (e.g., smart contract calls, will discuss smart contracts later).

Formal Blockchain Definition

A blockchain $\mathcal{C}$ is an ordered sequence of blocks:

$$\mathcal{C} = (B_0, B_1, \ldots, B_n)$$

satisfying the chain condition:

$$H_i = h(B_i) \quad \text{and} \quad B_i.H_{i-1} = H_{i-1} \quad \forall i > 0$$

where $B_i.H_{i-1}$ is the previous block hash stored in B_i.

Validity Conditions

A block B_i is **valid** ($\text{Valid}(B_i) = \text{true}$) if:

1. **Type validity**: Each component of B_i has the correct type according to the blockchain protocol specification.
2. **Hash linkage**: $B_i.H_{i-1} = H_{i-1}$
3. **Transaction validity**: $\forall t \in T_i : \text{ValidTransaction}(t) = \text{true}$
4. **Consensus compliance**: $\text{ConsensusCheck}(B_i) = \text{true}$

A blockchain $\mathcal{C}$ is valid if:

$$\forall B_i \in \mathcal{C} : \text{Valid}(B_i) = \text{true}$$

5 Crypto Wallets

Introduction

A **crypto wallet** is a secure interface for managing digital assets on a blockchain. Rather than storing physical coins, it manages the cryptographic keys that prove ownership of assets and authorizes transactions. Beyond basic asset management, modern wallets often include built-in web browsers for interacting with decentralized applications (dApps), enabling seamless access to the decentralized web without leaving the wallet environment.

Wallet Categories: Hot vs. Cold

Crypto wallets are primarily classified by their connection to the internet and how they store private keys.

Hot Wallets

A **hot wallet** is connected to the internet and designed for frequent, convenient access. It is typically used for day-to-day transactions and interacting with dApps. However, being online makes it more vulnerable to attacks. Common types include:

- **Mobile wallets**: Apps on smartphones (e.g., Trust Wallet, Zengo).
- **Desktop wallets**: Software installed on computers (e.g., Electrum, Exodus).
- **Web wallets**: Browser extensions or web-based interfaces (e.g., MetaMask, Coinbase Wallet).

Cold Wallets

A **cold wallet** remains offline, storing private keys in an isolated environment. It is used for long-term storage (or "cold storage") of substantial assets, providing enhanced security against online threats. Types include:

- **Hardware wallets**: Physical devices (e.g., Ledger, Trezor) that sign transactions offline.
- **Paper wallets**: Physical printouts of private and public keys (now less common due to usability risks).

Custody: Who Controls the Keys?

Custody refers to who controls the private keys - and therefore the assets. This distinction defines two wallet models:

- **Self-custody (non-custodial)**: You alone hold and manage your private keys. You bear full responsibility for security, but no third party can freeze or seize your assets.
- **Third-party custody (custodial)**: A service (e.g., an exchange or custodian) holds your private keys on your behalf. This simplifies key management but introduces third-party risk - you must trust the custodian's security and integrity.

Security Features

To protect assets, wallets implement multiple security layers:

- **Recovery phrase (seed phrase)**: A 12- or 24-word mnemonic generated from the BIP-39 standard that can restore all keys and accounts. This must be kept offline and secret.
- **PIN/password**: Used to unlock the wallet locally.
- **Two-factor authentication (2FA)**: An extra layer for logging into wallet services (common in custodial wallets).

Accounts within Wallets

A single wallet can contain multiple **accounts**, each with its own key pair. This allows users to segregate assets (e.g., separate accounts for savings, trading, or dApp interactions) without managing multiple wallets. Accounts are derived deterministically from the same recovery phrase.

A Smaller Bit of Formalism

While wallets are primarily user-facing tools, we can outline their components formally. Let a wallet $\mathcal{W}$ be a tuple:

$$\mathcal{W} = (\mathcal{K}, \mathcal{A}, \mathcal{F}, \text{type})$$

where:

- $\mathcal{K} = \{sk_1, sk_2, \dots\}$ is the set of private keys (or a single master key from which others are derived).
- $\mathcal{A} = \{\text{acc}_1, \text{acc}_2, \dots\}$ is the set of accounts, each a pair $(pk, addr)$ of public key and address.
- $\mathcal{F}$ is the set of security parameters (recovery phrase, PIN, 2FA settings).
- type $\in \{\text{hot}, \text{cold}\} \times \{\text{self-custody}, \text{third-party}\}$ denotes the wallet's category and custody model.

The wallet's core function is to sign transactions: for a transaction t and account acc_j, it computes

$$\sigma = \text{Sign}_{sk_j}(h(t))$$

where $sk_j \in \mathcal{K}$ is the private key corresponding to acc_j. The wallet must also manage key derivation, address generation, and secure storage of $\mathcal{K}$.

6 Consensus Mechanisms

Non-Formalism

A **consensus mechanism** is the method by which decentralized blockchain networks agree on the validity of transactions and the current state of the ledger, without relying on a central authority. Since most blockchains are decentralized, consensus is essential for:

- Ensuring all nodes agree on a single version of the truth.
- Keeping the blockchain secure and reliable.
- Ensuring that only valid blocks are added to the chain.
- Securing the network against fraud or manipulation.

Among the many consensus protocols in blockchain, two are the most prominent:

- **Proof of Work (PoW)**: Used by Bitcoin and Litecoin. In PoW, miners compete to solve complex mathematical puzzles. The first to solve the puzzle adds a new block and receives a reward. This process requires significant computational power and electricity.
- **Proof of Stake (PoS)**: Used by Ethereum (since the Merge) and Cardano. In PoS, validators are chosen to create new blocks based on the amount of cryptocurrency they "stake" as collateral. The more coins staked, the higher the chance to be selected to validate and earn rewards. PoS is much more energy-efficient than PoW.

Quite a Bit of Formalism

Consensus in a decentralized network can be modeled as a distributed agreement problem. Let the network consist of k nodes $\{n_1, n_2, \ldots, n_k\}$, each maintaining a local copy of the blockchain $\mathcal{C}$. The goal is to ensure that for any two honest nodes n_i and n_j, their chains $\mathcal{C}_i$ and $\mathcal{C}_j$ are consistent, i.e., they agree on a common prefix up to the last confirmed block.

Formally, let $\mathcal{B}$ be the set of all possible blocks. A consensus protocol Π defines a function:

$$\Pi : \mathcal{B}^* \times \mathcal{I} \to \mathcal{B}$$

where $\mathbb{B}^*$ is the current chain (sequence of blocks) and $\mathcal{I}$ is the set of inputs (transactions, network messages, etc.). The output is a new block B that extends the chain, subject to the protocol's rules.

