

CHƯƠNG 4. MẠNG MÁY TÍNH VÀ BLOCKCHAIN

Mục tiêu chương

- Hiểu được nguyên lý hoạt động của công nghệ Blockchain và mô hình mạng ngang hàng (P2P) trong bối cảnh mạng máy tính.
- Phân tích cấu trúc khối dữ liệu, cơ chế mã hóa và các giao thức đồng thuận trong Blockchain như **PoW**, **PoS**, **PBFT**.
- Giải thích được vai trò của thuật toán **SHA-256** và cấu trúc **Merkle Tree** trong đảm bảo tính toàn vẹn và bảo mật dữ liệu.
- Thực hành xây dựng mô hình chuỗi khối đơn giản và triển khai **Smart Contract** bằng Python với thư viện **Web3.py**.
- Nhận biết khả năng ứng dụng Blockchain vào khoa học dữ liệu thông qua mô hình lưu trữ phi tập trung và xác thực dữ liệu.
- Hiểu được vai trò và chức năng của ví điện tử (wallet) trong quá trình tương tác, lưu trữ và xác thực giao dịch trên Blockchain.

Mở đầu

- ❑ Blockchain, dựa trên mô hình mạng ngang hàng (Peer-to-Peer - P2P) để hoạt động mà không cần sự kiểm soát tập trung.
- ❑ Blockchain không chỉ là nền tảng cho tiền điện tử như Bitcoin hay Ethereum mà còn *ứng dụng rộng rãi trong tài chính, chuỗi cung ứng, y tế, và bảo mật dữ liệu.*
- ❑ Khả năng đảm bảo tính toàn vẹn dữ liệu, chống giả mạo và phân tán thông tin một cách an toàn,

Các thế hệ phát triển

- ❑ Giới thiệu lần đầu vào năm 2008 bởi [Satoshi Nakamoto](#) với hệ thống Bitcoin (Blockchain 1.x), đến nay Blockchain đã trải qua các thế hệ:
- Blockchain 1.0 (2008): Giao dịch **tiền điện tử ngang hàng**, khởi nguồn với **Bitcoin**.
- Blockchain 2.0 (2015): **Ethereum** ra đời, đưa vào khái niệm *hợp đồng thông minh*, cho phép lập trình các điều kiện giao dịch.
- Blockchain 3.0 (từ 2017): Mở rộng ra ngoài lĩnh vực tài chính, hỗ trợ các ứng dụng phi tập trung (**DApps**) như quản lý danh tính, giáo dục, y tế.
- Blockchain 4.0 (từ 2020): Tập trung vào **tích hợp doanh nghiệp**, cải thiện khả năng mở rộng, *liên chuỗi* và tích hợp các công nghệ như **IoT, AI**.



4.1. BLOCKCHAIN

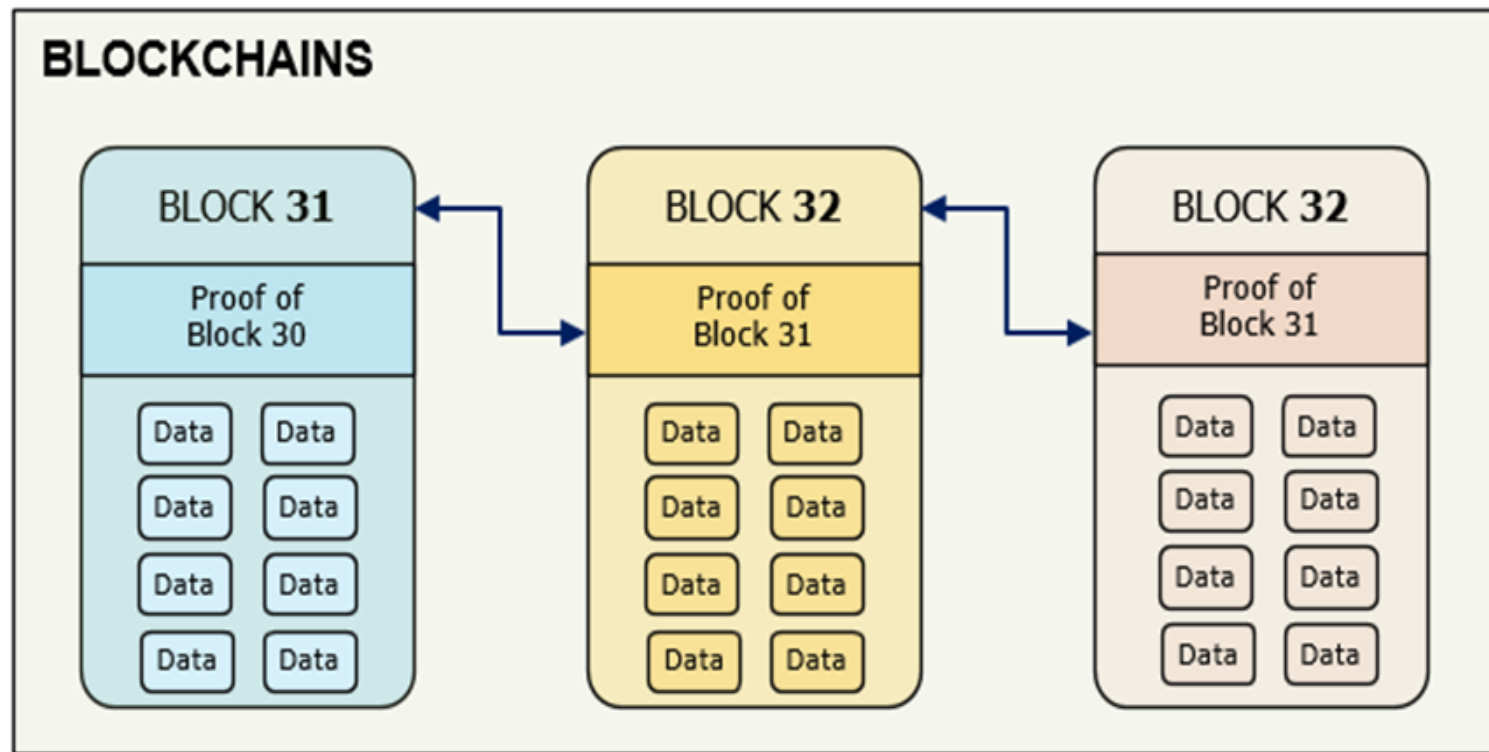
4.1.1. Định nghĩa

Blockchain là một công nghệ lưu trữ và truyền tải dữ liệu theo mô hình **sổ cái phân tán** (**Distributed Ledger Technology – DLT**), hoạt động **phi tập trung** trên nền mạng ngang hàng (**Peer-to-Peer – P2P**), trong đó *dữ liệu được ghi lại dưới dạng các khối (blocks) phát triển theo thứ tự thời gian và được bảo mật bằng các thuật toán mật mã.*

Mỗi khối chứa một tập hợp dữ liệu – thường là các *giao dịch* – cùng với **mã băm (hash) của khối trước đó**, tạo thành ***một chuỗi khối liên tục và không thể thay đổi.***

Việc *thêm và xác thực các khối mới* được thực hiện thông qua ***các cơ chế đồng thuận giữa các nút trong mạng*** [2, 5, 10]

- ✓ Mỗi khối chứa dữ liệu và bằng chứng (**Proof-Proof of Previous Block**) của khối trước đó đảm bảo tính toàn vẹn của chuỗi.
- ✓ Trong hình 4.1, có hai khối khác nhau mang bằng chứng của cùng một khối (Block 31), thể hiện hiện tượng phân nhánh (**fork**), khi có nhiều phiên bản Block 32 được đề xuất trong mạng ngang hàng (P2P).
- ✓ Hệ thống sẽ chọn một nhánh hợp lệ theo cơ chế đồng thuận.



Hình 4.1: Cấu trúc chuỗi Blockchain

Đặc điểm

1. Bất biến (Immutability):

Một khi dữ liệu đã được ghi vào Blockchain thì không thể thay đổi hoặc xóa bỏ. Điều này đảm bảo tính toàn vẹn và chống lại hành vi gian lận.

2. Phi tập trung (Decentralization):

Không có thực thể trung gian kiểm soát hoàn toàn chuỗi khối. Các nút (nodes) tham gia vào mạng đều có quyền ngang nhau trong việc xác minh và ghi nhận giao dịch.

3. Minh bạch (Transparency):

Mọi giao dịch đều có thể được các nút trong mạng xác minh. Mặc dù dữ liệu được mã hóa lịch sử giao dịch vẫn có thể được truy xuất công khai trên chuỗi.

6. Khả năng lập trình (Programmable):

Blockchain cho phép triển khai các logic tự động thông qua hợp đồng thông minh (Smart Contracts), đặc biệt trong các nền tảng như Ethereum.



Blockchain

4. Bảo mật cao (High Security):

Dữ liệu trong Blockchain được bảo vệ bằng các thuật toán mã hóa như SHA-256, đảm bảo rằng chỉ có những người có khóa giải mã mới có thể truy cập nội dung.

5. Không cần sự tin tưởng (Trustless):

Các bên tham gia không cần phải tin tưởng lẫn nhau, vì các giao dịch được xác thực bởi một cơ chế đồng thuận minh bạch và tự động.

4.1.2. Cách Blockchain ứng dụng mô hình mạng phân tán

- ✓ Blockchain sử dụng mô hình mạng ngang hàng (Peer-to-Peer - P2P) để đảm bảo tính **phi tập trung và bảo mật dữ liệu**.
- ✓ Trong mô hình này, không có máy chủ trung tâm kiểm soát, mà thay vào đó, tất cả các nút trong mạng đều có quyền ngang nhau trong việc xác minh và ghi nhận giao dịch.

❑ Cơ chế truyền tải dữ liệu:

- Khi một giao dịch mới phát sinh, nó sẽ được truyền tới tất cả các nút trong mạng.
- Các nút sẽ kiểm tra tính hợp lệ của giao dịch dựa trên các quy tắc đã được thiết lập.

❑ Xác thực và ghi nhận giao dịch:

- Giao dịch hợp lệ sẽ được tập hợp vào một khối mới và phát sóng đến các nút khác để xác thực.
- Các nút sử dụng cơ chế đồng thuận (như PoW, PoS) để thống nhất việc ghi nhận khối mới vào chuỗi.

So sánh Blockchain với các mạng phân tán khác:

Blockchain

Blockchain tập trung vào việc ghi nhận dữ liệu vĩnh viễn và không thể thay đổi

Blockchain

Mạng phi tập trung (Decentralized)

Không có máy chủ trung tâm, mọi giao dịch đều được xác thực bởi các nút ngang hàng.

Đều là mạng P2P

BitTorrent

Chia sẻ tập tin tạm thời.

Client-Server

Mạng tập trung (Centralized)

Có máy chủ điều khiển dữ liệu

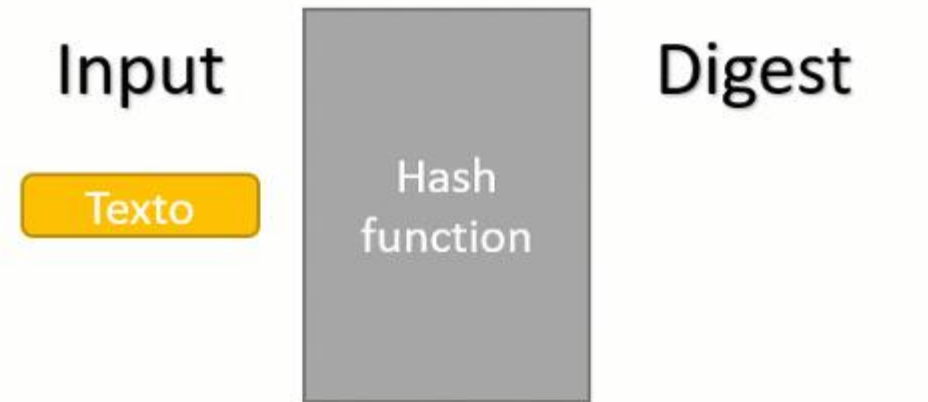
Vai trò của mô hình P2P trong Blockchain:

- ❑ Chống kiểm duyệt (Censorship Resistance): Không có thực thể nào có thể kiểm soát toàn bộ mạng, do đó không thể ngăn chặn giao dịch hợp lệ.
- ❑ Bảo mật cao hơn: Dữ liệu được phân phối trên nhiều nút, việc tấn công vào một nút đơn lẻ không ảnh hưởng đến toàn mạng.
- ❑ Tính mở rộng (Scalability): Số lượng nút càng nhiều thì khả năng xử lý giao dịch càng tăng, khác với mô hình tập trung có giới hạn về tài nguyên máy chủ.

4.1.3. Hàm băm và giá trị băm trong Blockchain

- ❑ Hàm băm (**hash function**) là một hàm toán học một chiều, thực hiện *chuyển đổi một chuỗi đầu vào có độ dài bất kỳ thành một chuỗi giá trị đầu ra cố định, thường có dạng số nhị phân hoặc mã hexa.*

- ❑ Hàm băm đóng vai trò then chốt
 - Toàn vẹn dữ liệu,
 - Xác minh nhanh chóng,
 - Gắn kết các khối với nhau theo chuỗi.



Đặc điểm của hàm băm

- **Tính một chiều:**

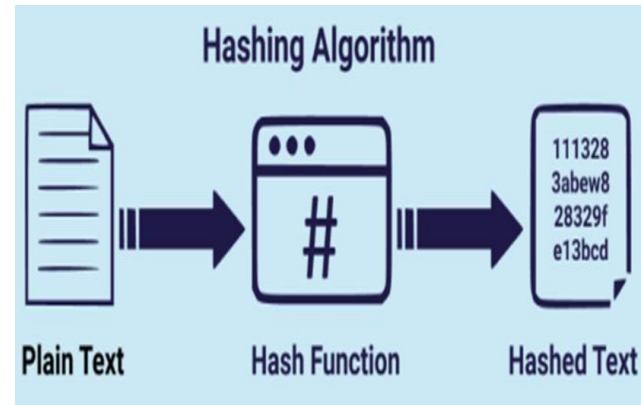
Hàm băm hoạt động theo hướng duy nhất: từ đầu vào sinh ra đầu ra. Tuy nhiên, từ đầu ra (giá trị băm) thì hoàn toàn không thể tính ngược lại để thu được dữ liệu gốc. Chính vì không thể đảo ngược nên hàm băm được sử dụng để xác minh dữ liệu chứ không dùng để mã hóa nội dung.

- **Khó xảy ra trùng lặp:**

Về mặt lý thuyết, rất hiếm khi xảy ra trường hợp hai dữ liệu đầu vào khác nhau nhưng lại tạo ra cùng một giá trị băm. Tính chất này giúp đảm bảo rằng mỗi giá trị băm là duy nhất cho mỗi tập dữ liệu cụ thể.

- **Tính xác định:**

Một đầu vào cụ thể sẽ luôn cho ra một giá trị băm duy nhất. Nếu ta áp dụng cùng một hàm băm cho cùng một dữ liệu đầu vào, kết quả sẽ luôn giống nhau. Đây là tính chất cần thiết để đối chiếu dữ liệu và phát hiện sai lệch.



- **Hiệu suất tính toán cao:**

Việc tính toán giá trị băm được thực hiện rất nhanh, phù hợp với các hệ thống xử lý dữ liệu thời gian thực. Chống va chạm (collision resistance):

- **Tính nhạy cảm cao với đầu vào (Avalanche Effect):**

Chỉ cần thay đổi một ký tự rất nhỏ trong đầu vào, giá trị băm sinh ra sẽ thay đổi hoàn toàn, không còn giống với kết quả trước đó. Tính chất này giúp phát hiện ngay lập tức bất kỳ thay đổi nào trong dữ liệu, dù là nhỏ nhất.

Ví dụ 4.1: So sánh giá trị băm khi thay đổi dữ liệu đầu vào.

```
1  import hashlib
2
3  # Hai chuỗi dữ liệu khác nhau chỉ một ký tự
4  data1 = "Blockchain là công nghệ nền tảng"
5  data2 = "blockchain là công nghệ nền tảng" # Chữ B viết thường
6
7  # Tính giá trị băm cho từng chuỗi
8  hash1 = hashlib.sha256(data1.encode()).hexdigest()
9  hash2 = hashlib.sha256(data2.encode()).hexdigest()
10
11 # In kết quả
12 print("Băm của chuỗi 1:", hash1)
13 print("Băm của chuỗi 2:", hash2)
```

Băm của chuỗi 1: d4e2614569730c7287b186763fe43ca21d3bc8dcd148453b93eefce08433e8df
Băm của chuỗi 2: 0844f3d44370e6e8a2b9042553cb01753423ec4b45d33e29a71ab83d09e435a4

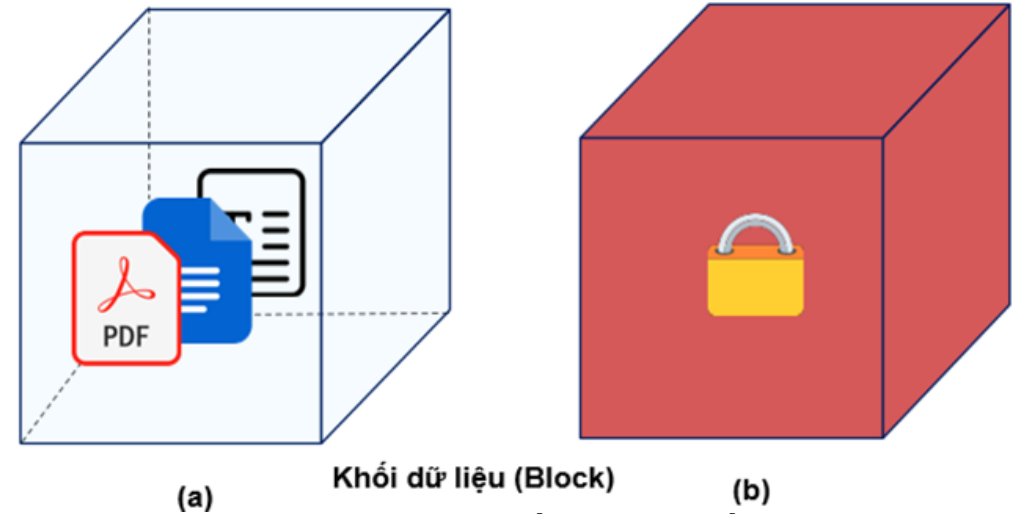
Nhận xét: *Ví dụ 4.1 thể hiện rõ tính “nhạy cảm với đầu vào” của hàm băm.*

4.2. CẤU TRÚC VÀ HOẠT ĐỘNG CỦA BLOCKCHAIN

4.2.1. Cấu trúc khối trong Blockchain

4.2.1.1. Khái niệm khối (Block)

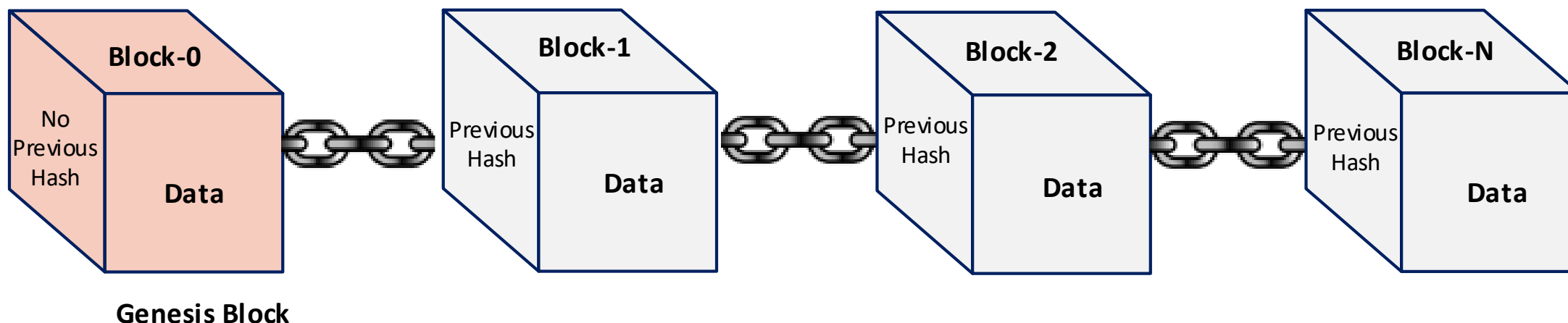
- ❑ Block (khối) là đơn vị cơ bản dùng để lưu trữ dữ liệu. Mỗi block giống như một chiếc "hộp dữ liệu" chứa các thông tin giao dịch đã được xác thực.



Hình 4.2: Từ dữ liệu giao dịch đến một khối được mã hóa.

- ❑ Tất cả các dữ liệu của chúng ta sẽ được đóng gói vào 1 khối, sau đó khối này sẽ được khóa lại bằng thuật toán mã hóa.
- ❑ Một khi đã được khóa, dữ liệu trong khối sẽ không thể bị thay đổi, tạo nên một bằng chứng bất di bất dịch, **tồn tại vĩnh viễn trong hệ thống**.
- ❑ Các block này liên kết với nhau thông qua mã băm (hash) của khối trước đó, tạo thành một chuỗi nối tiếp không thể tách rời – đó chính là blockchain.

4.2.1.1. Khái niệm khối (Block)



Hình 4.3: Cấu trúc chuỗi khối trong Blockchain bắt đầu từ Genesis Block

- Khối đầu tiên được tạo ra được gọi là **Genesis Block** (khối khởi nguyên).
- Khối này không chứa mã băm (hash) của khối trước vì đơn giản là... **không có khối trước nó**.
- Thay vào đó, giá trị hash của khối trước được gán mặc định (thường là toàn số 0).
- Genesis Block là nền móng ban đầu, được mọi nút mạng công nhận, và tất cả các khối sau đều nối tiếp từ đó.

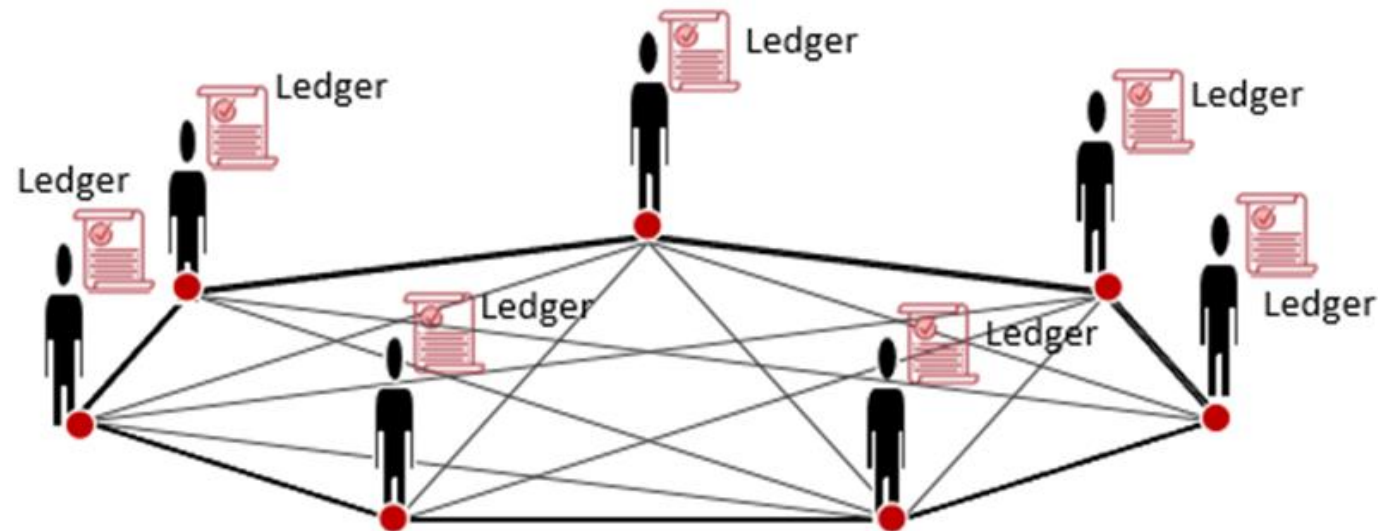
Ví dụ: Trong hệ thống Bitcoin, **Genesis Block** được tạo ra vào ngày **3/1/2009** bởi **Satoshi Nakamoto** và vẫn tồn tại nguyên vẹn đến ngày nay.

4.2.1.2. Sổ cái phân tán

a. Khái niệm:

Sổ cái phân tán là một hệ thống ghi chép mà trong đó dữ liệu được lưu trữ và **đồng bộ trên nhiều nút (nodes)** trong mạng máy tính. **Không có máy chủ trung tâm**, mỗi nút đều giữ một bản sao giống nhau của sổ cái.

Bất kỳ thay đổi nào – ví dụ như thêm một giao dịch – đều phải được xác minh và cập nhật đồng nhất trên tất cả các nút.

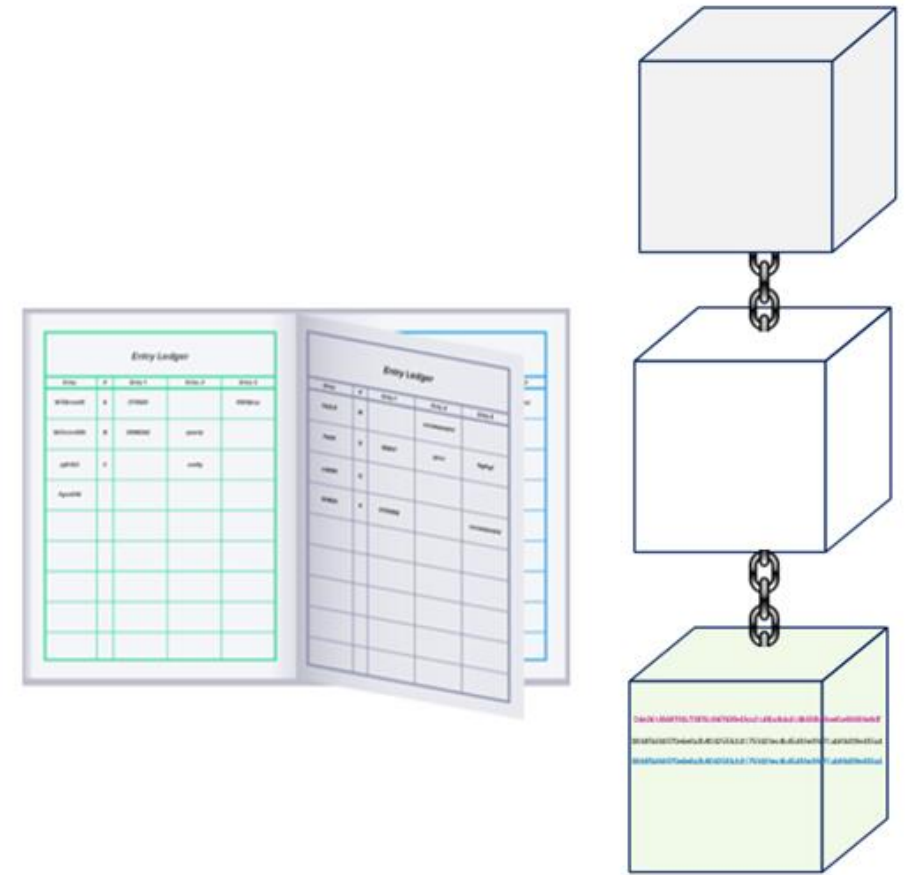


Hình 4.4: Minh họa sổ cái phân tán (Distributed Ledger)

Mỗi nút mạng (người dùng) đều giữ một bản sao của sổ cái (ledger-tờ giấy) và kết nối với các nút khác qua mạng ngang hàng (P2P). Dữ liệu được đồng bộ ngang hàng giữa các nút. Khi có giao dịch mới, mọi bản sao đều được cập nhật đồng bộ nếu đạt được sự thống nhất toàn mạng.

b. Từ sổ cái phân tán đến chuỗi khối

- ❑ Blockchain là một cách hiện thực cụ thể của sổ cái phân tán, trong đó dữ liệu không được lưu rải rác tự do, mà được tổ chức thành các khối (block).
- ❑ Mỗi khối chứa một nhóm các giao dịch đã xác thực, và được liên kết với khối trước đó thông qua mã băm (hash) (hash).
- ❑ Sự liên kết này tạo ra một chuỗi liên tục không thể thay đổi – chính là chuỗi khối (blockchain).



Hình 4.5: Chuyển đổi từ sổ cái truyền thống sang chuỗi khối (blockchain), *nguồn: Horizen Academy*

Mỗi block trong chuỗi blockchain có thể xem như một đơn vị lưu trữ tương đương một trang trong sổ cái. Trong khi sổ truyền thống lưu thông tin dạng bảng, blockchain mã hóa các thông tin giao dịch thành chuỗi giá trị băm (hash) và liên kết từng block bằng mã băm (hash) của khối trước đó

d) Vai trò của mạng ngang hàng (P2P) trong sổ cái phân tán

- Để duy trì được cấu trúc sổ cái phân tán như mô tả ở trên, Blockchain dựa hoàn toàn vào mô hình mạng ngang hàng (Peer-to-Peer – P2P).
- Trong mạng P2P, mọi nút đều bình đẳng: mỗi nút vừa lưu trữ bản sao sổ cái, vừa có quyền xác minh giao dịch, tham gia đồng thuận và phát sóng khối mới.
- Mạng P2P giúp Blockchain đảm bảo tính phi tập trung, vì không cần bất kỳ máy chủ trung tâm nào để điều phối.



- ✓ Tăng tính bảo mật (vì không có điểm tập trung dễ bị tấn công),
- ✓ nâng cao độ tin cậy (vì mạng vẫn hoạt động ngay cả khi một phần nút bị ngắt kết nối).

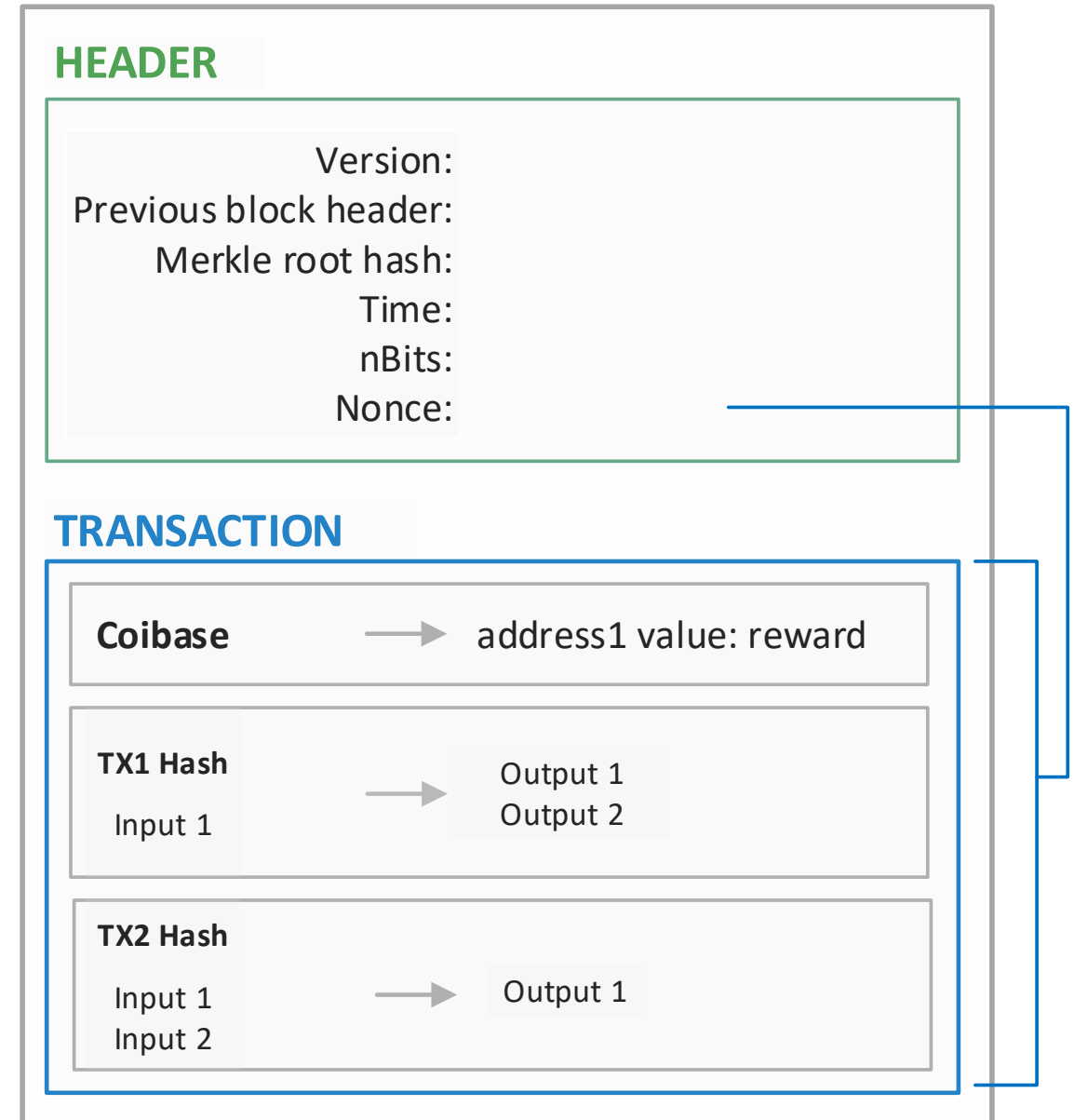
Mô hình sổ cái phân tán trên nền mạng P2P trở thành nền tảng cốt lõi của Blockchain – vừa đảm bảo minh bạch, vừa phòng chống gian lận hiệu quả.

4.2.1.3. Cấu trúc khối trong Blockchain

a) Cấu trúc

Mỗi khối là một cấu trúc dữ liệu dạng container dùng để tổng hợp các giao dịch và ghi vào sổ cái công khai, tức là blockchain.

- Phần tiêu đề khối (Block header): chứa các siêu dữ liệu (metadata), như mã băm (hash) của khối trước, thời gian tạo khối, cây Merkle, nonce,...
- Danh sách giao dịch (Transaction list): là phần chứa các giao dịch đã xác thực, chiếm phần lớn dung lượng của khối.



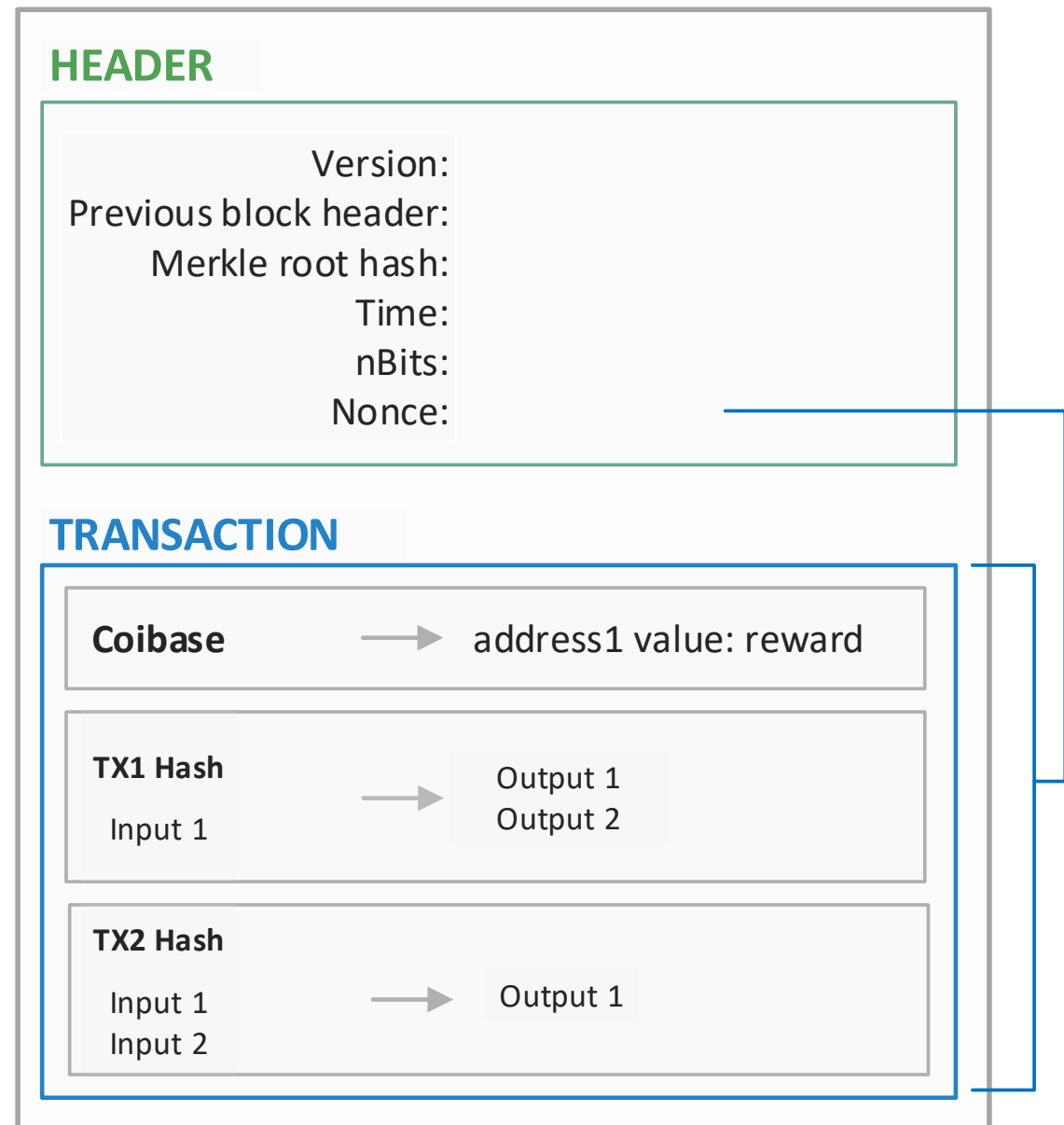
4.2.1.3. Cấu trúc khối trong Blockchain, tiếp...

a) Cấu trúc

Phần Header chứa thông tin siêu dữ liệu (version, previous hash, Merkle root,...).

Phần danh sách các giao dịch Transaction với giao dịch đầu tiên thường là một giao dịch hệ thống (System TX) dùng để phân bổ **phần thưởng (reward)**,

address1 trong hình đại diện cho địa chỉ đích nhận phần thưởng trong block. Đây là một ví dụ minh họa – có thể hiểu là **địa chỉ ví của recipient** (người nhận).



4.2.1.3. Cấu trúc khối trong Blockchain, tiếp...

a) Cấu trúc

- Phần tiêu đề của khối có kích thước 80 bytes, trong khi một giao dịch trung bình có kích thước ít nhất là 250 bytes và một khối trung bình chứa hơn 500 giao dịch.
- Vì vậy, một khối hoàn chỉnh với tất cả các giao dịch sẽ có kích thước lớn hơn tiêu đề của khối khoảng 1,000 lần.