A consensus protocol must satisfy:

- **Agreement**: All honest nodes eventually agree on the same block for a given position in the chain.
- **Termination**: Every honest node eventually decides on a block.
- **Validity**: Only valid blocks (containing valid transactions) are accepted.
- **Fault Tolerance**: The protocol can withstand a certain fraction of Byzantine (malicious) nodes.

Proof of Work (PoW)

In PoW, miners compete to find a nonce ν such that the hash of the block header meets a certain difficulty target D. Let h be a cryptographic hash function (e.g., SHA-256). The block header includes the previous block hash H_{i-1}, a Merkle root of transactions R, a timestamp τ, and the nonce ν. The miner must find ν such that:

$$h(H_{i-1}, R, \tau, \nu) < D$$

where D is a 256-bit number adjusted periodically to maintain a constant block time.

Algorithm 3 Proof of Work (Mining Process)

Input: Transaction set T, previous block hash H_{i-1}
Output: Valid block B_i with nonce ν

Construct block header: H_{i-1}, Merkle root $R = \text{MerkleRoot}(T)$, timestamp τ
Set $\nu \leftarrow 0$
loop
  Compute $candidate\_hash \leftarrow h(H_{i-1}, R, \tau, \nu)$
  if $candidate\_hash < D$ **then**
    $B_i \leftarrow (H_{i-1}, T, \tau, \nu, candidate\_hash)$
    return B_i
  else
    $\nu \leftarrow \nu + 1$
  end if
end loop

The difficulty D is adjusted every k blocks (e.g., every 2016 blocks in Bitcoin) to maintain an average block time. If the actual block time is faster than the target, D is decreased (making it harder); if slower, D is increased (making it easier). Intuitively, the network adjusts to work for almost any number of users.

Proof of Stake (PoS)

In PoS, validators are chosen based on their stake, i.e., the amount of cryptocurrency they have locked up as collateral. Let S_j be the stake of validator V_j, and let $total\_stake = \sum_{j=1}^{m} S_j$ be the total stake in the system. The probability of V_j being selected to propose the next block is proportional to $S_j/total\_stake$.

To prevent nothing-at-stake attacks, PoS protocols often use a pseudo-random function that considers the current state and the validator's stake. Some protocols (like Ethereum's) use a combination of stake and randomization to select a committee of validators for each slot.

Algorithm 4 Proof of Stake (Validator Selection and Block Proposal)

Input: Current state St, validator set $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$ with stakes S_j
Output: Selected validator V^* for the next slot

Compute total stake: $S_{\text{total}} \leftarrow \sum_{j=1}^{m} S_j$
Generate a random seed ξ from St (e.g., from previous block hashes)
For each validator V_j with public key pk_j, compute priority score:

$$score_j \leftarrow h(\xi, pk_j) \cdot \frac{S_j}{S_{\text{total}}}$$

Select validator V^* with the highest $score_j$
return V^*

Once selected, the validator creates a block and broadcasts it. Other validators then attest to the block's validity. If the proposer acts maliciously, their stake can be slashed (partially burned).

Proof of Authority (PoA)

Proof of Authority is a consensus mechanism used in private blockchains, where validators are known and trusted entities. Validators take turns producing blocks in a round-robin (fair cyclic order) fashion, or based on a fixed schedule. VeChain uses a modified PoA called *Proof of Authority 2.0* [8], which involves a committee of 101 known validators (called Authority Masternodes) that are elected by stakeholders.

In PoA, the identity of validators is publicly known and verified. There is no mining or staking; instead, validators are authorized by the network governance. This makes PoA highly efficient and suitable for enterprise use.

In VeChain's PoA, we have "active" status: A network-derived state indicating an activity masternode's validity to produce a block. An activity masternode (AM) marked "inactive" for missing its production slot cannot produce the next block. VeChain also has "Legitimacy Condition": AM a is legitimate if and only if it is selected by its own index within its candidate set.

In practice, PoA systems may include additional mechanisms for validator accountability, such as reputation systems and penalties for misbehavior. Since validators are known, legal repercussions can also deter malicious actions.

Algorithm 5 VeChainThor (VeChain's Blockchain) PoA from [8]

Global Parameters:
    Block interval: $\Delta = 10$ seconds
    Genesis block timestamp: t_0
    Set of Authority Masternodes (AMs): $\mathcal{M}$

Input: Current block height h, timestamp t
Output: Legitimate AM a for producing block $B(h, t)$

Precondition: Timestamp must be on schedule: $(t - t_0) \mod \Delta = 0$

Step 1: Compute Pseudo-Random Number
Let $\circ$ denote byte-array concatenation
$\gamma(h, t) \leftarrow \text{hash}(h \circ t)$ {Deterministic Pseudo-Random Process (DPRP)}

Step 2: Determine Active AM Set
Let A_B be the sorted set of AMs with "active" status in parent block's state
Let $PA(\cdot)$ be the function that returns the parent block

Step 3: Candidate Validation for AM a
Construct candidate set: $A^a_{B(h,t)} \leftarrow \text{sort}(A_{PA(B(h,t))}) \cup \{a\}$
Compute selection index: $i^a(h, t) \leftarrow \gamma(h, t) \mod \|A^a_{B(h,t)}\|$

Step 4: Legitimacy Check
if $A^a_{B(h,t)}[i^a(h, t)] = a$ **then**
    a is the legitimate block producer for $B(h, t)$
    return a
else
    a is not the legitimate producer
    return $\emptyset$
end if

Status Update & Trunk Selection

The algorithm is complemented by two other critical mechanisms:

- **AM Status Update**: After a block $B(h, t_1)$ is produced, the system checks all scheduled timeslots t between the parent block's timestamp and its own. For each t, it computes the AM a_t that was responsible. It marks any a_t that failed to produce its block as "*inactive*" and sets the producer of $B(h, t_1)$ as "*active*".
- **Trunk Selection Rule (Accumulated Witness Number)**: VeChain PoA does not use the "longest chain" rule (the longest chain is not necessarily the correct one). Instead, it chooses the chain with π, the largest *Accumulated Witness Number* (AWN). For block $B(h, t)$:

  $$\pi_{B(h,t)} = \pi_{PA(B(h,t))} + \|A_{B(h,t)}\|$$

  where $\|A_{B(h,t)}\|$ is the number of "active" AMs witnessing the block. This value is stored as `TotalScore` in the block header.