Bảng 4.1: Các trường trong khối Blockchain

Kích thước (Bytes)	Trường	Mô tả
4 bytes	Block Size	Kích thước của toàn bộ khối, tính bằng byte, bao gồm cả phần dữ liệu sau đó.
80 bytes	Block Header	Bao gồm nhiều trường con như phiên bản (version), mã băm (hash) của khối trước (previous block hash), và nonce.
1–9 bytes (VarInt)	Transaction Counter	Số lượng giao dịch có trong khối, sử dụng kiểu dữ liệu biến đổi để tối ưu dung lượng lưu trữ.
Variable	Transactions	Danh sách các giao dịch đã được xác minh và lưu lại trong khối. Kích thước phần này thay đổi tùy thuộc vào số lượng và loại giao dịch.

- Trường Block Size (4 bytes): Kích thước của khối

Ý nghĩa:

- Xác định kích thước của toàn bộ khối, bao gồm cả phần Block Header và Transactions.
- Giúp các nút trong mạng xác định được dung lượng dữ liệu cần xử lý khi truyền tải và lưu trữ.
- Thông số cụ thể: trường kích thước khối sử dụng 4 bytes để lưu trữ, cho phép biểu diễn kích thước tối đa khoảng 4 GB.

Ví dụ: Nếu Block Size = 1,000,000 bytes, nghĩa là khối này có kích thước 1 MB.

- Block Header (80 bytes): Tiêu đề của khối

Ý nghĩa:

- Chứa các thông tin quan trọng để liên kết khối hiện tại với khối trước đó, đảm bảo tính toàn vẹn của chuỗi khối (blockchain).
- Block header bao gồm nhiều trường con sẽ được phân tích cụ thể ở mục c.

4.2.1.3. Cấu trúc khối trong Blockchain, tiếp...

- Transaction Counter (1–9 bytes): Bộ đếm giao dịch

Ý nghĩa:

- Cho biết số lượng giao dịch được ghi nhận trong khối.
- Sử dụng **VarInt** (Integer biến đổi) để tối ưu hóa kích thước dữ liệu, chỉ dùng nhiều byte khi có quá nhiều giao dịch.

Ví dụ: Bộ đếm giao dịch sẽ có kích thước:

- 1 byte: Nếu có ít hơn 255 giao dịch.
- 3 bytes: Nếu có hàng ngàn giao dịch trong khối.

4.2.1.3. Cấu trúc khối trong Blockchain, tiếp...

- Transactions (Variable): Các giao dịch trong khối

Ý nghĩa:

- Là phần dữ liệu chiếm dung lượng lớn nhất của khối, bao gồm danh sách các giao dịch đã được xác nhận và mã hóa.
- Mỗi giao dịch có cấu trúc riêng, bao gồm:
 - Input: Địa chỉ gửi và chữ ký số xác nhận.
 - Output: Địa chỉ nhận và số lượng tài sản được chuyển.
 - Script: Điều kiện để giao dịch hợp lệ.
 - Kích thước biến đổi (Variable): Tùy thuộc vào số lượng và độ phức tạp của giao dịch.
Ví dụ: Khối chứa 100 giao dịch sẽ có kích thước lớn hơn khối chỉ chứa 10 giao dịch.

b) Khái niệm Merkle Tree (Cây băm)

□ Merkle Tree là cấu trúc dữ liệu dạng cây nhị phân, trong đó:

- Các nút lá là giá trị băm của từng giao dịch.
- Các nút cha là giá trị băm của sự kết hợp hai nút con.
- Đỉnh của cây là Merkle Root – được lưu vào Block Header.

Khái niệm này được đặt theo tên Ralph Merkle – người đã đăng ký bằng sáng chế vào năm 1979.

b) Khái niệm Merkle Tree (Cây băm), ...

Cách hoạt động:

Quá trình xây dựng cây Merkle diễn ra theo các bước:

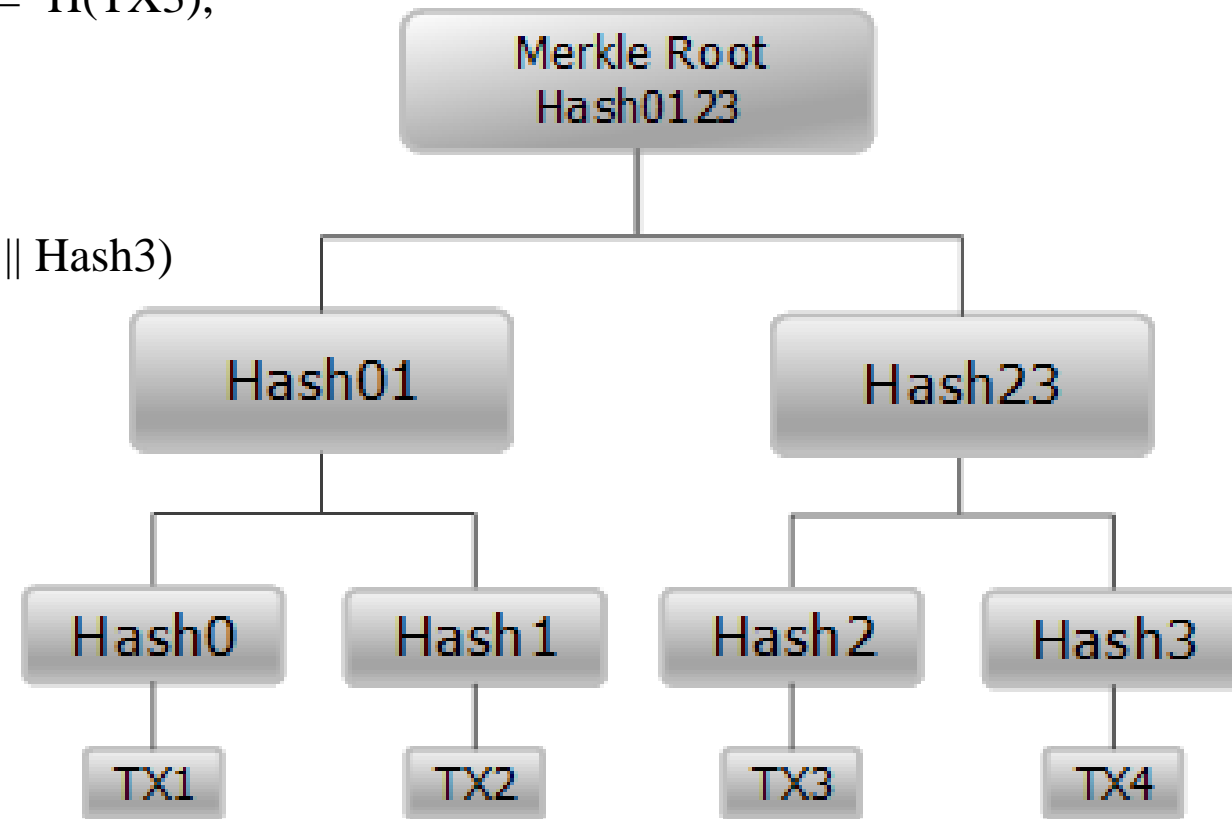
1. Băm từng giao dịch (TX1, TX2, ...) bằng hàm băm SHA256 để tạo thành các nút lá.
2. Ghép từng cặp nút lá, nối hai giá trị băm lại với nhau, rồi băm tiếp → tạo ra các nút ở tầng tiếp theo.
3. Lặp lại quá trình này, tiếp tục ghép cặp và băm, cho đến khi chỉ còn một nút duy nhất ở đỉnh cây → đó là Merkle Root.

Nếu số lượng giao dịch là số lẻ, giao dịch cuối cùng sẽ được nhân đôi chính nó để đảm bảo số lượng cặp là chẵn.

b) *Khái niệm Merkle Tree (Cây băm), ...*

Ví dụ : Với 4 giao dịch: TX1, TX2, TX3, TX4

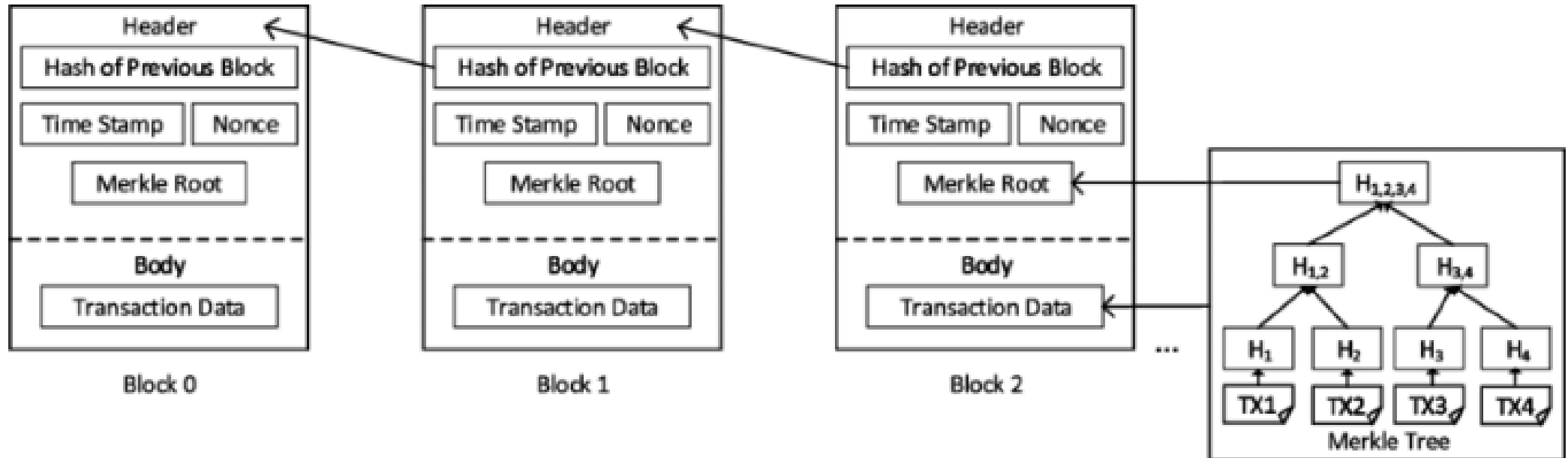
- Bước 1: Tạo các hash gốc
→ $\text{Hash0} = H(\text{TX1})$, $\text{Hash1} = H(\text{TX2})$, $\text{Hash2} = H(\text{TX3})$,
 $\text{Hash3} = H(\text{TX4})$
- Bước 2: Ghép đôi → băm tiếp
→ $\text{Hash01} = H(\text{Hash0} \parallel \text{Hash1})$, $\text{Hash23} = H(\text{Hash2} \parallel \text{Hash3})$
- Bước 3: Tạo Merkle Root
→ $\text{Merkle Root} = H(\text{Hash01} \parallel \text{Hash23})$



Hình 4.7: Minh họa cấu trúc Merkle Tree đầy đủ với 4 giao dịch (TX1 → TX4)

Vai trò cây Merkle trong Blockchain:

- Xác minh tính toàn vẹn của tập hợp giao dịch mà không cần tải toàn bộ.
- Phát hiện thay đổi dữ liệu nhanh chóng – vì chỉ cần một giao dịch thay đổi → Merkle Root sẽ thay đổi.
- Tiết kiệm không gian và tăng hiệu suất xác minh trong hệ thống phân tán.



c) Thành phần Block Header

Block Header là vùng chứa thông tin quan trọng cốt lõi dùng để xác định danh tính của khối, liên kết với khối trước đó và phục vụ quá trình xác minh toàn vẹn chuỗi khối.

Tổng dung lượng của Block Header là 80 bytes, gồm 6 trường chính được mô tả trong bảng 4.2

Bảng 4.2: Các trường của Block Header

Kích thước	Trường	Mô tả
4 bytes	Phiên bản (Version)	Số phiên bản để theo dõi các nâng cấp của phần mềm/giao thức.
32 bytes	Mã băm (hash) của khối trước (Previous Block Hash)	Mã băm (hash) của khối trước trong chuỗi, tạo sự liên kết giữa các khối.
32 bytes	Gốc cây Merkle (Merkle Root)	Mã băm (hash) của gốc cây Merkle, tóm tắt tất cả các giao dịch trong khối này.
4 bytes	Dấu thời gian (Timestamp)	Thời gian tạo khối (tính bằng giây kể từ Unix Epoch).
4 bytes	Mục tiêu độ khó (Difficulty Target)	Mức độ khó của thuật toán Proof-of-Work cho khối này.
4 bytes	Nonce	Số ngẫu nhiên dùng trong thuật toán Proof-of-Work.

Phân tích các trường trong bảng 4.2:

- *Version (4 bytes)*: Xác định phiên bản của giao thức Blockchain và các thay đổi trong cấu trúc khối hoặc thuật toán. Đảm bảo các nút (nodes) sử dụng cùng một phiên bản có thể hiểu và xử lý khối đúng cách.

Ví dụ: Nếu Version = 2, có thể là dấu hiệu của một bản cập nhật để hỗ trợ tính năng mới như SegWit (Segregated Witness) trong Bitcoin.

- *Previous Hash (32 bytes)*: Việc liên kết giữa các khối được thực hiện thông qua trường Previous Block Hash. **Mỗi khối chứa mã băm (hash) của khối liền trước**, giúp liên kết các khối thành chuỗi liên tục và đảm bảo tính toàn vẹn và bất biến. *Nếu dữ liệu của một khối bị thay đổi, mã băm (hash) của khối đó cũng thay đổi dẫn đến sự mất đồng bộ với các khối sau và làm vô hiệu hóa toàn bộ các khối sau nó* – một đặc tính quan trọng đảm bảo tính toàn vẹn và chống giả mạo của blockchain.

Ví dụ: Mã băm (hash): **000000000000000000000000a7b32c6c23f...** Nếu bị thay đổi: Các khối sau sẽ bị vô hiệu hóa.

Phân tích các trường trong bảng 4.2:

- *Merkle Root (32 bytes)*: Là **gốc của cây Merkle** (Merkle Tree) – một cấu trúc dữ liệu dạng cây nhị phân, được tạo ra bằng cách băm toàn bộ các giao dịch trong khối theo từng cặp, sau đó tiếp tục băm các kết quả cho đến khi chỉ còn một giá trị duy nhất ở đỉnh cây: đó là Merkle Root. Trường này đóng vai trò như dấu vân tay đại diện duy nhất cho toàn bộ các giao dịch trong khối.

Merkle Root giúp xác minh tính toàn vẹn của dữ liệu giao dịch một cách hiệu quả, mà không cần phải tải toàn bộ khối. Bất kỳ thay đổi nhỏ nào trong một giao dịch cũng sẽ làm thay đổi toàn bộ Merkle Root, giúp phát hiện giả mạo dễ dàng..

Ví dụ: Mã băm (hash) Merkle Root: 3e23e8160039594a33894f6564e1b13....

Phân tích các trường trong bảng 4.2:

- *Timestamp (4 bytes)*: Ghi lại thời điểm khối được tạo ra dưới dạng giây kể từ Unix Epoch (00:00:00 UTC ngày 1/1/1970). Trường này giúp sắp xếp thứ tự các giao dịch một cách chính xác. Timestamp cũng là căn cứ để tính toán độ khó (Difficulty Target) và thời gian trung bình giữa các khối.

Ví dụ: Timestamp = 1635425800 tương ứng với 02/11/2021 10:30:00 UTC.

- *Difficulty Target (4 bytes)*:
Biểu thị mục tiêu độ khó của thuật toán đồng thuận. Trường này yêu cầu mã băm (hash) đầu ra của khối phải nhỏ hơn một ngưỡng xác định (mục tiêu), từ đó điều chỉnh mức độ cạnh tranh trong quá trình đào (mining).

Phân tích các trường trong bảng 4.2:

- *Nonce (**N**umber **O**nly **U**sed **O**nce - Số chỉ dùng một lần) (4 bytes):*

Là một giá trị số được thay đổi liên tục trong quá trình tìm kiếm một mã băm (hash) thỏa mãn điều kiện của trường Difficulty Target trong cơ chế đồng thuận (ví dụ như Proof of Work – PoW). Trường này cho phép lặp lại quá trình băm nhiều lần với các giá trị khác nhau cho đến khi tìm ra một kết quả hợp lệ.

Nếu không có Nonce, quá trình thử sai sẽ không thể thực hiện vì không có yếu tố nào thay đổi đầu vào để tạo ra mã băm (hash) mới. Chính ở đây, hoạt động “đào Bitcoin” – hay còn gọi là mining – diễn ra: các thợ đào sẽ thử liên tục hàng triệu giá trị Nonce khác nhau để tìm ra mã băm (hash) phù hợp, từ đó tạo ra khối mới.

Ví dụ: Nonce = 2083236893 là giá trị ngẫu nhiên trong một khối Bitcoin.

d) Thành phần Transaction

Transaction là thành phần chính chiếm phần lớn dung lượng trong mỗi khối (block). Mỗi Transaction (giao dịch) là đơn vị dữ liệu mô tả một hành động được ghi nhận trong blockchain.

Tùy vào hệ thống, hành động này có thể là **chuyển giá trị, cập nhật trạng thái, thực hiện hợp đồng thông minh, hoặc xác minh dữ liệu.**

Mỗi khối (block) chứa một danh sách các transaction đã được xác thực từ người gửi đến người nhận, kèm theo các thông tin xác thực và điều kiện đi kèm.

Trước danh sách giao dịch, khối sẽ ghi lại một trường gọi là **Transaction Counter** – cho biết tổng số giao dịch trong khối. Trường này sử dụng định dạng VarInt (kiểu số nguyên biến đổi) để tiết kiệm không gian lưu trữ:

- Nếu số lượng giao dịch < 253 → dùng 1 byte.
- Nếu từ 253 đến 65.535 → dùng 3 byte.
- Nếu > 65.535 → dùng 5 hoặc 9 byte.

d) Thành phần Transaction

- *Cấu trúc một giao dịch*: mỗi giao dịch thường gồm 3 phần chính:

1. Input (Đầu vào)

- Chỉ định(tham chiếu) đến một giao dịch trước đó mà người gửi đang sử dụng.
- Bao gồm:
 - Mã băm (hash) (mã nhận diện) của giao dịch gốc và chỉ số (index) của output được sử dụng.
 - Bảng chứng xác thực (ví dụ: chữ ký số) của người gửi để xác thực quyền sở hữu.

2. Output (Đầu ra)

- Chỉ định đích đến hay địa chỉ người nhận và giá trị sẽ được gửi.
- Bao gồm:
 - Địa chỉ đích (thường là khóa công khai)
 - Giá trị hoặc dữ liệu được chuyển/gán (ví dụ: số lượng token, hoặc tài sản kỹ thuật số).
 - Điều kiện sử dụng (được xác định qua script)

d) Thành phần Transaction

3. Script (Tập lệnh điều kiện)

- Là một đoạn mã ngắn định nghĩa điều kiện để giao dịch hợp lệ, chẳng hạn như yêu cầu chữ ký hợp lệ từ người nhận.
- Thường có 2 loại:
 - ScripSig: nằm trong input – cung cấp bằng chứng để mở khóa giá trị.
 - ScripPubkey: nằm trong output – quy định điều kiện để giá trị đó có thể được sử dụng trong tương lai.
- *Đặc điểm của giao dịch (Transaction)*
 - Kích thước giao dịch không cố định – tùy thuộc vào số lượng input, output và độ dài (phức tạp) của script. Trong thực tế, phần này có thể chiếm đến 90% dung lượng của một block.
 - Danh sách các giao dịch trong một khối chính là dữ liệu đầu vào để tạo nên cây Merkle – từ đó tính ra Merkle Root.

Ví dụ: Giả sử trong một khối có 3 giao dịch

TX1: User A → User B: 50 tokens

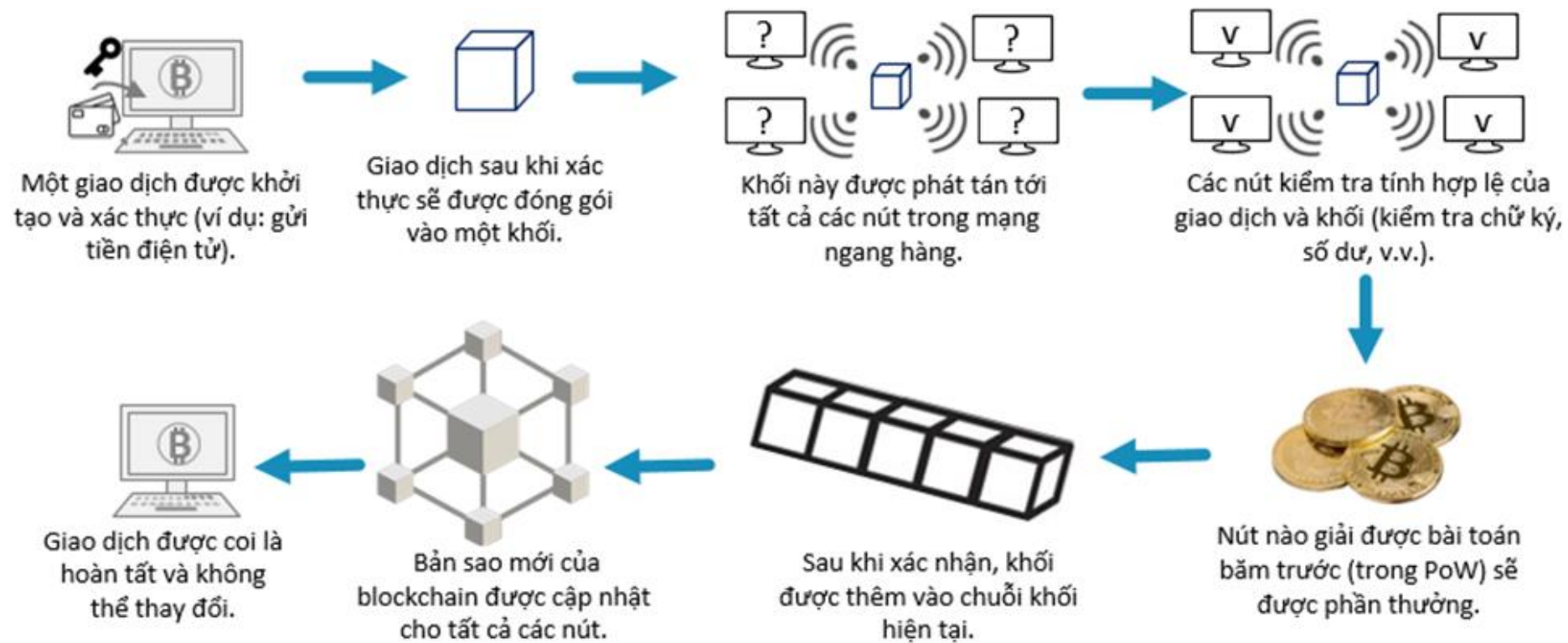
TX2: User C → User D: 20 tokens

TX3: User E → User F: ghi nhận quyền sở hữu tài sản kỹ thuật số.

→ Mỗi giao dịch sẽ gồm đầy đủ thông tin về input, output, và script xác thực.

Các giao dịch này được tổ chức thành danh sách trong khối, sau đó dùng để tạo Merkle Tree và sinh ra Merkle Root ghi trong Block Header.

[Token trong Blockchain](#) là **một đơn vị giá trị kỹ thuật số** được tạo và quản lý trên một chuỗi khối (blockchain). Nó có thể đại diện cho tài sản, quyền truy cập, điểm thưởng, hoặc giá trị tiền tệ trong hệ sinh thái của chuỗi đó.



Hình 4.8: Quy trình xử lý giao dịch trong blockchain – từ khởi tạo đến cập nhật sổ cái phân tán

Ví dụ 4.2. Minh họa cấu trúc khối bằng Python: Dưới đây là một đoạn mã đơn giản tạo một khối và tính toán mã băm (hash) của khối đó:

```
1 import hashlib # Thư viện tích hợp sẵn trong Python
2               # để sử dụng các thuật toán băm như SHA-256
3
4 class Block:
5     def __init__(self, previous_hash, data):
6         # Lưu mã băm của khối trước (liên kết chuỗi)
7         self.previous_hash = previous_hash
8         # Dữ liệu giao dịch hoặc thông tin cần ghi vào khối
9         self.data = data
10        # Tự động tính mã băm khi khối được tạo
11        self.hash = self.calculate_hash()
12
13    def calculate_hash(self):
14        # Gộp thông tin khối hiện tại và khối trước
15        block_data = self.previous_hash + self.data
16        # Tính mã băm SHA-256 (đầu ra là chuỗi 64 ký tự hexa)
17        return hashlib.sha256(block_data.encode()).hexdigest()
18
19 # Tạo một khối mới với dữ liệu giả định
20 block = Block("0000a7c8...", "Transaction Data Example")
21
22 # In ra mã băm của khối
23 print("Mã băm (hash) của khối:", block.hash)
```

Kết quả thực hiện chương trình:

Mã băm (hash) của khối:

e60f21790925a680f4b3219513eb10015295075aa58fdadfa6e0572eacc87c28

Đoạn mã trong ví dụ 4.2. trên giúp chúng ta thấy rõ:

- ❑ Cấu trúc một khối gồm: **dữ liệu, mã băm khối trước, mã băm hiện tại**
- ❑ Cách mã băm được tính toán dựa trên nội dung khối:
→ thay đổi dữ liệu → hash thay đổi
- ❑ Tính liên kết giữa các khối được xây dựng bằng cách dùng **previous_hash**

Nguyên lý cốt lõi của Blockchain:

“Khối mới không thể tự tồn tại – nó phải gắn mã băm của khối trước để đảm bảo tính toàn vẹn và chuỗi liên kết không thể bị phá vỡ.”

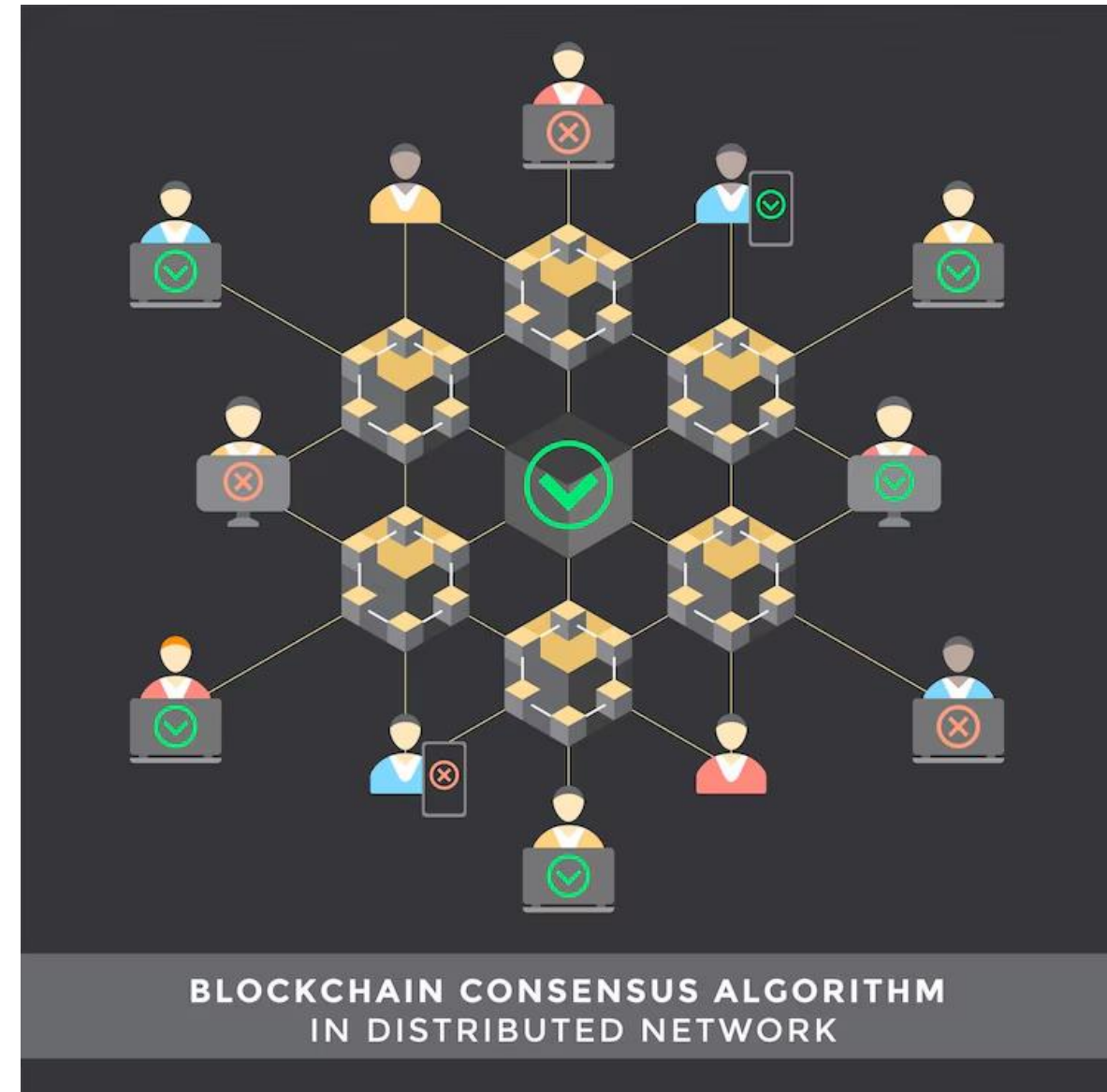
Trong ví dụ 4.2. Thay block.data = "Something else" và in lại block.hash → ?

4.2.2. Giao thức đồng thuận trong Blockchain

- ❑ Giao thức đồng thuận (**Consensus Protocol**) là cơ chế giúp các nút trong mạng Blockchain đạt được sự thống nhất về trạng thái của sổ cái phân tán mà không cần sự kiểm soát của một thực thể trung tâm.
- ❑ Các giao thức đồng thuận đảm bảo rằng chỉ có các giao dịch hợp lệ mới được ghi nhận vào chuỗi khối.

Một giao thức đồng thuận hiệu quả cần đáp ứng các tiêu chí:

- ❑ **Khả năng chịu lỗi** (Fault Tolerance): Hệ thống vẫn hoạt động đúng dù một số node gặp sự cố hoặc gửi thông tin sai.
- ❑ **Đồng thuận chung** (Agreement): Nếu một node chấp nhận một giá trị A, thì tất cả các node còn lại cuối cùng cũng sẽ đồng thuận với giá trị đó.
- ❑ **Tính toàn vẹn** (Integrity): Các node chỉ chấp nhận giá trị đúng; không có node nào chấp nhận dữ liệu giả mạo hoặc chưa được đề xuất hợp lệ.



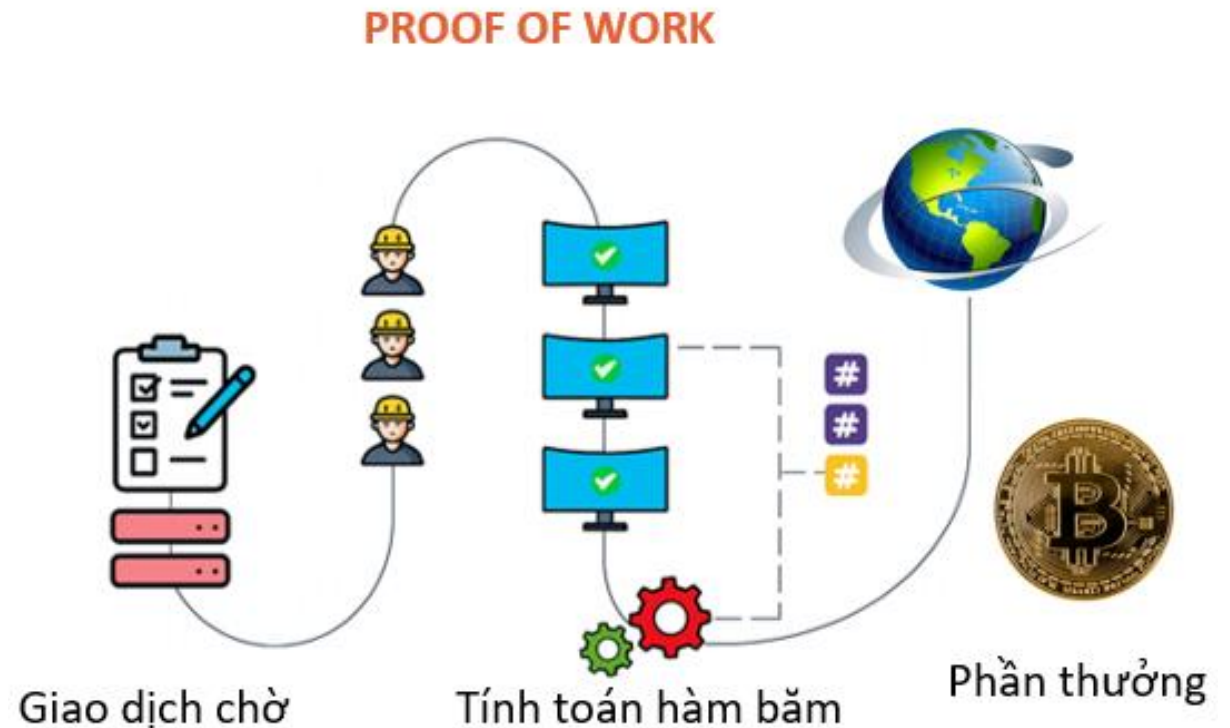
4.2.2.1. Giao thức Proof of Work (PoW) - Bằng chứng công việc:

a) Định nghĩa

- ❑ Proof of Work (PoW) là một cơ chế đồng thuận yêu cầu các nút trong mạng (thợ đào - miners) thực hiện các phép toán tính toán phức tạp để tìm ra giá trị Nonce hợp lệ, từ đó tạo ra một mã băm (hash) (hash) đáp ứng yêu cầu của hệ thống.
- ❑ Nút nào tìm ra kết quả trước sẽ được quyền ghi khối mới vào Blockchain và nhận phần thưởng (thường là tiền điện tử).

b) Cơ chế hoạt động:

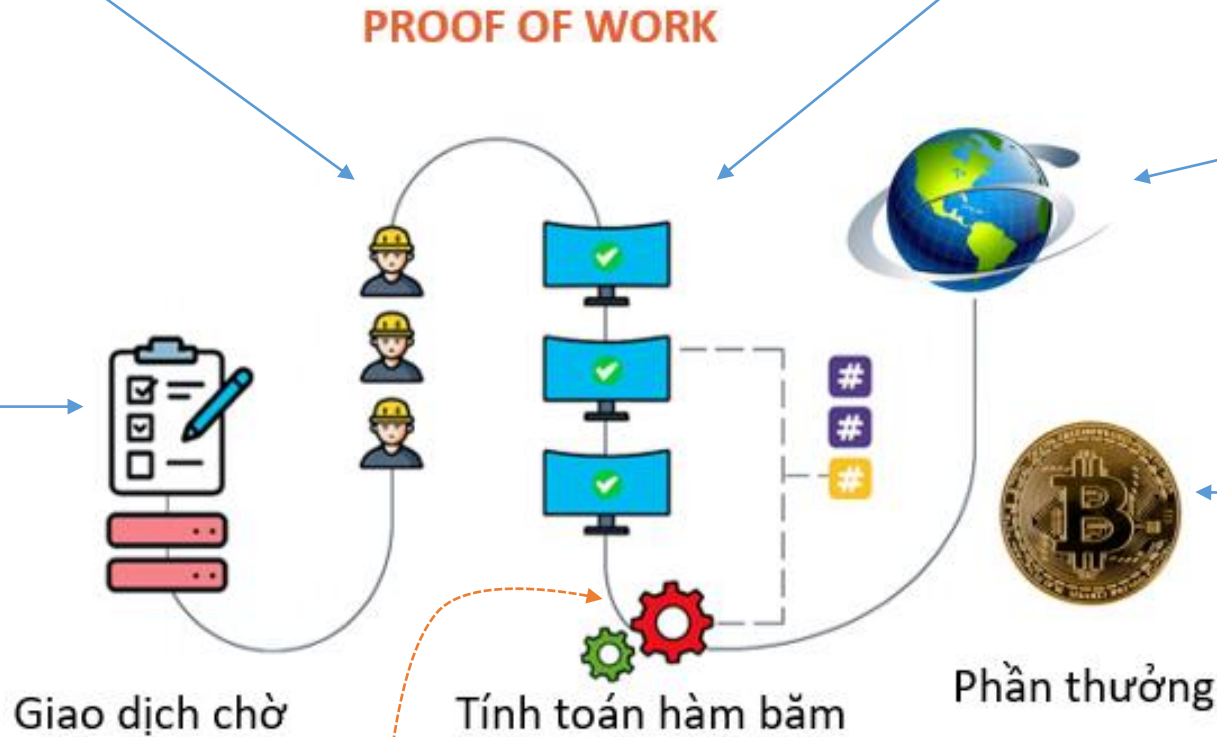
1. Hệ thống tổng hợp các giao dịch chờ vào một block mới.
2. Các **miner** cạnh tranh giải bài toán hash bằng cách thử Nonce.
3. Ai tìm được hash hợp lệ sớm nhất → broadcast kết quả.
4. Các node khác kiểm tra → nếu hợp lệ → block được ghi vào chuỗi.
5. Miner chiến thắng nhận phần thưởng (Bitcoin).