Table 1: Comparison of Consensus Mechanisms

Property	**Proof of Work (PoW)**	**Proof of Stake (PoS)**	**Proof of Authority (PoA)**
Primary Use	Public blockchains (Bitcoin)	Public blockchains (Ethereum)	Private/consortium blockchains
Energy Consumption	Very High	Low	Very Low
Throughput (TPS)	Low (7-10 for Bitcoin)	Medium to High	High
Decentralization	High	Medium to High	Low (centralized trust)
Security Basis	Computational work	Economic stake	Identity/reputation
Validator Selection	Miners with most hash power	Validators with stake	Pre-approved authorities

Each consensus mechanism represents a trade-off between decentralization, security, and efficiency, and is chosen based on the specific goals of the blockchain network.

7 Bitcoin

The Bitcoin Whitepaper and Network Architecture

Recall that in the abstract of this paper I asked you to search for "Financial Crisis 2008." That crisis was the reason Bitcoin was created; it was introduced in 2008 via the white paper **Bitcoin: A Peer-to-Peer Electronic Cash System** [9], authored by the pseudonymous *Satoshi Nakamoto*. The paper presented a solution for a decentralized digital currency without the need for a trusted third party.

The core architectural model of the Bitcoin network, as outlined by Nakamoto, consists of the following sequential process:

1. **Transaction Broadcast**: All new transactions are broadcast to every participating node (peer) on the network.
2. **Block Assembly**: Each node (miner) collects these new transactions into a candidate block.

3. **Proof-of-Work Competition**: Each node works to solve a computationally difficult cryptographic puzzle (the PoW algorithm) for its assembled block.

4. **Block Propagation**: The first node to solve the puzzle broadcasts the new, solved block to the entire network.

5. **Block Validation**: Other nodes receive the new block, verify that all transactions within it are valid and have not been spent before (no double-spending), and then accept the block.

6. **Chain Extension**: Nodes express their acceptance by starting work on the next block in the chain, using the hash of the newly accepted block as the previous hash.

The Genesis Block

The very first Bitcoin block, known as the Block #0, was mined by Satoshi Nakamoto on January 3, 2009. Its coinbase transaction (the special transaction awarding the mining reward) contains a notable text inscription:

```
The Times 03/Jan/2009 Chancellor on brink of second bailout for banks
```

This text is the headline from that day's edition of the British newspaper *The Times*. Its inclusion serves two profound purposes:

- **Timestamp Proof**: It provides independent, cryptographic proof that the block could not have been created before this date.
- **Philosophical Statement**: It is a clear commentary on the instability of the traditional financial system and the need for excessive government bailouts. Bitcoin was conceived as a direct response to this very problem - a decentralized, trustless financial system that operates without central banks or intermediaries.

Bitcoin Mining Farms

A Bitcoin mining farm is a large-scale dedicated facility housing hundreds or thousands of ASIC (application-specific integrated circuit) miners or, sometimes, GPU (graphics processing unit) miners. These are the industrial engines of the Bitcoin network, moving far beyond the hobbyist setup of a few home computers. GPUs were used in the early days of Bitcoin by hobbyist miners. ASICs, which are far more efficient, caused a massive increase in network difficulty, leading to industrial-scale mining and rendering GPU mining unprofitable for Bitcoin.

Mining Hardware: GPU vs. ASIC

Mining is the process of performing the computational work to secure the network and discover new blocks. Different types of hardware are used for this purpose. GPUs are realtively general-purpose processors designed for parallel tasks (e.g., graphics, scientific computing, AI). ASICs, as the name suggests, are microchips designed and manufactured for the sole purpose of executing the SHA-256 hash algorithm for Bitcoin mining.

Feature	GPU	ASIC
Efficiency & Speed	Moderate hash rate and energy efficiency. Flexible for different algorithms.	Extremely high hash rate and superior energy efficiency. The only viable hardware for competitive Bitcoin mining today.
Cost & Accessibility	Lower initial cost, widely available (e.g., gaming cards).	Very high initial cost, purchased from specialized manufacturers (e.g., Bitmain, MicroBT).
Flexibility	Can be used to mine various cryptocurrencies with different algorithms.	Completely inflexible; can only mine coins using the specific algorithm it was built for (SHA-256 for Bitcoin).

Non-Formal Overview

Farms are strategically located based on two critical factors:

- **Cheap, Reliable Electricity**: Electricity is the primary ongoing cost. Farms are often built near hydroelectric dams, geothermal sources, or in regions with energy subsidies.
- **Cooling Capacity**: ASICs generate immense heat. Large farms employ industrial ventilation, immersion cooling (submerging hardware in non-conductive fluid), or are located in cold climates for natural cooling.

Operations are managed professionally, with 24/7 monitoring, dedicated internet connectivity, and security. Participants often join **mining pools**, combining their computational power to have a more consistent chance of earning block rewards, which are then shared proportionally.

Costs?

The economic viability of a mining farm can be modeled as a profit function Π over a period T:

$$\Pi(T) = R(T) - C_{\text{op}}(T) - C_{\text{cap}}$$

Where:

- $R(T)$ is the total revenue from block rewards and transaction fees, approximated by:

  $$R(T) \approx \frac{H_{\text{farm}}}{H_{\text{network}}} \times B(T) \times P_{\text{BTC}}$$

  Here, H_{farm} is the farm's total hash rate, H_{network} is the global network hash rate, $B(T)$ is the total number of BTC mined in period T, and P_{BTC} is the market price of Bitcoin. This shows that a farm's share of rewards is directly proportional to its contribution to the network's total computational power.

- $C_{\text{op}}(T)$ is the operational cost, dominated by electricity: $C_{\text{op}}(T) \approx P_{\text{elec}} \times E_{\text{total}}(T)$, where P_{elec} is the price per kWh and E_{total} is the total energy consumed.
- C_{cap} is the capital expenditure, the initial cost of ASIC hardware, facility construction, and cooling infrastructure, often amortized over the hardware's lifespan.

A farm is profitable when $\Pi(T) > 0$. The global network hash rate H_{network} is a critical, constantly increasing variable in this equation, representing the competitive difficulty of mining. This model illustrates why scale, energy cost, and hardware efficiency are the decisive factors in industrial Bitcoin mining.

$\frac{1}{2}$

Bitcoin halving is a pre-programmed, periodic event that reduces the reward for mining new blocks by 50%. It is the core mechanism enforcing Bitcoin's predictable and diminishing supply, ultimately leading to the hard cap of exactly 21 million BTC. There will never be more than 21 million Bitcoins! From an economic perspective, Bitcoin halving is crucial because it systematically reduces new supply growth, creating predictable scarcity that supports lower inflation and long-term value preservation.

Mathematically, the block reward R at a given block height h is defined by:

$$R(h) = \frac{50}{2^{\lfloor h/210{,}000 \rfloor}} \text{ BTC}$$

where:

- h is the block height (Genesis Block is $h = 0$).
- 210,000 is the number of blocks between halvings (approximately every 4 years).

The sequence of rewards forms a geometric series: $50, 25, 12.5, 6.25, \ldots$ BTC. The total maximum supply $S_{\max}$ is the sum of this infinite series multiplied by the number of blocks per epoch:

$$\text{Total Number of Bitcoins} = 210{,}000 \times 50 \times \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k = 210{,}000 \times 50 \times 2 = 21{,}000{,}000 \text{ BTC}$$

8 Ethereum

Introduction & Historical Context

Ethereum, proposed in 2013 by Vitalik Buterin and launched in 2015, extends the blockchain concept beyond a simple ledger for currency. Its core innovation is the *Ethereum Virtual Machine (EVM)*, a decentralized, Turing-complete (meaning able to compute anything that is computable) global computer. This allows developers to write and deploy *smart contracts* - self-executing programs that run exactly as coded - enabling decentralized applications (dApps, more on this late) that are transparent, autonomous, and resistant to fraud.

The Ethereum Virtual Machine (EVM)

The EVM is the deterministic, sandboxed runtime environment for smart contracts on Ethereum. It is completely isolated, meaning code running inside the EVM has no access to the network, filesystem, or other processes, ensuring security and predictable execution across all nodes.

Formalism

Formally, the Ethereum state σ can be defined as a world-state mapping [10]:

$$\sigma : \mathcal{A} \to \mathcal{A}c$$

where $\mathcal{A}$ is the set of 160-bit addresses and $\mathcal{A}c$ is an account state. An account state is a tuple:

$$\mathcal{A}c = (n, b, s, c)$$

where:

- n is the *nonce* (number of transactions sent from this address, or for contracts, the number of creations).

- b is the account's *balance* in **wei**.
- s is the *storage root*, a 256-bit hash of the account's storage contents (Merkle tree variant).
- c is the *code hash*, the hash of the EVM code for contract accounts (for externally owned accounts, $c = \text{hash}(\emptyset)$).

A valid state transition is defined by the function δ:

$$\sigma_{t+1} = \delta(\sigma_t, T)$$

where T is a transaction. The EVM executes as a stack-based machine with a word size of 256 bits. Its execution can be modeled as a tuple sequence [11]:

$$(\mu, \Upsilon, \sigma) \rightarrow^* (\mu', \Upsilon', \sigma', o, g_r)$$

where μ is the machine state (stack, memory, program counter), Υ is the execution environment, o is the output, and g_r is the remaining gas.

Smart Contracts

A smart contract is a program stored on the blockchain that automatically executes predefined rules. Key features include autonomy (no third party), transparency (code is public), and immutability (code cannot be changed after deployment).

Smart Contract Programming

Smart contracts are primarily written in *Solidity*, a high-level programming language purpose-built for the EVM. Its syntax is heavily inspired by JavaScript, C++, and Python, making it accessible to a broad range of developers. Crucially, Solidity is an *object-oriented* language. Developers define *contracts*, which are analogous to classes in languages like Java or C++. These contracts encapsulate state variables (data) and functions (methods), support inheritance, and use modifiers (similar to decorators) to control function behavior. Another language, *Vyper* (with a syntax closer to Python), is also used but is less common.

The Role of Gas and Wei

To understand gas fees, one must first understand Ethereum's unit scale. The base unit is *wei*, named after Wei Dai, a cryptography pioneer. One *ether* (ETH) is defined as:

$$1 \text{ ETH} = 10^{18} \text{ wei}$$

Common denominations are:

- **Gwei (Giga-wei)**: 1 Gwei $= 10^9$ wei $= 10^{-9}$ ETH. This is the standard unit for quoting gas prices.
- **Ether**: The unit used for valuing assets and large transfers.

Every EVM operation (computation, storage) consumes *gas*, a unit of computational work. Users pay for gas in Gwei. This serves two purposes:

1. *Prevents Infinite Loops*: Execution halts when gas is exhausted, ensuring network liveness.
2. *Allocates Resources*: Miners (pre-Merge) and validators (post-Merge) are compensated for their computational and infrastructural work.

The total gas cost for a transaction T_x is:

$$g_{total} = g_{base} + g_{exec} + g_{data}$$

where g_{base} is a fixed cost, g_{exec} is the sum of opcode costs, and g_{data} is cost for non-zero transaction data bytes [12]. The fee in wei is: $\text{Fee} = g_{total} \times \text{gasPrice}_{(\text{Gwei})} \times 10^9$.

Lifecycle & Best Practices Contracts are deployed via a special transaction. Contracts on Mainnet (the primary, live blockchain network where real transactions and native tokens are processed) are permanent. Updates require deploying a new contract. Therefore, rigorous testing on testnets (like Sepolia) is essential before mainnet deployment.

Formalizing a Simple Contract

Consider a contract C for an asset sale. Its state includes variables for *seller*, *buyer*, *price*, and *state*. It defines functions like *list()*, *buy()*, and *transfer()*. The execution E of function *buy()* with arguments and remaining gas g can be modeled as:

$$E(\sigma, g, C, buy, I) = (\sigma', g', A)$$

where I is the call input (e.g., buyer address, payment), σ' is the updated state (ownership transferred, balance updated), g' is remaining gas, and A is a set of log events.

Token Standards: ERC-20, ERC-721, ERC-1155

Ethereum's flexibility is standardized through "Ethereum Request for Comments" (ERC) standards. These standards define a common interface, ensuring tokens are compatible across wallets and dApps. The core concept is *fungibility*:

- **Fungible**: Items are interchangeable because each unit is identical. Traditional currency (e.g., one $1 bill for another) or Bitcoin are fungible.
- **Non-Fungible**: Items are unique and not directly interchangeable. A deed to a specific house or a unique painting is non-fungible.
- **Semi-Fungible**: An item can have properties of both. For example, a concert ticket for a specific seat is unique (non-fungible), but before the event, all unsold tickets for the same section might be treated as interchangeable (fungible) for bulk operations.

Standard	Type	Key Characteristics and Use Case
ERC-20 (2015)	Fungible Token	Tokens are identical and interchangeable. Has functions like `transfer()` and `balanceOf()`. Used for cryptocurrencies, governance tokens.
ERC-721 (2018)	Non-Fungible Token (NFT)	Each token has a unique ID and is distinct. Has functions like `ownerOf(tokenId)`. Used for digital art, collectibles, real-world asset deeds.
ERC-1155 (2019)	Multi-Token / Semi-Fungible	A single contract can manage multiple token types (fungible, non-fungible, semi-fungible). Efficient for gaming assets and metaverse economies.