Phân tích từng bước sơ đồ PoW

2. Các thợ đào (miners) – tham gia giải bài toán mật mã.

1. Danh sách các giao dịch chờ xác nhận – tạo thành một khối (block) tiềm năng



3. Quá trình tính toán Hash, thử nhiều giá trị Nonce để tìm ra một Hash hợp lệ (đạt yêu cầu số lượng bit 0 ở đầu chuỗi)

6. Sau khi tìm được kết quả đúng, miner phát broadcast block đến toàn mạng để kiểm tra và xác nhận

7. Khi block được xác nhận → miner thắng sẽ **nhận phần thưởng khối (block reward)** + phí giao dịch

5. Thẻ #

Thể hiện cho kết quả băm (hash) sau mỗi lần thử

4. Biểu tượng cho quá trình “đào” (mining) cần nhiều tài nguyên tính toán

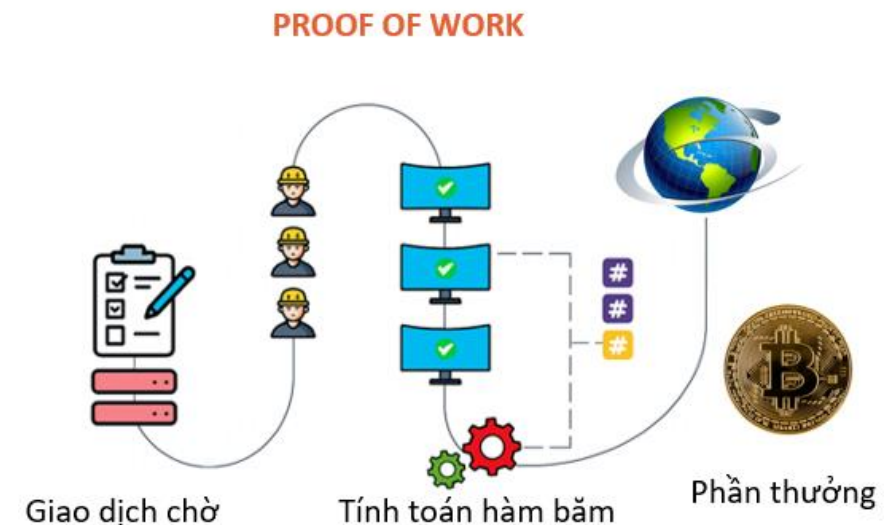
Ưu điểm:

- Bảo mật cao, chống lại các cuộc tấn công Sybil Attack và các hình thức gian lận khác.
- Không cần sự tin tưởng giữa các nút nhờ vào tính phi tập trung.

Nhược điểm:

- Tiêu tốn năng lượng lớn và tài nguyên tính toán khổng lồ.
- Tốc độ xử lý giao dịch chậm.

Ví dụ : Bitcoin và Ethereum (trước khi chuyển sang PoS).



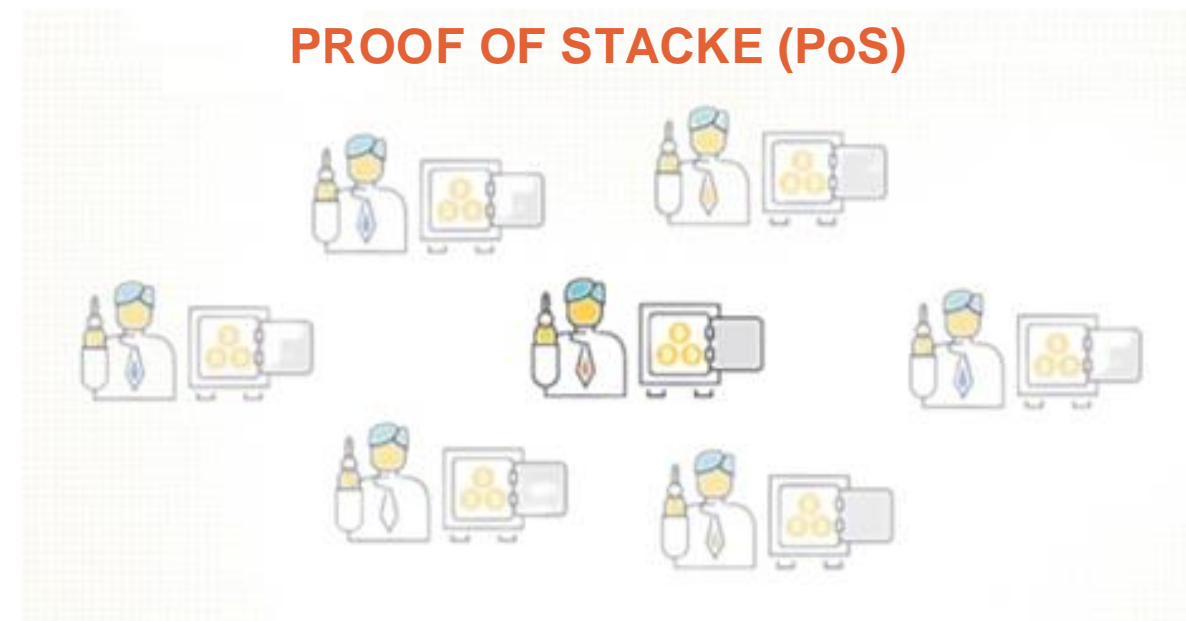
b) Cơ chế hoạt động:

- 1. Tập hợp giao dịch hợp lệ:* Các giao dịch mới được gửi đến mạng sẽ được các nút xác minh và đưa vào vùng nhớ tạm thời (**mempool**).
- 2. Tạo khối ứng viên:* thợ mỏ (**miner**) chọn một tập hợp giao dịch từ mempool để đóng gói thành một khối mới.
- 3. Tính toán mã băm (hash) hợp lệ:* Miner thử nhiều giá trị Nonce khác nhau để tìm ra một mã băm (hash) bắt đầu bằng một số lượng nhất định các số 0, đúng theo độ khó hiện hành.
- 4. Gửi khối ra toàn mạng:* Khi tìm được mã băm (hash) hợp lệ, miner sẽ phát sóng khối tới các nút khác.
- 5. Xác minh và thêm vào chuỗi:* Các nút kiểm tra tính hợp lệ của khối mới. Nếu hợp lệ, khối sẽ được thêm vào chuỗi và miner nhận phần thưởng (**coin mới sinh ra + phí giao dịch**).

4.2.2.2. Giao thức Proof of Stake (PoS) – Bằng chứng cổ phần

a) Định nghĩa

- ❑ **P**roof of **S**take (PoS) là cơ chế đồng thuận lựa chọn người xác thực (**validators**) dựa trên số lượng tài sản (coin) họ nắm giữ và thời gian nắm giữ.
- ❑ Các validators không cần thực hiện tính toán phức tạp như PoW mà thay vào đó, họ ký xác nhận giao dịch và được thưởng dựa trên số coin của họ.



PoS – cơ chế xác thực dựa trên cổ phần nắm giữ

Stake' trong giao thức Bằng chứng cổ phần (PoS)

- ❑ Trong cơ chế **PoS**, "**stake**" là số lượng coin mà người dùng cam kết khóa lại để tham gia xác thực Block
- ❑ Người stake càng nhiều thì cơ hội được chọn làm người xác thực càng cao.

Vai trò của Stake:

- Cam kết tài chính: giúp đảm bảo người tham gia không gian lận.
- Căn cứ lựa chọn người xác thực: stake càng lớn -> càng dễ được chọn.
- Tăng độ an toàn của mạng lưới: vì tấn công sẽ trở nên tốn kém hơn



- **Stake**: Số coin đặt cược/cam kết tài sản
- **Staking**: Hoạt động cam kết tài sản - Hành động khóa coin để tham gia xác thực
- **Validator**: Người xác thực
- **Slashing**: Slashing là cơ chế phạt (cắt coin) khi validator có hành vi gian lận hoặc sai lệch trong quá trình xác thực block

b) Cơ chế hoạt động

- ❑ Validators sẽ khóa một lượng coin nhất định làm tài sản đảm bảo.
- ❑ Nếu họ xác thực các giao dịch không hợp lệ, số coin này sẽ bị mất một phần ().
- ❑ Cơ chế này khuyến khích các validators hành xử trung thực.

- Lựa chọn người xác thực dựa trên số lượng tài sản (coin) họ nắm giữ và **thời gian nắm giữ**.
- Người xác thực sẽ ký các giao dịch và nhận phần thưởng tương ứng.

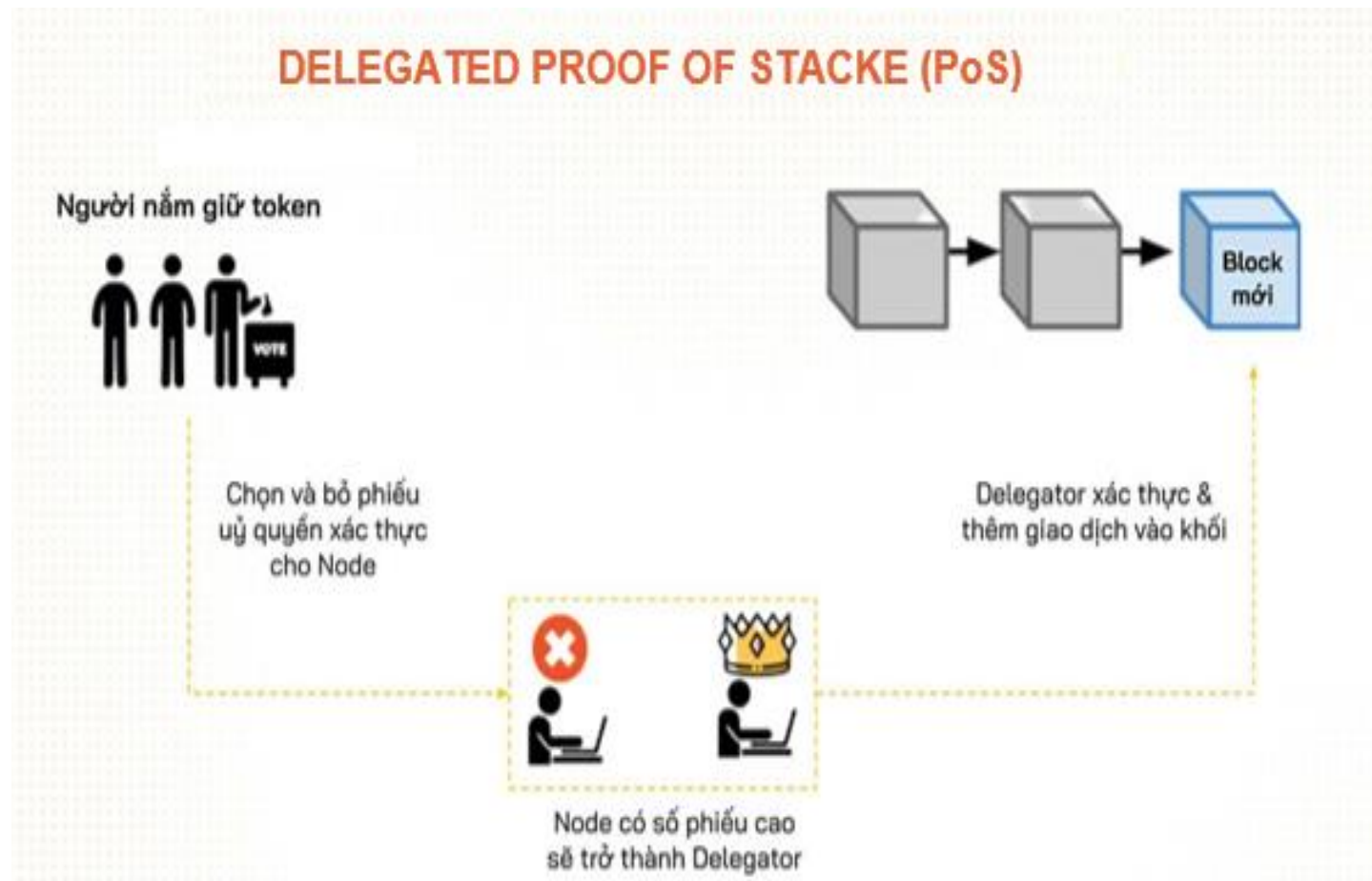


Slashing: cơ chế phạt (cắt coin) khi giao dịch gian lận.

4.2.2.3. Delegated Proof of Stake (DPoS) - Bằng chứng cổ phần được ủy quyền

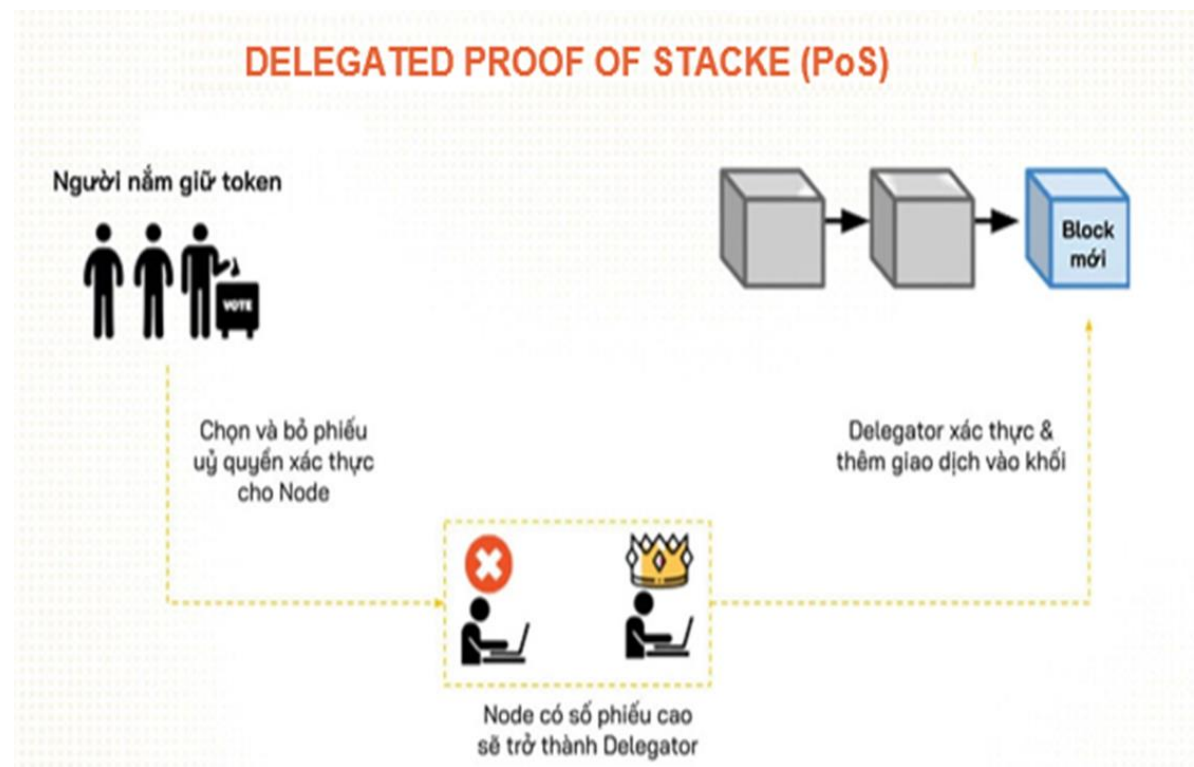
a) Định nghĩa:

- ❑ Delegated Proof of Stake (DPoS) là phiên bản mở rộng của PoS, trong đó các chủ sở hữu coin bỏ phiếu để chọn ra một **nhóm nhỏ các đại biểu** (delegates) xác thực giao dịch và tạo khối.
- ❑ Các đại biểu này thay mặt toàn bộ mạng lưới xác thực giao dịch



b) Cơ chế hoạt động:

- ❑ Các chủ sở hữu coin bỏ phiếu để chọn ra các đại biểu xác thực giao dịch thay vì toàn bộ các nút tham gia như PoS.
- ❑ Các đại biểu này thay phiên nhau xác thực giao dịch và tạo khối.
- ❑ Nếu đại biểu nào hành xử không đúng, họ có thể bị loại bỏ thông qua bỏ phiếu.



Ưu điểm:

- Tốc độ xác thực nhanh, phù hợp với các ứng dụng quy mô lớn.
- Cơ chế bỏ phiếu giúp mạng lưới duy trì tính dân chủ.

Nhược điểm:

- Có nguy cơ tập trung quyền lực vào các đại biểu được bầu chọn nếu các đại biểu thông đồng với nhau.
- Phụ thuộc nhiều vào cơ chế bỏ phiếu và tính minh bạch của nó.

Ví dụ thực tế: **EOS, Tron.**

4.2.3.4. Practical Byzantine Fault Tolerance (PBFT) - Chống lỗi Byzantine hiệu quả

Bài toán các vị tướng Byzantine

Hãy tưởng tượng một đội quân gồm nhiều tướng lĩnh đang bao vây một thành trì. Mỗi vị tướng đóng quân ở một vị trí khác nhau và không thể giao tiếp trực tiếp với toàn bộ đội hình – *chỉ có thể gửi thông điệp thông qua các sứ giả*.

Nhiệm vụ đặt ra là:

tất cả các tướng **phải nhất trí một hành động chung:**

**cùng tấn công,
cùng rút lui.**

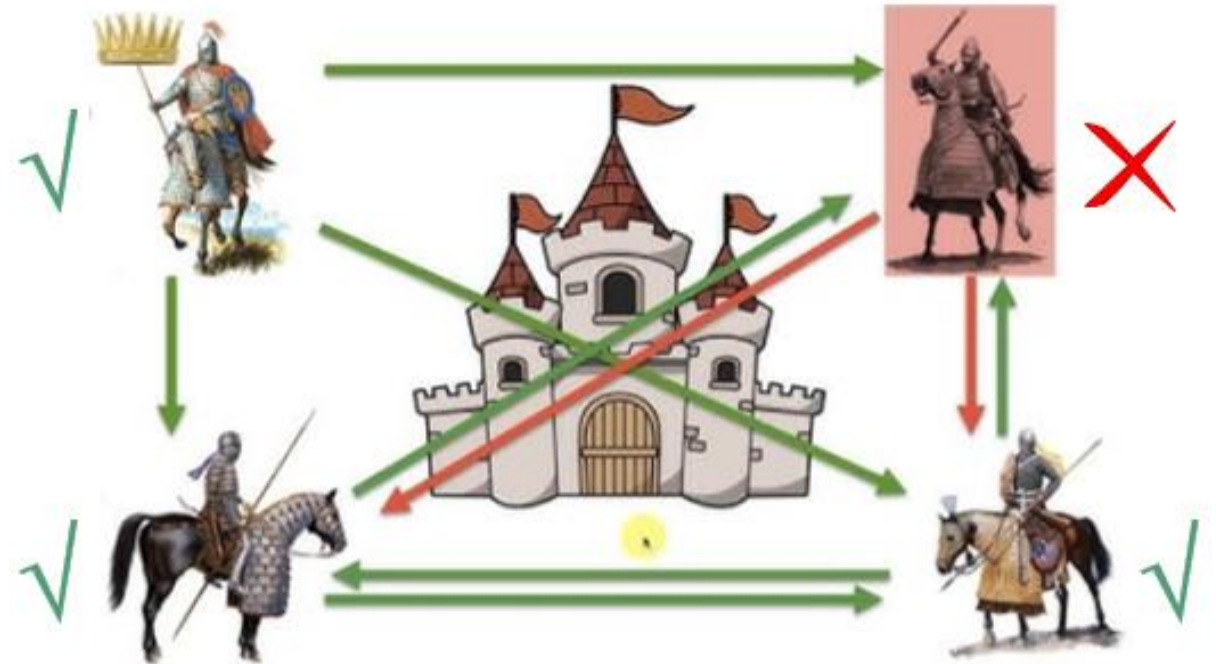
Nếu:

một số tướng đưa ra **quyết định sai lệch hoặc cố tình phản bội,**

→ **toàn bộ kế hoạch có thể sụp đổ.**

Hình minh họa cho thấy:

- Có ba tướng trung thực (✓) cùng muốn tấn công thành.
- Một tướng (✗) bị nhiễm độc thông tin hoặc cố tình phản bội, gửi tín hiệu sai lệch về hành động rút lui.