Ethereum 2.0: The Transition to Proof of Stake

The "Merge" in September 2022 transitioned Ethereum from energy-intensive Proof of Work (PoW) to Proof of Stake (PoS) under the "Eth2" roadmap. This was a change *only to the consensus layer, not the EVM or smart contracts.*

Clarification on Layers Ethereum's architecture can be conceptually split into two main layers:

1. **Execution Layer**: This contains the EVM, smart contracts, account states, and transaction pool. It defines *what* the next state of the blockchain should be.

2. **Consensus Layer**: This contains the protocol (PoW or PoS) that determines *who* gets to propose the next valid block and how nodes *agree* on the canonical chain.

The Merge swapped the consensus mechanism from PoW to PoS. The execution layer - including the EVM, gas rules, smart contract bytecode, and all existing dApps - remained *completely unchanged*. Smart contracts do not "use" PoW or PoS; they are simply executed by the network's nodes (miners or validators). The transition only changed the rules for selecting which node publishes the next block, making the network more energy-efficient and secure. The logic, security, and behavior of every deployed smart contract were preserved exactly as before.

9 Layer Solutions

The Blockchain Trilemma

A fundamental challenge in blockchain design, articulated by Ethereum founder Vitalik Buterin, is the *Blockchain Trilemma*. It posits that a decentralized network can, at any given time, robustly achieve only two of the following three properties:

- **Decentralization**: The system's governance and operation are distributed across a wide set of participants, avoiding control by a single entity or small group.

- **Security**: The network can resist and survive attacks (e.g., 51% attacks, Sybil attacks) and fraudulent transactions without incurring prohibitive costs.

- **Scalability**: The network can handle a growing number of transactions per second (TPS) without a corresponding increase in latency or cost.

If we represent the strength of each property as D (Decentralization), S (Security), and C (Scalability), the trilemma suggests a constraint: maximizing any two inherently limits the third. For instance, Bitcoin prioritizes D and S, leading to lower C (≈ 7 TPS). The quest for modern blockchains is to find architectural solutions that push the boundaries of this trade-off. The primary strategy involves creating a multi-layered architecture, where the base layer (Layer 1) guarantees security and decentralization, while additional layers (Layer 0, Layer 2) are built to enhance scalability and functionality.

Layer 0: The Network Infrastructure Layer

Layer 0 provides the foundational *"blockchain of blockchains"* infrastructure - the protocols and networks that enable different, independent blockchains (Layer 1) to interoperate, share security, and communicate seamlessly.

Purpose & Function: It solves the problem of isolated blockchains operating as "siloed islands" (separate networks that can't easily share data or assets). Key functions include:

- Providing the underlying peer-to-peer network and data transport protocols.

- Enabling secure cross-chain communication and messaging.

- Allowing new Layer 1 blockchains to be created and launched with shared security models.

Examples:

- **Polkadot**: Uses a central Relay Chain for shared security, connecting custom parallel blockchains called *Parachains*.
- **Cosmos**: Employs the Inter-Blockchain Communication (IBC) protocol to allow sovereign, application-specific blockchains (Zones) to transfer data and tokens.
- **Avalanche**: Features a primary network that secures multiple built-in and custom blockchains (Subnets) running in parallel.

Layer 1: The Base Protocol Layer

Layer 1 is the core, foundational blockchain itself. It is the settlement layer where transactions are ultimately ordered, validated, and finalized.

Purpose & Components: This layer defines the essential rules of the network. Its key components are:

- The **consensus mechanism** (e.g., PoW, PoS).
- The **data structure** (chain of blocks).
- The **native cryptocurrency** (e.g., BTC, ETH) used for fees and security.
- The **virtual machine** (e.g., EVM) for smart contract execution (if applicable).

Its primary function is to maintain a secure, decentralized, and immutable ledger. Most trilemma compromises are made at this layer.

Examples:

- **Bitcoin**: The archetypal Layer 1, optimized for decentralized, secure value transfer.
- **Ethereum**: A general-purpose Layer 1 supporting Turing-complete smart contracts and dApps.
- **Solana, Cardano, BNB Chain**: Other prominent Layer 1s with varying consensus models (e.g., Proof-of-History, Ouroboros) aiming for different points on the trilemma spectrum.

Layer 2: The Scaling & Enhancement Layer

Layer 2 solutions are protocols built *on top of* a Layer 1 blockchain. Their primary goal is to **offload transaction processing** from the main chain to achieve greater scalability (higher TPS, lower fees, faster confirmation) while still inheriting the security guarantees of the underlying Layer 1.

Core Idea & Security Model: Transactions are executed outside Layer 1 (off-chain), but their final proofs or batched summaries are periodically posted back to Layer 1 (on-chain). This allows Layer 1 to act as a secure arbitration and settlement layer. The security can be modeled as: if the Layer 2 protocol fails or acts maliciously, users can always fall back to the Layer 1 to recover their assets (a property known as *censorship resistance* or *self-custody*).

Primary Types & Examples:

- **Rollups (Ethereum)**:
  - Execute transactions outside Ethereum but post transaction data (*Optimistic Rollups*) or cryptographic proofs (*ZK-Rollups*) to the main chain.
  - **Examples:** Arbitrum, Optimism (Optimistic); zkSync, StarkNet (ZK).
- **State & Payment Channels (Bitcoin, Ethereum)**:
  - Allow participants to conduct numerous off-chain transactions, only settling the net result on-chain.
  - **Example:** The Lightning Network for Bitcoin.
- **Sidechains (Ethereum)**:
  - Independent blockchains with their own validators, connected to the main chain via a two-way bridge. They offer scalability but have separate security assumptions.
  - **Example:** Polygon POS Chain (though the Polygon ecosystem now includes various L2 solutions).

10 Exchanges

Introduction to Crypto Exchanges

Cryptocurrency exchanges are digital marketplaces where users can trade digital assets, such as Bitcoin for Ethereum, or convert cryptocurrency to and from traditional fiat currency (like USD or EUR). Based on their underlying architecture and philosophy, they are categorized into two fundamental types: Centralized Exchanges (CEX) and Decentralized Exchanges (DEX).

Centralized Exchanges (CEX)

A Centralized Exchange (CEX) is a platform operated by a specific company or organization. It functions much like a traditional stock exchange or bank, acting as a trusted intermediary and custodian for user funds.