Byzantine Fault Tolerance

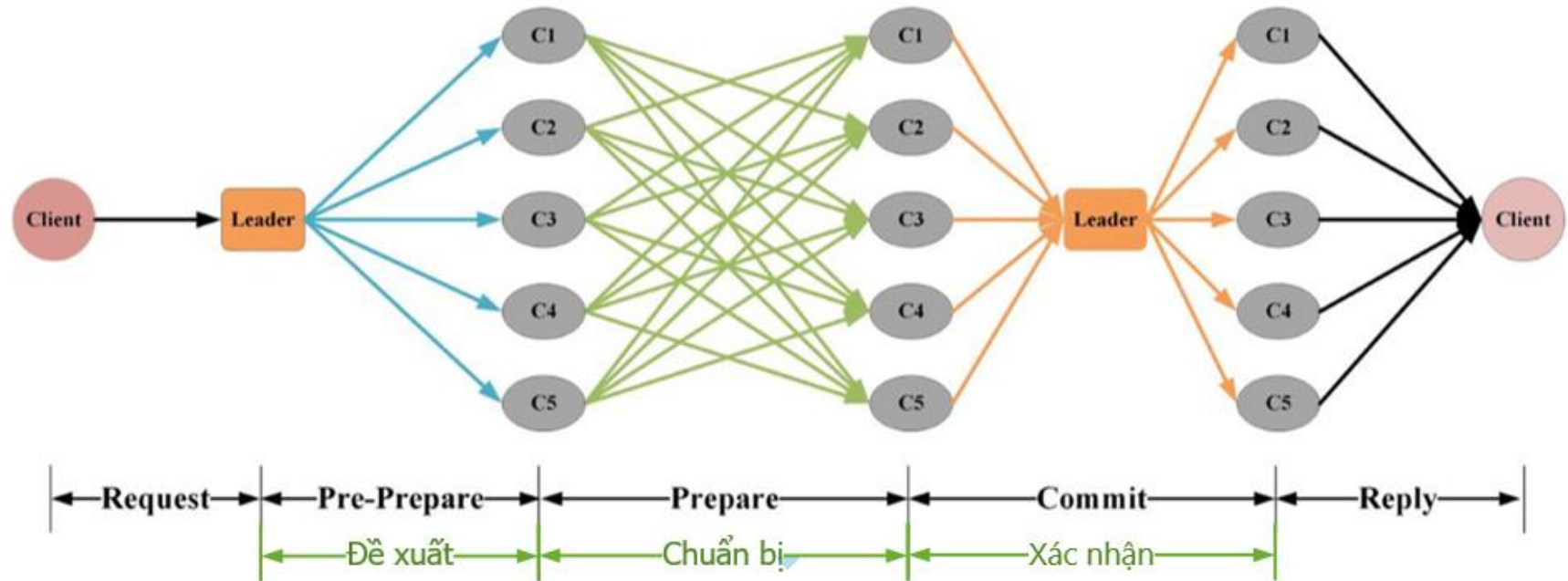
không đạt được sự nhất trí
→ **hành động tập thể thất bại.**

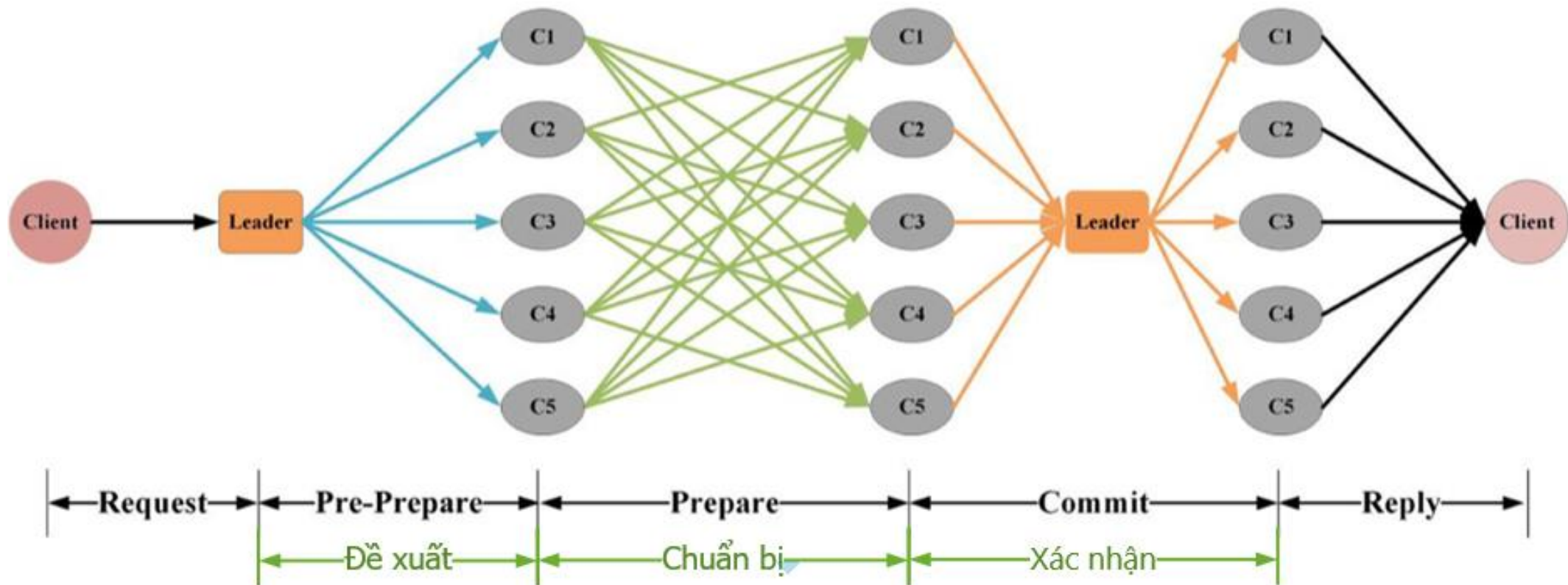
Đây chính là bản chất của vấn đề đồng thuận trong môi trường phân tán với lỗi Byzantine – nơi mà một số node có thể gửi dữ liệu không đồng nhất, gây nguy cơ phá vỡ sự toàn vẹn của hệ thống.

4.2.3.4. Practical Byzantine Fault Tolerance (PBFT) - Chống lỗi Byzantine hiệu quả

a) Định nghĩa:

- ❑ PBFT là cơ chế đồng thuận được thiết kế để xử lý các lỗi **Byzantine** (tức là các hành vi gian lận hoặc tấn công từ các nút bên trong mạng).
- ❑ PBFT đảm bảo tính toàn vẹn của giao dịch ngay cả khi có tới $\frac{1}{3}$ số nút trong mạng có thể bị lỗi hoặc tấn công.





1. **Request** Client gửi yêu cầu đến Leader (**Primary node**) là nút chủ động khởi tạo quá trình đồng thuận.
2. **Pre-Prepare** Leader gửi thông điệp “pre-prepare” cho tất cả Replica (backup) Nodes (C1–C5)
3. **Prepare** **Tất cả Replica nodes** gửi prepare messages cho nhau → xác nhận đã nhận đúng đề xuất
4. **Commit** Sau khi nhận đủ prepare hợp lệ, các nodes gửi commit messages để xác nhận cam kết thực thi
5. **Reply** Sau khi commit thành công, các node trả kết quả về client

Nguyên lý đồng thuận của PBFT

Nếu

có ít nhất $2/3$ số node phản hồi giống nhau ở pha Prepare & Commit,

thì

block sẽ được xác nhận.

→ Cho phép **chịu lỗi Byzantine lên đến:**

$$\mathbf{f} = (n-1)/3 \text{ node lỗi}$$

mà hệ thống vẫn đạt được đồng thuận.

Ưu điểm:

- Phù hợp với các Blockchain riêng tư và doanh nghiệp.
- Tốc độ xử lý giao dịch nhanh và ổn định.

Nhược điểm:

- Không thích hợp cho các mạng công khai có quy mô lớn do chi phí bảo phí cao.
- Đòi hỏi số lượng nút không quá lớn để đảm bảo hiệu năng.

Ví dụ :

- Hyperledger Fabric nền tảng blockchain mã nguồn mở cấp doanh nghiệp, được lưu trữ bởi Linux Foundation.,
- Zilliqa một nền tảng blockchain công khai được thiết kế để có khả năng mở rộng cao và hiệu quả.

Bảng 4.3: So sánh các giao thức đồng thuận

Tiêu chí	PoW	PoS	DPoS	PBFT
Tiêu thụ năng lượng	Cao	Thấp	Thấp	Trung bình
Tốc độ xử lý	Chậm	Nhanh	Rất nhanh	Nhanh
Nguy cơ tập trung quyền lực	Thấp	Trung bình	Cao	Trung bình
Bảo mật	Rất cao	Cao	Cao	Cao
Ứng dụng phù hợp	Công khai (Bitcoin)	Công khai và riêng tư	Công khai quy mô lớn	Riêng tư và doanh nghiệp

Kết luận:

- ✓ Mỗi giao thức đồng thuận có những ưu và nhược điểm riêng phù hợp với từng mục đích sử dụng cụ thể.
- ✓ Việc lựa chọn giao thức nào phụ thuộc vào mục tiêu bảo mật, tốc độ xử lý và đặc điểm của mạng Blockchain.

4.3. BẢO MẬT VÀ MÃ HÓA DỮ LIỆU TRONG BLOCKCHAIN

4.3.1. Mã hóa dữ liệu bằng thuật toán SHA-256

- ❑ Mã hóa (Encryption) là quá trình chuyển đổi dữ liệu từ dạng rõ (***plaintext***) sang dạng mã hóa (***ciphertext***) bằng các thuật toán mật mã, đảm bảo rằng chỉ có những người có khóa giải mã hợp lệ mới có thể truy cập được dữ liệu ban đầu.
- ❑ Trong Blockchain, mã hóa đóng vai trò cốt lõi để đảm bảo tính bảo mật và tính bất biến của dữ liệu [6].

4.3.1.1. Thuật toán SHA-256 là gì?

- SHA-256 (Secure Hash Algorithm 256-bit) là một thuật toán băm mật mã thuộc họ SHA-2, được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA).
- SHA-256 nhận đầu vào là một chuỗi ký tự bất kỳ và chuyển đổi nó thành một chuỗi 256 bit (32 byte) duy nhất.

Đặc tính quan trọng của SHA-256 là:

- **Không thể đảo ngược** (Irreversible): Không thể tính toán ngược từ giá trị băm để tìm ra dữ liệu ban đầu.
- **Tính xác định** (Deterministic): Cùng một đầu vào luôn tạo ra cùng một giá trị băm.
- **Hiệu suất cao**: Xử lý nhanh và hiệu quả trên các nền tảng tính toán khác nhau.
- **Tính nhiễu loạn cao** (Avalanche Effect): Một thay đổi nhỏ trong đầu vào sẽ dẫn đến sự thay đổi hoàn toàn trong giá trị băm đầu ra.

4.3.1.2. Cơ chế SHA-256 hoạt động:

- **Chia dữ liệu thành các khối (blocks):** Dữ liệu đầu vào được chia thành các khối 512-bit. Nếu không đủ, dữ liệu sẽ được thêm đuôi (*padding*) để đạt kích thước phù hợp.
- **Bổ sung đuôi (Padding):** SHA-256 thêm một bit '1' sau đó là các bit '0' để đảm bảo tổng số bit của dữ liệu là bội số của 512.
- **Áp dụng các hàm nén (Compression Functions):** Các khối dữ liệu được xử lý tuần tự qua 64 vòng nén với các hằng số (constants) cố định và phép toán logic như XOR, AND, ROTATE.
- **Sinh giá trị băm cuối cùng:** Kết quả là một chuỗi băm dài 256-bit duy nhất đại diện cho dữ liệu đầu vào.

Ví dụ 4.2. Minh họa mã hóa SHA-256 bằng Python

4.3.1.3. Phân biệt giữa hàm băm và thuật toán mã hóa

Bảng 4.4: Phân biệt Hashing và Encryption

Tiêu chí	Hashing (Băm)	Encryption (Mã hóa)
Mục đích chính	Kiểm tra tính toàn vẹn dữ liệu	Bảo mật nội dung dữ liệu
Khả năng đảo ngược	Kết quả không thể đảo ngược (một chiều)	Kết quả có thể đảo ngược nếu có khóa
Sử dụng khóa	Không dùng khóa	Dùng khóa (đối xứng hoặc bất đối xứng)
Đầu ra	Chuỗi ký tự cố định, không thể dùng để truy nội dung gốc	Dữ liệu đã mã hóa, có thể giải mã lại được
Ứng dụng trong Blockchain	Tạo dấu vết, liên kết block, xác minh dữ liệu	Mã hóa giao dịch, bảo vệ quyền riêng tư người dùng

4.3.2. Xác minh và bảo mật tính toàn vẹn dữ liệu

- ❑ Tính toàn vẹn dữ liệu (Data Integrity) là khả năng đảm bảo rằng dữ liệu không bị thay đổi, giả mạo hoặc mất mát trong quá trình lưu trữ và truyền tải.
- ❑ Trong Blockchain, tính toàn vẹn được đảm bảo thông qua mã băm (hash) và các cơ chế đồng thuận.

4.3.2.1. Cơ chế xác minh tính toàn vẹn dữ liệu trong Blockchain:

a) Mã băm (hash) của khối (Block Hash):

- Mỗi khối trong Blockchain chứa một mã băm (hash) của khối trước đó (Previous Hash).
- Nếu bất kỳ dữ liệu nào trong khối thay đổi, mã băm (hash) của khối đó sẽ thay đổi, làm mất liên kết với các khối sau.

Cây Merkle (Merkle Tree):

- Merkle Root giúp xác minh nhanh chóng tính toàn vẹn của toàn bộ giao dịch trong khối mà không cần kiểm tra từng giao dịch riêng lẻ.

Ví dụ 4.3. Kiểm tra tính toàn vẹn dữ liệu bằng Python:

```
1 import hashlib
2
3 def calculate_hash(data):
4     return hashlib.sha256(data.encode()).hexdigest()
5
6 # Dữ liệu ban đầu
7 data1 = "Giao dịch 1"
8 data2 = "Giao dịch 2"
9
10 # Tính mã băm (hash)
11 hash1 = calculate_hash(data1)
12 hash2 = calculate_hash(data2)
13
14 # Tạo Merkle Root
15 parent_hash = calculate_hash(hash1 + hash2)
16
17 print("Mã băm (hash) giao dịch 1:", hash1)
18 print("Mã băm (hash) giao dịch 2:", hash2)
19 print("Merkle Root:", parent_hash)
```

Kết quả thực hiện chương trình:

Mã băm (hash) giao dịch 1:

5c5aeb8b9536fab953afa94c29865d349de9f050e5ac6b227a770863044d9c9e

Mã băm (hash) giao dịch 2:

ec5159949245af8e64943e981ce1d603a2a73926de963e46ae93dca2c4d5e95d

Merkle Root:

ef779c7eda7705f197725cae1a98c07b5ecf6d023811c1c5925616f4a92dedd9

4.3.2.2. Bảo mật dữ liệu trong Blockchain:

a. Mã hóa bất đối xứng (Asymmetric Encryption):

Blockchain sử dụng mã hóa bất đối xứng để bảo vệ danh tính người dùng và xác thực quyền thực hiện giao dịch.

Cơ chế này dựa trên một cặp khóa:

- **Khóa riêng (Private key)** : Được giữ bí mật bởi người dùng. Dùng để **tạo chữ ký số** cho giao dịch, xác nhận rằng người đó là chủ sở hữu hợp pháp.
- **Khóa công khai (Public key)**: Được chia sẻ công khai với toàn mạng. Dùng để **xác minh chữ ký số** của người gửi, kiểm tra tính toàn vẹn và xác thực nguồn gốc giao dịch.

Public Key và Private

So sánh Public Key và Private Key

Khái niệm	Ví dụ trong thực tế	Trong mật mã
Khóa công khai (Public key)	Địa chỉ email / số tài khoản ngân hàng	Dùng để mã hóa dữ liệu gửi đến người dùng
Khóa riêng (Private key)	Mật khẩu email / mã PIN tài khoản	Dùng để giải mã dữ liệu người dùng nhận được

Khóa công khai (Public key):

- ✓ Ai cũng có thể biết.
- ✓ Dùng để mã hóa thông tin gửi đến bạn.
- ✓ Giống như việc bạn chia sẻ địa chỉ email hoặc số tài khoản ngân hàng cho người khác để họ gửi thư hoặc chuyển tiền cho bạn.

Khóa riêng (Private key):

- ✓ Chỉ bạn mới được giữ.
- ✓ Dùng để giải mã thông tin đã được mã hóa bằng khóa công khai của bạn.
- ✓ Giống như việc chỉ bạn có thể đăng nhập vào tài khoản email, hoặc rút tiền bằng mã PIN.

Lưu ý quan trọng (về pháp lý):

- ❑ Tính đến thời điểm hiện tại, pháp luật Việt Nam chưa công nhận tiền mã hóa như Bitcoin, Ethereum là phương tiện thanh toán hợp pháp, và chưa cấp phép cho bất kỳ sàn giao dịch tiền số nào hoạt động chính thức.
- ❑ Do đó, nội dung liên quan đến ví điện tử trong tài liệu này **chỉ nhằm mục đích học thuật và nghiên cứu công nghệ blockchain**, không khuyến khích sử dụng vào giao dịch tài chính thực tế tại Việt Nam.

4.4. SMART CONTRACT VÀ ỨNG DỤNG BLOCKCHAIN

4.4.1. Ví điện tử (Wallet) và vai trò trong giao dịch

4.4.1.1. Khái niệm ví điện tử trong Blockchain

Ví điện tử trong blockchain là **một phần mềm (hoặc thiết bị phần cứng)** cho phép người dùng *lưu trữ khóa riêng, quản lý danh tính, và thực hiện giao dịch tài sản số một cách an toàn và phi tập trung.*

- **Khóa riêng (Private key):** Dùng để ký số và xác thực giao dịch.
Phải tuyệt đối bí mật.
- **Khóa công khai (Public key):** được tạo ra từ khóa riêng bằng thuật toán mật mã. Dùng để xác thực chữ ký số, nhưng không thể dùng để đoán ngược lại khóa riêng. Khóa công khai được dùng để tạo địa chỉ nhận tài sản số từ mạng Blockchain. Có thể chia sẻ công khai.
- **Địa chỉ ví (Public Address):** là chuỗi định danh được tạo từ khóa công khai, dùng để nhận tài sản trên blockchain (tương tự như số tài khoản ngân hàng). Có thể chia sẻ công khai.



Ví điện tử – nơi lưu trữ tài sản kỹ thuật số trên blockchain

Lưu ý:

- ❑ Ví blockchain *không* duy trì "số dư tài khoản" theo cách truyền thống mà chỉ giúp **truy cập** và **kiểm soát quyền sở hữu**.
- ❑ Hệ thống sẽ quét toàn bộ lịch sử giao dịch và tính toán xem có bao nhiêu tài sản đã được chuyển tới địa chỉ ví của người dùng, và chưa được chi tiêu.

VIỆC BẢO VỆ KHÓA RIÊNG (Private Key) CỰC KỲ QUAN TRỌNG!!!

Nếu: **mất khóa riêng,**

*người dùng sẽ mất toàn bộ quyền truy cập vào tài sản,
& không có cách nào phục hồi lại được.*

Ví dụ: Một ví trên mạng Ethereum không "chứa" ETH, mà chứa khóa riêng để điều khiển địa chỉ sở hữu ETH trên Blockchain.



Ví dụ : Một số vụ mất tài sản lớn vì chủ quan với ví cá nhân



❑ **Rủi ro mất ví do chủ quan cá nhân (James Howells (Anh):**

Vứt nhầm ổ cứng chứa khóa ví 8.000 BTC (~**800 triệu USD**).

❑ **Stefan Thomas (Đức):**

Quên mật khẩu ví IronKey chứa 7.002 BTC, chỉ còn 2 lần thử trước khi khóa vĩnh viễn.

❑ **Peter Schiff (Mỹ):**

Mất Bitcoin vì không ghi lại cụm từ khôi phục (seed phrase).

❑ **QuadrigaCX (Canada):**

CEO qua đời, mang theo toàn bộ khóa ví lạnh chứa 190 triệu USD tài sản của khách hàng.

Bài học rút ra: Trong blockchain, khóa riêng tư = quyền sở hữu tài sản.

Mất khóa = mất vĩnh viễn, không ai khôi phục giúp được.!!!

4.4.1.2. Vai trò trong giao dịch

- ❑ Ở phía gửi, ví **dùng khóa riêng (private key) để tạo chữ ký số xác nhận giao dịch**;
- ❑ Ở phía nhận, **ví nhận tài sản thông qua địa chỉ công khai**.

Khi một người dùng gửi tài sản, **giao dịch sẽ chỉ được mạng xác nhận nếu chữ ký số khớp với khóa công khai đã đăng ký trước đó** – đảm bảo rằng chỉ chủ sở hữu thực sự của tài sản mới có thể chi tiêu.

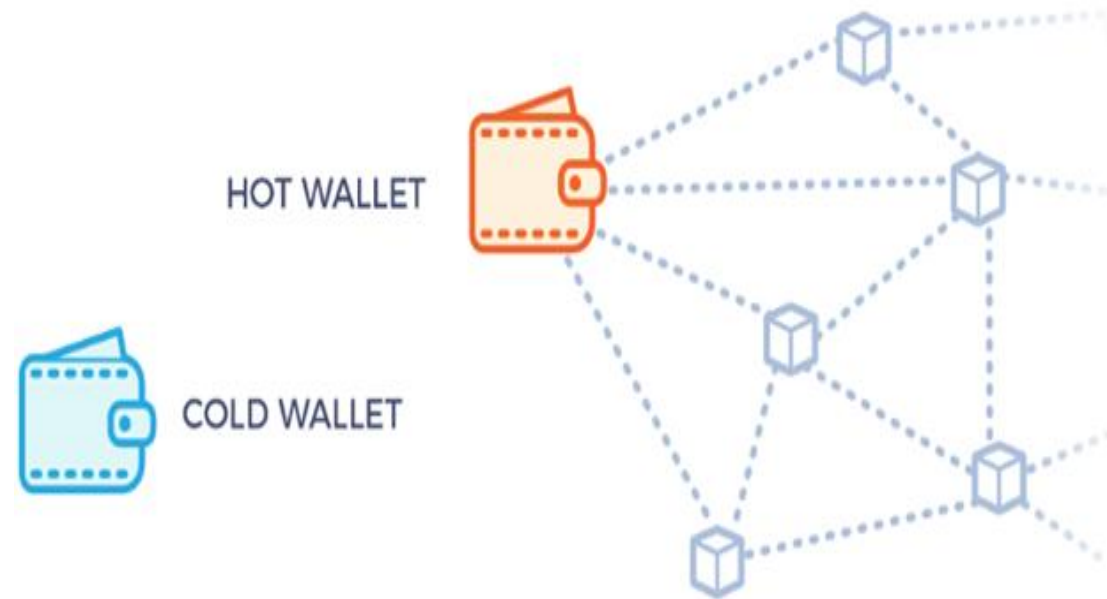
Các chức năng cơ bản của ví trong mạng Blockchain:

- ❖ Khởi tạo giao dịch: Người dùng sử dụng ví để gửi tài sản số đến một địa chỉ khác trong mạng lưới.
- ❖ Nhận tài sản: Ví có thể nhận các đơn vị giá trị kỹ thuật số từ các giao dịch hợp lệ.
- ❖ Xác thực giao dịch: Việc ký số bằng khóa riêng đảm bảo tính xác thực và toàn vẹn của giao dịch.
- ❖ Bảo vệ quyền sở hữu: Ví đóng vai trò như một cổng kiểm soát quyền truy cập đối với tài sản số của người dùng.

4.4.1.3. Phân loại ví điện tử

Ví điện tử có thể tồn tại dưới nhiều hình thức khác nhau, tùy vào cách lưu trữ khóa riêng.

- ❑ Nếu khóa riêng được lưu trữ trong phần mềm hoặc ứng dụng có kết nối Internet, đó được gọi là **ví nóng (hot wallet)** – tiện lợi nhưng *tiềm ẩn rủi ro bảo mật cao hơn*.
- ❑ Ngược lại, nếu khóa riêng được lưu trữ ngoại tuyến (**không kết nối mạng**) , chẳng hạn trong thiết bị USB chuyên dụng hoặc viết ra giấy, thì đó là **ví lạnh (cold wallet)** – *có mức độ bảo mật cao hơn, phù hợp với việc lưu trữ lâu dài*.



Hình 4.12: So sánh ví nóng (hot wallet) và ví lạnh (cold wallet) .

Hot Wallet và Cold Wallet

Ví điện tử nóng (Hot Wallet):

Một số ví phổ biến như MetaMask hoặc **Trust Wallet** cho phép người dùng dễ dàng thực hiện giao dịch trực tuyến, tương thích với trình duyệt web và thiết bị di động.



Hình 4.13: Minh họa ví điện tử nóng lạnh

Ví điện tử lạnh (Cold Wallet): Các thiết bị phần cứng như **Ledger Nano S** hoặc **Trezor** thường được sử dụng để lưu trữ khóa riêng ngoại tuyến, nhằm tăng cường mức độ bảo mật cho tài sản số.

4.4.1.4. Đặc điểm của ví điện tử

Hầu hết các ví đều tích hợp chức năng ký số và gửi giao dịch trực tiếp lên mạng Blockchain.

Helps in Exchange of funds
(Hỗ trợ trao đổi tiền/tài sản)



Blockchain wallet

Privacy is maintained
(Bảo mật danh tính)

Dù mọi giao dịch được ghi lại công khai, danh tính thực tế của người dùng không bị lộ trừ khi họ tự tiết lộ.

Transactions are secure
(Giao dịch được bảo mật)

Việc sử dụng khóa công khai và khóa riêng là cơ chế bảo mật quan trọng để đảm bảo chỉ người sở hữu khóa riêng mới có quyền thực hiện giao dịch.

Hình 4.14: Đặc điểm của ví điện tử

Accessible from web or mobile devices
(Có thể truy cập từ web hoặc thiết bị di động)

Đặc điểm này giúp người dùng linh hoạt truy cập và quản lý tài sản số ở bất kỳ đâu.

4.4. SMART CONTRACT VÀ ỨNG DỤNG BLOCKCHAIN

4.4.2. Smart Contract là gì?

Smart Contract (Hợp đồng thông minh) là một chương trình máy tính tự động thực thi các điều khoản và điều kiện của hợp đồng khi các điều kiện được định sẵn được đáp ứng. Không giống như các hợp đồng truyền thống yêu cầu bên thứ ba để xác thực và thực thi, *Smart Contract hoạt động trên nền tảng Blockchain*, đảm bảo tính *minh bạch*, *bảo mật* và *phi tập trung* [3, 11].

Ví dụ:

Một hợp đồng thông minh có ***thể tự động chuyển tiền*** từ tài khoản A sang tài khoản B nếu một sản phẩm đã được giao nhận thành công.

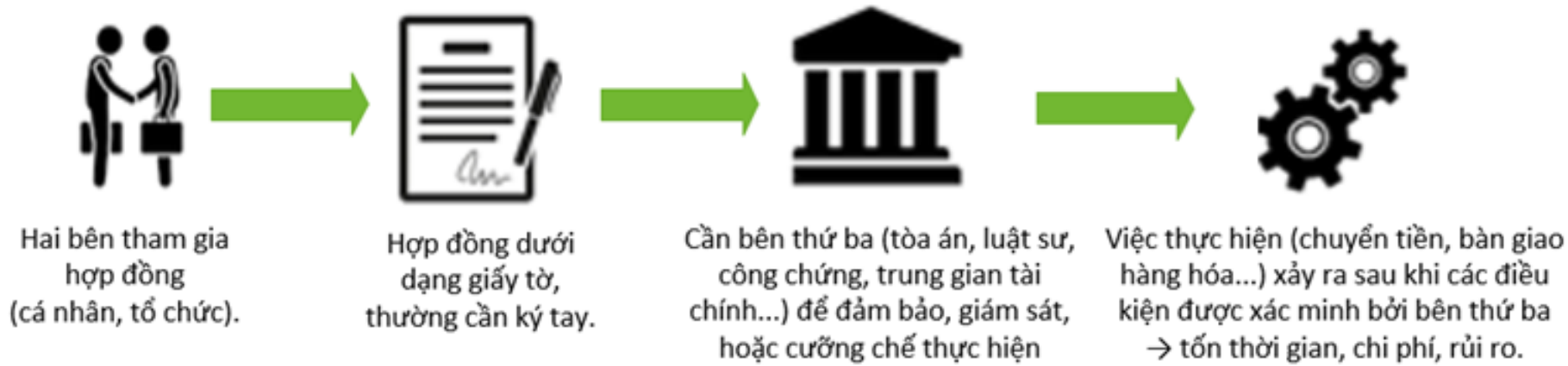
Điều này giúp đảm bảo tính minh bạch và an toàn của giao dịch.

Các nền tảng hỗ trợ Smart Contract phổ biến:

- **Ethereum**: Nền tảng phổ biến nhất, sử dụng ngôn ngữ lập trình Solidity.
- **Binance Smart Chain** (BSC): Tương thích với Ethereum nhưng có phí giao dịch thấp hơn.
- **Solana, Cardano**: Sử dụng các ngôn ngữ như Rust, Plutus với tốc độ xử lý nhanh.

4.4.2. Smart Contract là gì?

HỢP ĐỒNG TRUYỀN THỐNG (TRADITIONAL CONTRACT)



HỢP ĐỒNG THÔNG MINH (SMART CONTRACT)



Hình 4.12: So sánh hợp đồng truyền thống và hợp đồng thông minh

4.4.2.1. Hoạt động của Smart Contract:

- ❑ Smart Contract như một bộ giao thức giúp thực thi các điều khoản của hợp đồng một cách **tự động** mà **không cần đến sự can thiệp của bên thứ ba** [Nick Szabo - 1994].

Cách hoạt động:

- ❖ Smart Contract được viết bằng các ngôn ngữ lập trình như *Solidity, Vyper, JavaScript*
- ❖ Được triển khai trên các Blockchain hỗ trợ hợp đồng thông minh như **Ethereum, Binance Smart Chain, Solana**.
- ❖ Mỗi khi các điều kiện được đáp ứng, **hợp đồng sẽ tự động thực thi** các hành động tương ứng như *chuyển tiền, cập nhật dữ liệu hoặc kích hoạt* các hợp đồng khác.

4.4.2.2. Các đặc điểm của Smart Contract:

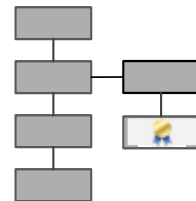
Hợp đồng thông minh là một thành phần cốt lõi trong hệ sinh thái blockchain, mang lại sự đổi mới trong cách thức thiết lập, thực thi và giám sát các thỏa thuận số.

Đặc điểm nổi bật của hợp đồng thông minh, được phân thành hai nhóm: **nền tảng kỹ thuật** và **đặc điểm vận hành – quản lý**.

8 ĐẶC ĐIỂM CHÍNH CỦA HỢP ĐỒNG THÔNG MINH (Smart Contract)



1. Lập trình (Programmability):
Hợp đồng được viết dưới dạng mã lập trình và triển khai trên blockchain.



2. Tính minh bạch (Transparency):
mọi người đều có thể xem được logic điều kiện của hợp đồng.
Hợp đồng là một phần không thể tách rời của blockchain công khai.



3. Tính ẩn danh (Anonymity):
Các bên tham gia hợp đồng là ẩn danh (**địa chỉ ví blockchain**)



4. Tính bất biến (Immutability):
Sau khi được triển khai, mã nguồn và điều kiện của hợp đồng không thể bị thay đổi, đảm bảo ngăn chặn gian lận



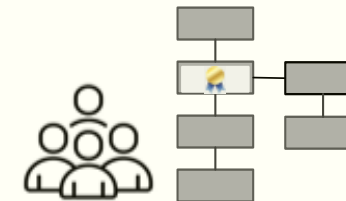
5. Bảo mật cao (Security):
Hợp đồng được mã hóa và lưu trữ trên Blockchain, giúp ngăn chặn việc giả mạo hoặc thay đổi dữ liệu.



6. Hợp đồng tự động thực thi (Auto-execution)
khi điều kiện được thỏa mãn.



7. Tiết kiệm chi phí giao dịch (Cost-efficiency)
Loại bỏ sự cần thiết của các bên trung gian như ngân hàng, luật sư



8. Cơ quan quản lý có thể giám sát hợp đồng (Auditability) qua blockchain.

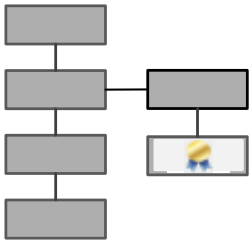
4.4.2.2. Các đặc điểm của Smart Contract:

Đặc điểm vận hành – quản lý. Nền tảng kỹ thuật

8 ĐẶC ĐIỂM CHÍNH CỦA HỢP ĐỒNG THÔNG MINH (Smart Contract)



1. Lập trình (Programmability):
Hợp đồng được viết dưới dạng mã lập trình và triển khai trên blockchain.



2. Tính minh bạch (Transparency):
mọi người đều có thể xem được logic điều kiện của hợp đồng. Hợp đồng là một phần không thể tách rời của blockchain công khai.



3. Tính ẩn danh (Anonymity):
Các bên tham gia hợp đồng là ẩn danh (địa chỉ ví blockchain)



4. Tính bất biến (Immutability) :
Sau khi được triển khai, mã nguồn và điều kiện của hợp đồng không thể bị thay đổi, đảm bảo ngăn chặn gian lận



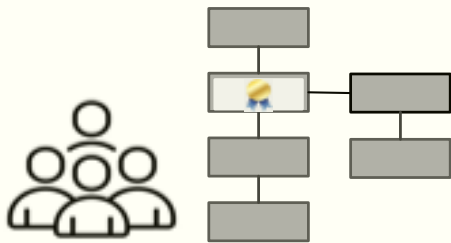
5. Bảo mật cao (Security):
Hợp đồng được mã hóa và lưu trữ trên Blockchain, giúp ngăn chặn việc giả mạo hoặc thay đổi dữ liệu.



6. Hợp đồng tự động thực thi (Auto-execution)
khi điều kiện được thỏa mãn.



7. Tiết kiệm chi phí giao dịch (Cost-efficiency)
Loại bỏ sự cần thiết của các bên trung gian như ngân hàng, luật sư



8. Cơ quan quản lý có thể giám sát hợp đồng (Auditability) qua blockchain.

4.4.2.3. Ứng dụng của Smart Contract:

a) Tài chính phi tập trung (DeFi):

- ❑ *Hợp đồng thông minh* là nền tảng kỹ thuật của các giao thức tài chính phi tập trung (**DeFi**), cho phép người dùng *thực hiện các hoạt động như giao dịch token, gửi tiết kiệm, vay và cho vay tài sản số* mà **không cần thông qua tổ chức tài chính trung gian**.

Ví dụ:

- Trên các nền tảng như Uniswap, người dùng có thể hoán đổi token trực tiếp với hợp đồng thông minh thông qua các *pool thanh khoản* (liquidity pool).
- Trong khi đó, các nền tảng như Aave hoặc Compound cho phép người dùng gửi tài sản vào hợp đồng thông minh để nhận lãi suất, hoặc vay tài sản bằng cách thế chấp các token khác.
- Tất cả các thao tác trên được thực hiện tự động, minh bạch, và không cần sự can thiệp của bên thứ ba, đồng thời mọi giao dịch đều được ghi lại vĩnh viễn trên blockchain

4.4.2.3. Ứng dụng của Smart Contract:

b) Quản trị phi tập trung (DAO - Decentralized Autonomous Organization):

- ❑ Các tổ chức phi tập trung sử dụng Smart Contract để thực thi các quyết định của cộng đồng thông qua cơ chế bỏ phiếu minh bạch.

Ví dụ:

- Một dự án blockchain có thể thành lập một tổ chức DAO để quản lý quỹ phát triển sản phẩm.
- Những người nắm giữ token của dự án có quyền biểu quyết bằng cách tham gia bỏ phiếu trên nền tảng phi tập trung như Snapshot hoặc Aragon.
- Khi một đề xuất (proposal) được đa số tán thành, hợp đồng thông minh sẽ tự động thực thi quyết định, ví dụ như giải ngân ngân sách cho một nhóm phát triển, thay đổi lộ trình kỹ thuật, hoặc cập nhật thông số giao thức.

4.4.2.3. Ứng dụng của Smart Contract (tiếp,...)

c) Chuỗi cung ứng (Supply Chain):

- ❑ Smart Contract có thể theo dõi quá trình vận chuyển hàng hóa theo thời gian thực, **đảm bảo dữ liệu không bị chỉnh sửa** và **tự động xác nhận thanh toán** khi hàng đến nơi.

Ví dụ:

- Trong ngành cà phê, một chuỗi cung ứng có thể sử dụng hợp đồng thông minh để tự động ghi nhận và xác minh quá trình vận chuyển từ nông trại đến nhà rang xay.
- Mỗi lần sản phẩm được vận chuyển qua một điểm trung gian (như kho lưu trữ, đơn vị logistics), một thiết bị IoT hoặc người giám sát sẽ gửi dữ liệu (về thời gian, vị trí, nhiệt độ...) lên blockchain thông qua hợp đồng thông minh.
- Hợp đồng sẽ tự động ghi nhận các mốc vận chuyển, và nếu tất cả các điều kiện chuỗi được thỏa mãn (ví dụ: không quá nhiệt độ 25°C, không trễ quá 2 giờ...), hệ thống có thể kích hoạt thanh toán tự động cho bên vận chuyển.

4.4.2.3. Ứng dụng của Smart Contract (tiếp,...)

d) Bảo hiểm:

- ❑ Hợp đồng thông minh có thể được ứng dụng trong lĩnh vực bảo hiểm để tự động xử lý chi trả bồi thường khi các điều kiện được xác minh qua dữ liệu bên ngoài (oracle).

Ví dụ,

Trong một hợp đồng bảo hiểm nông nghiệp, nếu dữ liệu thời tiết từ nguồn đáng tin cậy (chẳng hạn như API của cơ quan khí tượng) cho thấy lượng mưa dưới ngưỡng trong nhiều ngày liên tiếp, hợp đồng thông minh sẽ tự động xác định sự kiện mất mùa đã xảy ra và thực hiện thanh toán bồi thường cho người nông dân – không cần qua trung gian hay xử lý thủ công.

Ví dụ: Các công ty bảo hiểm như Etherisc đã triển khai sản phẩm bảo hiểm chuyến bay, trong đó nếu một chuyến bay bị hoãn quá một khoảng thời gian nhất định (dựa trên dữ liệu của hãng hàng không), hợp đồng tự động thanh toán cho khách hàng mà không cần yêu cầu nộp đơn hay bằng chứng giấy tờ.