Core Architecture & Process

In a CEX, users create an account by depositing funds into wallets controlled by the exchange. The exchange's internal ledger tracks ownership, and trades occur within this private ledger. When User A buys Bitcoin from User B, the exchange updates its internal database; the actual on-chain transaction may only occur when funds are deposited or withdrawn. This requires users to complete Know Your Customer (KYC) and Anti-Money Laundering (AML) verification, providing personal identification.

- KYC (Know Your Customer): a process by which firms verify the identity of their customers to prevent fraud and ensure compliance with legal and regulatory requirements.
- AML (Anti-Money Laundering): laws and procedures designed to detect, prevent, and report attempts to disguise illegally obtained funds as legitimate.

Advantages

- **User-Friendly**: Offers intuitive interfaces, customer support, and educational resources suitable for beginners.
- **Fiat Integration**: Direct support for depositing and withdrawing traditional currency via bank transfers or cards.
- **Speed**: Trades are instant as they occur off-chain.

Challenges & Risks

- **Custodial Risk**: Users do not control their private keys. The exchange holds the assets, creating third-party risk - if the exchange is hacked, becomes insolvent, or acts maliciously, users can lose their funds.
- **Privacy Concerns**: Requires submission of extensive personal data for KYC/AML compliance.
- **Central Point of Failure**: The platform can suffer downtime, freeze withdrawals, or de-list assets based on company or regulatory decisions.
- **Potential for Manipulation**: As private entities, their internal operations (like trading volume) are not fully transparent.

Decentralized Exchanges (DEX)

A Decentralized Exchange (DEX) is a protocol that enables peer-to-peer trading directly between users' wallets without an intermediary holding funds. It is typically implemented as one or more smart contracts on a blockchain like Ethereum.

Core Architecture & Process

Trades on a DEX are executed automatically by smart contracts. Users connect their personal wallets (e.g., MetaMask) to the DEX interface. To provide liquidity, users can lock their assets in a smart contract called a *liquidity pool*. Trades are executed against these pools using automated market maker (AMM) algorithms. All transactions are settled directly on-chain.

Advantages

- **Self-Custody**: Users retain control of their private keys and funds at all times. The smart contract only facilitates the swap.
- **Permissionless & Private**: No account registration or KYC is required. Trading is accessible to anyone with a crypto wallet.
- **Censorship-Resistant**: No central authority can freeze assets or prevent a user from trading.
- **Transparency**: All transaction logic and pool reserves are verifiable on the public blockchain.

Challenges & Risks

- **Technical Barrier**: Requires understanding of wallets, gas fees, and private key management. A user error can lead to permanent loss.
- **Network Dependency**: Speed and cost are tied to the underlying blockchain (e.g., high gas fees on Ethereum during congestion).
- **Impermanent Loss**: Liquidity providers are exposed to the risk that the value of their deposited assets changes compared to simply holding them.

The choice between a CEX and a DEX represents a fundamental trade-off between convenience and control, a theme central to the broader blockchain ecosystem. For beginners and those trading with fiat, CEXs offer a familiar gateway. For users prioritizing sovereignty, privacy, and direct interaction with decentralized finance (DeFi), DEXs provide the essential infrastructure.

11 Tokenomics

Market indicators are quantitative metrics derived from financial data to estimate overall market dynamics, sentiment, and trends. Unlike individual security analysis, they provide a macro view for traders and investors to forecast movements, confirm trends, and manage risks. These tools, plotted separately from price charts, help filter noise in volatile markets.

Intuition: Think of them as the "pulse" of the market - revealing whether a price increase is broadly supported or driven by just a few big stocks, much like checking if an entire crowd is cheering or only a small group.

Before diving in, here are some key terms you'll see a lot:

- Altcoins: Any cryptocurrency other than Bitcoin, often offering alternative features or use cases.
- **Asset**: Something valuable that you own, like stocks, crypto, or even a house - basically anything that can make you money or hold value.
- **Equity**: Ownership in a company through stocks (also called shares). It represents your "piece of the pie" in the business.
- **Outstanding shares**: The total number of shares a company has given out to investors that are currently held by the public (or institutions). Differs from "just shares" because companies might have authorized more shares they haven't issued yet, or they buy back some (treasury shares) that don't count as outstanding.
- **Liquidity**: How easily you can buy or sell something without messing up its price too much. High liquidity means lots of buyers and sellers - like cash in your pocket, ready to use fast.
- **Bullish/Bearish**: Bullish means optimistic, expecting prices to go up (like a bull charging upward). Bearish means pessimistic, expecting prices to go down (like a bear swiping downward).
- **Put options**: Bets that a stock price will fall - like buying "down insurance" on a stock.
- **Call options**: Bets that a stock price will rise - like buying "up insurance" on a stock.
- **Hedging**: Protecting your investments from losses, kind of like buying insurance to cover bad scenarios.

- **Circulating Supply**: Tokens that are currently in circulation.
- **Locked Tokens**: Tokens that exist but cannot be transferred or spent until specific conditions are met.
- **Total Supply**: The sum of circulating tokens and locked tokens.
- **Max Supply**: All tokens that will ever be produced.

Market Breadth

Measures participation in price moves across securities. The advance-decline line ratios advancing versus declining stocks, often weighted by market cap (e.g., NYSE Advance-Decline Index). A rising line signals broad optimism; divergences from indices warn of reversals.

Intuition/Example: If the S&P 500 hits new highs but the advance-decline line lags (bearish divergence), it suggests only mega-caps like NVIDIA are pushing the index up, while most stocks weaken - often coming before price drops, as seen before the 2008 bear market.

Market Sentiment

Contrasts prices with volume to assess investor psychology. The put-call ratio (put options traded/call options) above 1 indicates bearish fear; below 0.7 suggests bullish greed. High ratios often come before price bounces.

Intuition/Example: Extreme fear (ratio > 1.2) means investors are heavily protecting against drops with puts - like buying insurance during panic - which oddly signals potential bottoms, as overly pessimistic crowds are often wrong and markets bounce back.

Moving Averages

Smooth price data over periods (e.g., 50-day SMA: $\frac{\sum_{i=1}^{50} P_i}{50}$). Upward slopes confirm uptrends; crossovers (e.g., 50-day over 200-day) generate buy/sell signals.

Intuition/Example: The "golden cross" (50-day crossing above 200-day) signals bullish momentum shift, like short-term enthusiasm overtaking long-term averages - historically marking bull market starts; opposite "death cross" warns of bearishness.

On-Balance Volume (OBV)

Cumulative volume indicator: $OBV_t = OBV_{t-1} \pm V_t$ (volume) where we add V_t if price rises, subtract V_t if falls, unchanged $V_t = 0$ otherwise. Rising OBV validates uptrends; divergences highlight potential shifts.

Intuition/Example: If price hits new highs but OBV makes lower highs (bearish divergence), it shows buying lacks volume conviction - "smart money" isn't participating, often coming before reversals.

Market Capitalization

Market capitalization (market cap) quantifies a company's equity value as perceived by the market:

$$\text{Market Cap} = \text{Current Share Price} \times \text{Number of Outstanding Shares}.$$

Distinct from enterprise value (equity + debt), it ranks firms by size for comparison.
Categories:

- **Large Cap** ($>$ \$10B): Stable giants (e.g., NVIDIA at over \$4.2T in late 2025); lower risk.

- **Mid Cap** ($1B–$10B): Growth-focused, moderate volatility.
- **Small Cap** ($250M–$1B): High-reward potential, riskier.
- **Micro Cap** (< $250M): Speculative, research-intensive.

Investors use market cap for diversification - large caps for stability, small for growth - balancing risk and opportunity.

Intuition/Example: Mega-caps like NVIDIA dominate indices due to size, so market price increases can feel strong even if smaller stocks lag; Tesla (around $1.5T in late 2025) exemplifies growth volatility in large-cap space.

Bullish and Bearish Sentiments

Bullish: Optimism expecting price rises. Indicators: 20%+ recovery from lows, rising volumes. Prices ascend (e.g., $100 to $120). Strategies: Buy/hold during uptrends, leveraging momentum.

Intuition/Example: Post-2022 recovery into 2025 saw broad optimism with AI-driven gains in tech, though narrow breadth raised caution.

Bearish: Pessimism anticipating declines. Indicators: 20%+ drop from peaks, fear gauges like VIX spikes. Prices fall (e.g., $200 to $100). Strategies: Sell/short, protect with puts.

Intuition/Example: Early 2025 corrections amid the US tariff fears showed pessimistic sentiment, but quick bounces highlighted rapid shifts - monitor for extremes like 2008 crisis big drops.

Volume of Trade

Trading volume tallies shares/contracts exchanged daily, signaling liquidity and conviction:

$$\text{Volume} = \sum(\text{Buy Shares} + \text{Sell Shares}).$$

High volume eases execution, peaks at session opens/closes or Mondays/Fridays. Driven by HFT (High-Frequency Trading) and funds.

Relation to prices: Confirms trends - high volume on breakouts validates moves; low suggests weakness. In technical analysis, average volume over periods informs entry/exit, especially in options/futures where uncertainty amplifies activity.

Intuition/Example: A stock breaking resistance on 2x average volume shows strong conviction (likely continuation); low-volume breakouts often fail as "fakeouts," trapping eager buyers.

Marked Dominance

Market dominance in crypto measures the portion of the overall cryptocurrency market that a particular coin - typically Bitcoin - accounts for. For instance, a 50% Bitcoin dominance means Bitcoin alone represents half of the total market value. High dominance usually reflects strong investor confidence in that coin, whereas lower dominance can indicate growing interest in altcoins. Analyzing dominance helps traders gauge market trends, such as whether the market is concentrating on Bitcoin or spreading into other assets.

Economic Models & Emissions

The **inflationary model** has no capped supply, allowing token emission to continue indefinitely. In contrast, the **deflationary model** imposes a fixed maximum supply, so emissions gradually deplete the remaining pool.

As of December 2025, Bitcoin follows a deflationary model with a maximum supply of 21 million coins. Approximately 19.95 million BTC have already been mined, leaving about 1.05 million btc still to be created.

Fear & Greed Index

The Fear & Greed Index measures market sentiment, indicating whether fear or greed is driving investor behavior. Scored from 0 to 100, low values signal extreme fear, while high values reflect extreme greed. When the index spikes into the greed zone, many investors treat it as a cautionary signal to tighten risk controls and consider taking profits.

12 Blockchain Security

Blockchain is often praised for being highly secure thanks to its decentralized structure and cryptographic protections. However, it's not invincible. Understanding how these systems can be attacked is important for protecting user funds and data, keeping the network running smoothly, preventing smart contract exploits, and making consensus mechanisms more robust.

Blockchain systems can be attacked at several different levels:

- **Application Layer**: Wallets, user interfaces, and dApps can be compromised through bugs in the code or by tricking users.
- **Network Layer**: Since nodes communicate over the internet, they can be isolated, delayed, or spoofed by attackers.
- **Consensus Layer**: The rules that help nodes agree on the blockchain's state can be exploited by attackers trying to manipulate how blocks are validated.
- **Smart Contracts**: Self-executing programs on the blockchain might have bugs or logic errors that attackers can take advantage of.
- **Social Engineering**: Humans are often the weakest link - attackers exploit trust and carelessness to bypass even the strongest technical defenses.

12.1 Majority Hash Power Attack

When one person or group controls more than half of a blockchain's computing power (or staked tokens), they can manipulate how the network reaches consensus. This is more likely to happen on smaller blockchains that have fewer miners or validators.

What Can Happen

With majority control, an attacker can:

- **Double-spend coins**: They can create an alternative version of the blockchain where they never spent certain coins, effectively spending them twice.
- **Block specific transactions**: They can prevent certain transactions from being confirmed.
- **Disrupt block creation**: They can slow down or temporarily stop new blocks from being added.

That said, there are limits. Attackers can't steal coins from wallets they don't control, and they can't easily change blocks that are already deep in the chain. But even with these limitations, such attacks seriously damage trust in the network.

Real-World Example

In 2020, Ethereum Classic was hit by multiple 51% attacks. The attacker gained majority control and reorganized the blockchain to reverse their own transactions after already receiving goods or services in return. This led to losses of several million dollars across multiple incidents.

12.2 Identity Multiplication Attack

In this type of attack, a single attacker creates many fake identities or nodes to gain unfair influence in the network. The name comes from the 1973 book *Sybil*, which tells the story of a woman with 16 distinct personalities.

What Attackers Want to Achieve

By creating fake identities, attackers can:

- Flood voting and consensus processes with fake votes they control.
- Manipulate how nodes connect to each other, controlling what information flows through the network and isolating honest nodes.
- Undermine the trustless nature that decentralized systems promise.

How to Defend Against It

Several strategies help prevent identity multiplication attacks:

- **Resource-based consensus**: Proof-of-Work and Proof-of-Stake make creating fake identities expensive - either you need to spend a lot on computing power or lock up real money.
- **Reputation systems**: Tracking historical behavior and requiring some form of identity verification can limit the impact of brand-new identities.
- **Limiting influence**: Designing protocols so that no single identity can have too much power over network decisions.

12.3 Smart Contract Vulnerabilities

Smart contracts present a unique security challenge because they're immutable. Unlike regular software where you can patch bugs after deployment, once a smart contract is on the blockchain, it usually can't be changed. This means every bug becomes a permanent vulnerability, making thorough security testing before deployment absolutely critical.

Common Types of Vulnerabilities

Reentrancy Attacks

This happens when a contract makes an external call before updating its own state. A malicious contract can keep calling back into the vulnerable function, draining funds before the balance is updated. The famous DAO hack in 2016 used this technique to steal about $60 million worth of Ether, eventually leading to Ethereum's hard fork.

Integer Overflow and Underflow

When numbers get too big or too small for their variable type, they can wrap around unexpectedly. Before Solidity 0.8.0 added automatic checks, these errors could go unnoticed, causing account balances to jump from zero to the maximum value or vice versa.

Access Control Problems

Sometimes functions that should only be callable by certain users (like the contract owner) are accidentally left open to everyone. Or the permission checks might have flaws that let unauthorized people get through. This can allow attackers to take over contract functions or steal locked funds.

Front-Running

Since pending transactions are visible in the mempool before they're confirmed, attackers can see profitable transactions and submit their own with higher gas fees to get processed first. This is especially common in DeFi, where attackers can manipulate prices or exploit arbitrage opportunities.

12.4 Social Engineering and Phishing

Even the most secure blockchain technology can be defeated if attackers go after people instead of code. Social engineering works by manipulating human psychology and trust to get around technical security measures.

Common Attack Methods

Fake Websites and Wallets

Attackers create convincing copies of real cryptocurrency platforms, often using similar-looking domain names with tiny differences. These fake sites are designed to steal your seed phrase, private keys, or login credentials when you think you're accessing the real service.

Impersonating Support Staff

Scammers pretend to be official support on platforms like Telegram, Discord, Reddit, or Twitter. They use people's trust in support staff to trick them into revealing private keys or sending funds. Remember: real support will never ask for your private keys or seed phrase.

Phishing Emails and Messages

These attacks come through email, direct messages, or social media, offering fake token airdrops, fake security alerts, or fake investment opportunities. They usually try to create a sense of urgency to pressure you into acting quickly without thinking.

12.5 Privacy-Focused Cryptocurrencies

Privacy coins are cryptocurrencies designed specifically to keep your transactions private and anonymous. Unlike Bitcoin or Ethereum where all transaction details are visible on the public blockchain, privacy coins use special techniques to hide who's sending, who's receiving, how much is being sent, and the transaction history.

Well-known examples include Monero ($XMR), which makes all transactions private by default; Zcash ($ZEC), which lets users choose between private and public transactions; and Dash ($DASH), which offers optional privacy features.

Privacy coins use advanced cryptography to hide transaction details while still keeping the blockchain verifiable:

Ring Signatures (Monero)

This mixes the real sender's signature with several decoy signatures from random addresses on the blockchain. The result is a "ring" where any member could have sent the transaction. The blockchain can verify that one legitimate signature exists in the ring, but can't tell which one is real.

Stealth Addresses (Monero)

Every transaction creates a unique, one-time receiving address that's mathematically derived from your public address but can't be linked to it by outside observers. Even if someone knows your public Monero address, they can't see which transactions you've received.

Confidential Transactions

This technique hides transaction amounts using cryptographic commitments. The blockchain can still verify that the transaction is valid (inputs equal outputs) without revealing the actual numbers. This uses mathematical constructions like Pedersen commitments and range proofs.

Zero-Knowledge Proofs (Zcash)

Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) let you prove something is true without revealing any other information. In Zcash, users can choose to send "shielded" transactions that are completely private, or "transparent" transactions that work like Bitcoin. With shielded transactions, zk-SNARKs prove the transaction is valid while hiding the sender, receiver, and amount.

How Zero-Knowledge Proofs Work

A zero-knowledge proof involves two parties: a prover P who wants to prove something, and a verifier V who checks the proof. For a statement x and secret witness w, the system needs three properties:

1. **Completeness**: If the statement is true and everyone follows the rules, the verifier will accept:
$$\mathbb{P}(V(x, \pi) = \text{accept}) = 1$$
where π is the proof created by $P(x, w)$.

2. **Soundness**: If the statement is false, no cheating prover can create a convincing fake proof (except with extremely tiny probability):
$$\mathbb{P}(V(x, \pi^*) = \text{accept}) \leq \epsilon$$
for any fake proof π^* and very small ϵ.

3. **Zero-Knowledge**: The verifier learns nothing except whether the statement is true. Formally, there's a simulator S that can create fake transcripts that look just like real ones, without knowing the secret w:
$$\{V(P(x, w))\} \approx \{S(x)\}$$
where $\approx$ means computationally indistinguishable.

Privacy vs. Regulation

Privacy coins exist in a complicated legal space. They offer real benefits like financial privacy (similar to using cash), protection from surveillance, and safety from targeted attacks based on visible wealth. But these same privacy features have made regulators worried about potential misuse for money laundering or illegal activities.

As a result, many cryptocurrency exchanges have removed privacy coins from their platforms, either due to regulatory pressure or to avoid future problems. Some countries have banned or restricted privacy coin usage. The debate between privacy rights and regulatory needs continues to be a major issue in the cryptocurrency world.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976. (Introduces discrete logarithm problem and discusses computational hardness).

[2] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976. (Original paper proving security of Diffie-Hellman key exchange protocol).

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978. (Original RSA paper, security based on factoring hardness).

[4] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer, 1998.

[5] V. S. Miller, "Use of elliptic curves in cryptography," CRYPTO '85, 1985.

[6] J. M. Pollard, "Monte Carlo methods for index computation mod p," *Math. Comp.*, 1978.

[7] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *J. Cryptology*, 2001.

[8] VeChain Foundation, "Consensus Deep Dive", VeChain Documentation.

[9] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System"

[10] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Project Yellow Paper, 2014.

[11] Y. Hirai, "Defining the Ethereum Virtual Machine for Interactive Theorem Provers," in *Financial Cryptography and Data Security*, 2017.

[12] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, 2014.

[13] V. Buterin et al., "Combined Fault Tolerance and Consensus Mechanism for Ethereum 2.0," Ethereum Research, 2020.

[14] *Corporate Finance Institute* Website: `https://corporatefinanceinstitute.com/`