4.4.2.3. Ứng dụng của Smart Contract (tiếp,...)

e) Quản lý tài sản số (NFT và bản quyền):

- ❑ Hợp đồng thông minh có vai trò quan trọng trong việc **xác thực quyền sở hữu và giao dịch tài sản số**, đặc biệt là các tài sản không thể thay thế (NFT – Non-Fungible Token).
- ❑ Các hợp đồng này có thể đại diện cho ảnh, video, âm nhạc, vật phẩm trong trò chơi hoặc bất kỳ nội dung kỹ thuật số nào, và đảm bảo rằng mỗi tài sản đều là duy nhất, có nguồn gốc xác thực và được kiểm soát tự động.
- ❑ Hợp đồng thông minh còn cho phép tác giả gốc của tài sản số được nhận phí bản quyền (royalty) mỗi khi tài sản được chuyển nhượng.

Ví dụ: Các nền tảng như OpenSea, Rarible hay các game blockchain sử dụng hợp đồng thông minh để mã hóa quyền sở hữu, quản lý giao dịch và tạo điều kiện phát triển thị trường tài sản số một cách minh bạch và phi tập trung.

4.4.3. Viết Smart Contract đơn giản

Viết và triển khai một Smart Contract đơn giản bằng cách sử dụng Python và thư viện **Web3.py**

4.4.3.1. Giới thiệu ngôn ngữ Solidity:

Solidity là một ngôn ngữ lập trình hướng đối tượng được phát triển đặc biệt để viết các hợp đồng thông minh chạy trên Ethereum Virtual Machine (EVM). Với cú pháp tương tự như JavaScript và C++, Solidity giúp lập trình viên dễ dàng triển khai và quản lý các ứng dụng phi tập trung (DApps) và quản lý các giao dịch một cách minh bạch và bảo mật.

❑ Đặc điểm của Solidity:

- **Hướng đối tượng** (Object-Oriented): Hỗ trợ lập trình với các lớp (class), kế thừa (inheritance) và giao diện (interface).
- **Statically Typed**: Kiểu dữ liệu cần được xác định trước khi sử dụng.
- **Tích hợp mã hóa**: Hỗ trợ các hàm băm như SHA-256 và các phương thức ký số.
- **Phí gas**: Mỗi lệnh trong Solidity đều có chi phí **gas**, đòi hỏi lập trình viên phải tối ưu mã nguồn để giảm chi phí khi triển khai.

4.4.3. Viết Smart Contract đơn giản, (tiếp ...)

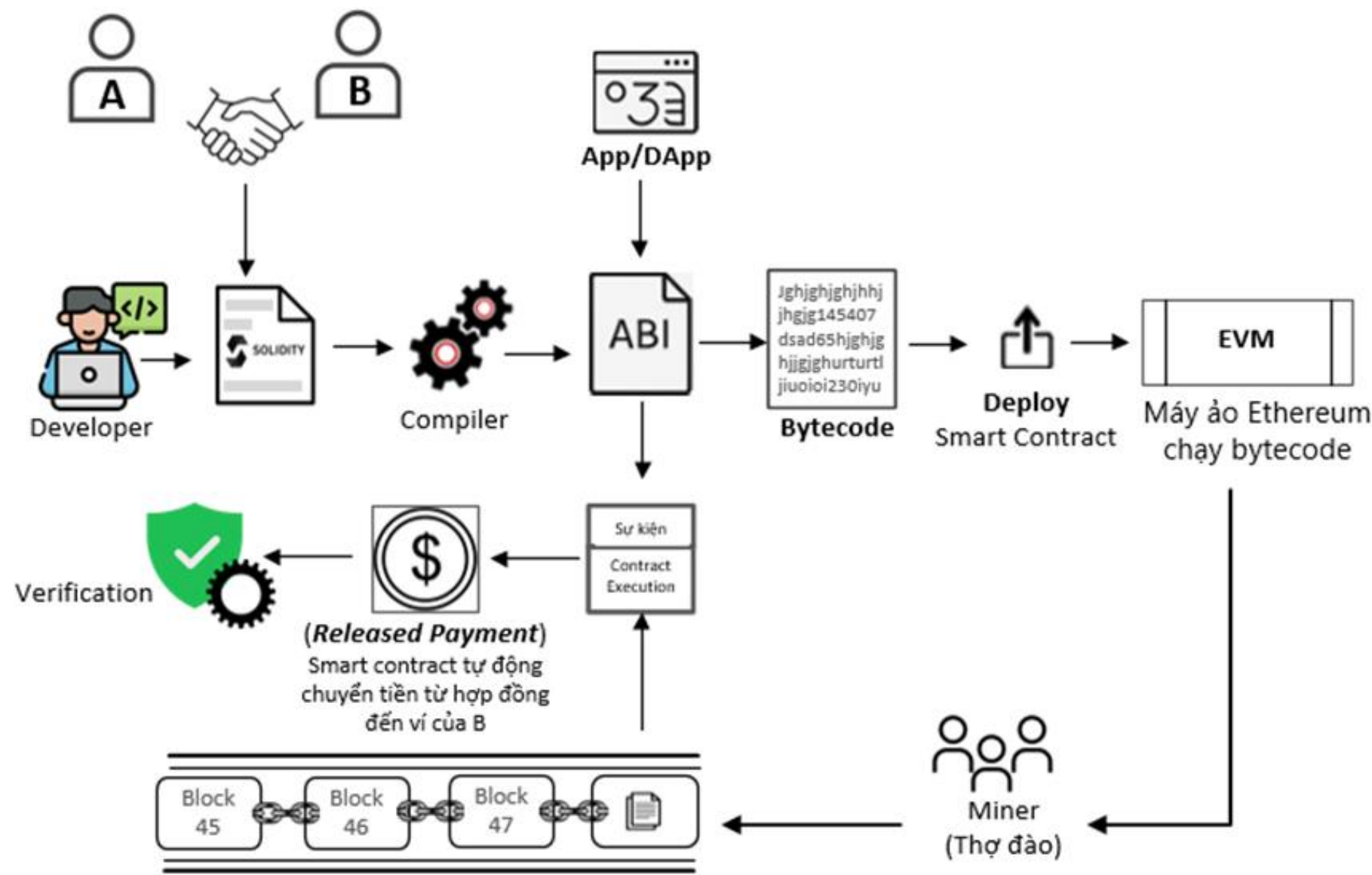
Khái niệm Gas

Gas là đơn vị đo lường sức mạnh tính toán cần thiết để thực hiện một hành động hoặc giao dịch trên blockchain Ethereum, chẳng hạn như:

- Gửi ETH cho người khác
- Ký hợp đồng thông minh
- Tương tác với DApp (ứng dụng phi tập trung)

☞ **Chúng ta có thể hình dung Gas giống như “nhiên liệu” để xe chạy – blockchain muốn thực thi lệnh thì cần năng lượng, và đó chính là Gas.**

4.4.3. Viết Smart Contract đơn giản, (tiếp ...)



Hình 4.18: Quy trình triển khai và thực thi hợp đồng thông minh (Smart Contract) trên nền tảng Ethereum

Các bước chính trong quy trình (hình 4.18)

1. **A & B (Thỏa thuận):** Bắt đầu quy trình bằng một sự đồng thuận giữa hai bên tham gia, là cơ sở để xây dựng nội dung hợp đồng thông minh.
2. **Developer & Solidity:** Nhà phát triển viết mã nguồn hợp đồng thông minh bằng ngôn ngữ Solidity, phản ánh các điều khoản đã thỏa thuận.
3. **Compiler:** Mã nguồn được biên dịch để tạo ra hai thành phần chính:
 - Bytecode: mã máy dùng để triển khai trên blockchain
 - ABI (Application Binary Interface): giao diện lập trình cho phép các ứng dụng tương tác với hợp đồng.
4. **Triển khai (Deploy):** Đây là bước đưa Bytecode lên blockchain thông qua một giao dịch triển khai, giúp khởi tạo hợp đồng và gán địa chỉ cố định cho nó. Việc bổ sung rõ ràng khối "Deploy" trong sơ đồ giúp mô tả đầy đủ vòng đời hợp đồng.
5. **Bytecode & EVM:** Sau khi triển khai, Bytecode được lưu trữ và thực thi bởi Ethereum Virtual Machine (EVM) trên mỗi nút của mạng.
6. **Miner & Blockchain:** Các nút (thợ đào) xác thực giao dịch triển khai và mọi tương tác sau này với hợp đồng. Kết quả được ghi vĩnh viễn vào chuỗi khối (blockchain).
7. **ABI & App/DApp:** Ứng dụng phi tập trung (DApp) sử dụng ABI và địa chỉ hợp đồng để gửi yêu cầu gọi hàm hợp đồng thông qua thư viện Web3 (Web3.js/Web3.py).
8. **Contract Execution & Sự kiện:** Khi hợp đồng được gọi, EVM thực thi hàm tương ứng, có thể sinh ra các sự kiện (event) để thông báo trạng thái cho hệ thống bên ngoài.
9. **Released Payment:** Nếu hợp đồng chứa logic chi trả (như bảo hiểm, vay vốn...), việc thực thi có thể dẫn đến việc tự động chuyển tiền cho các bên liên quan.
10. **Verification:** Các bước kiểm tra hoặc xác nhận kết quả (nếu có), nhằm đảm bảo tính đúng đắn, an toàn, và minh bạch cho toàn bộ quá trình.

4.4.3. Viết Smart Contract đơn giản

4.4.3.1. Giới thiệu ngôn ngữ Solidity:

a) *Khái niệm:* Solidity là một ngôn ngữ lập trình hướng đối tượng được phát triển đặc biệt để viết các hợp đồng thông minh chạy trên Ethereum Virtual Machine (EVM). Với cú pháp tương tự như JavaScript và C++, Solidity giúp lập trình viên dễ dàng triển khai và quản lý các ứng dụng phi tập trung (DApps) và quản lý các giao dịch một cách minh bạch và bảo mật.

b) *Đặc điểm của Solidity:*

- Hướng đối tượng (Object-Oriented): Hỗ trợ lập trình với các lớp (class), kế thừa (inheritance) và giao diện (interface).
- Statically Typed: Kiểu dữ liệu cần được xác định trước khi sử dụng.
- Tích hợp mã hóa: Hỗ trợ các hàm băm như SHA-256 và các phương thức ký số.
- **Phí gas:** Mỗi lệnh trong Solidity đều có chi phí **gas**, đòi hỏi lập trình viên phải tối ưu mã nguồn để giảm chi phí khi triển khai.

4.4.3.2. *Viết Smart Contract*

Smart Contract được viết bằng ngôn ngữ Solidity[3], tuy nhiên trong khuôn khổ tài liệu này để tiết kiệm thời gian chúng ta hoàn toàn có thể viết Smart Contract với thư viện Web3.py và ngôn ngữ lập trình Python.

Lưu ý: có thể tìm hiểu thêm về hợp đồng thông minh với ngôn ngữ Solidity tại <https://docs.soliditylang.org/en/v0.8.29/>

Chuẩn bị:

Cài đặt **Web3.py**:

Để bắt đầu, cần cài thư viện Web3.py bằng pip:

pip install web3

4.4.3.2. *Viết Smart Contract*

Bước 1. Viết Smart Contract bằng Solidity (lưu dưới dạng file **Message.sol**):

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract MessageContract {
    string public message;

    constructor(string memory _message) {
        message = _message;
    }

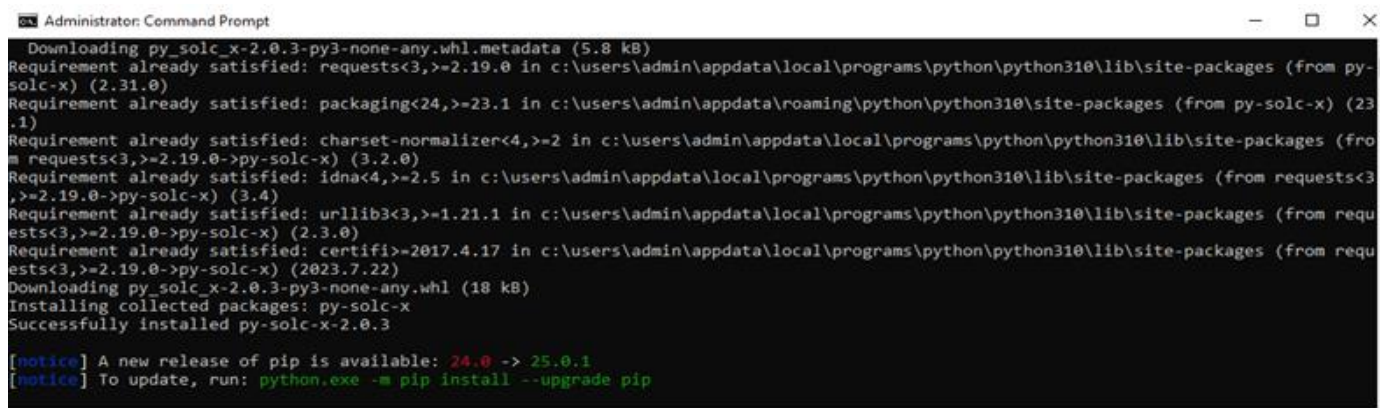
    function updateMessage(string memory _newMessage)
    public {
        message = _newMessage;
    }
}
```

Bước 2: Biên dịch Smart Contract bằng solc (Solidity Compiler):

Cài đặt **solcx** để biên dịch hợp đồng:

pip install py-solc-x

pip install solcx *# Để biên dịch Smart Contract*



```
Administrator: Command Prompt
Downloading py_solc_x-2.0.3-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: requests<3,>=2.19.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from py-solc-x) (2.31.0)
Requirement already satisfied: packaging<24,>=23.1 in c:\users\admin\appdata\roaming\python\python310\site-packages (from py-solc-x) (23.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from requests<3,>=2.19.0->py-solc-x) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from requests<3,>=2.19.0->py-solc-x) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from requests<3,>=2.19.0->py-solc-x) (2.3.0)
Requirement already satisfied: certifi<=2017.4.17 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from requests<3,>=2.19.0->py-solc-x) (2023.7.22)
Downloading py_solc_x-2.0.3-py3-none-any.whl (18 kB)
Installing collected packages: py-solc-x
Successfully installed py-solc-x-2.0.3

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Sau khi cài xong, gõ lệnh '*pip check*' để kiểm tra, C:\>pip check

nếu kết quả là No broken requirements found.

nghĩa là tất cả các xung đột về thư viện đã được giải quyết hoàn toàn.

Biên dịch hợp đồng

```
1 from solcx import set_solc_version, compile_standard
2
3 # Đặt đúng phiên bản `solc` đã cài
4 set_solc_version('v0.8.20')
5 PATH = 'Đường dẫn đến file Message.sol'
6 # Đọc nội dung của file `Message.sol` và lưu vào biến `contract_source_code`
7 with open(r"PATH", "r", encoding="utf-8") as file:
8     contract_source_code = file.read()
9 print("Đã đọc mã nguồn hợp đồng:")
10 print(contract_source_code[:200] + "...") # In thử 200 ký tự đầu để kiểm tra
11 # Dùng `compile_standard` thay vì `set_solc_version` để biên dịch hợp đồng
12 compiled_sol = compile_standard({
13     "language": "Solidity",
14     "sources": {
15         "Message.sol": {
16             "content": contract_source_code
17         }
18     },
19     "settings": {
20         "outputSelection": {
21             "*": {
22                 "*": ["abi", "evm.bytecode"]
23             }
24         }
25     }
26 })
27 print("Đã biên dịch thành công!")
```

Kết quả thực hiện chương trình:

Đã đọc mã nguồn hợp đồng:

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.0;
```

```
contract MessageContract {  
    string public message;  
  
    constructor(string memory _message) {  
        message = _message;  
    }  
  
    functio...
```

Đã biên dịch thành công!

Từ kết quả chương trình chúng ta thấy rằng :

Đã biên dịch thành công:

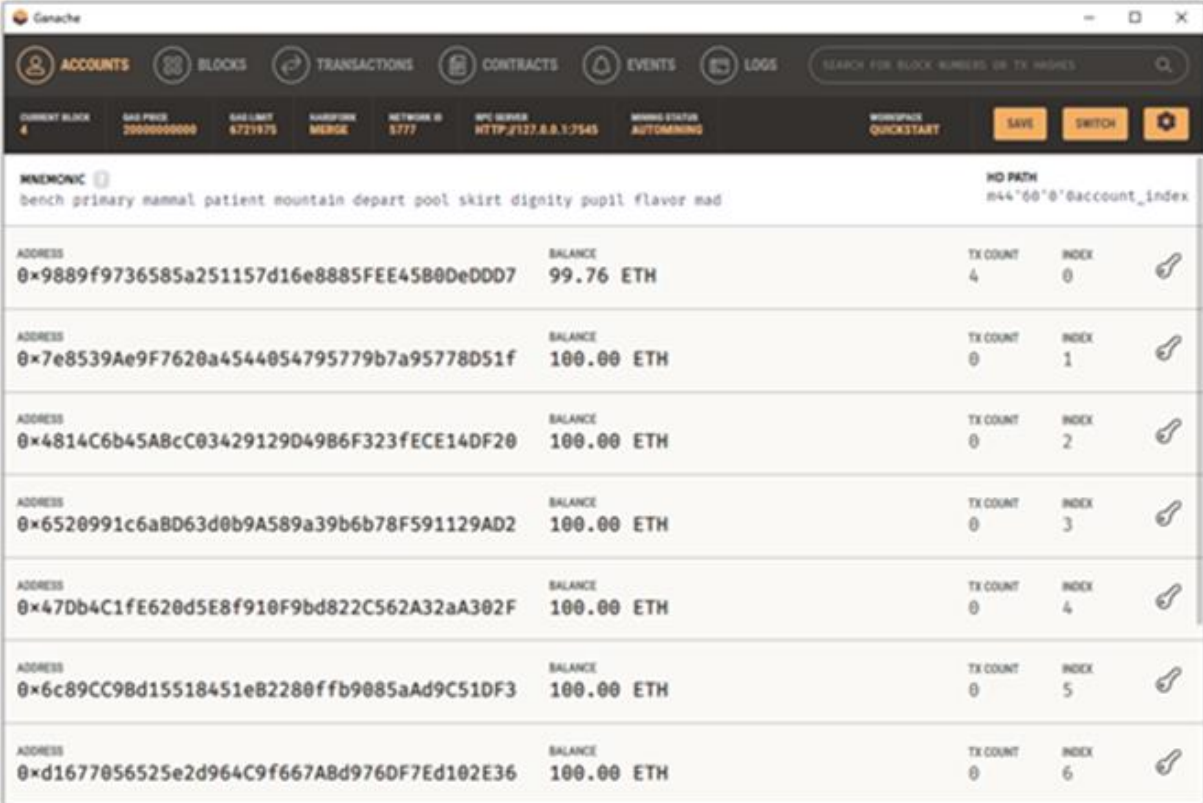
Mã nguồn đã được biên dịch thành công thành ABI (Application Binary Interface) và Bytecode, sẵn sàng để triển khai lên Blockchain.

Bước 3. Cài đặt và sử dụng công cụ Ganache:

Link tải và cài đặt ganache:

<https://archive.trufflesuite.com/ganache/>

Ganache là một công cụ giả lập **mạng Blockchain Ethereum cục bộ**, cho phép nhà phát triển kiểm thử và triển khai hợp đồng thông minh (Smart Contract) một cách nhanh chóng, an toàn và không cần kết nối mạng thật (mainnet). Đây là sản phẩm thuộc bộ công cụ Truffle Suite, nổi tiếng trong cộng đồng phát triển ứng dụng phi tập trung (dApp).



The screenshot shows the Ganache application window. At the top, there's a navigation bar with tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this, a status bar displays various metrics like CURRENT BLOCK, GAS PRICE, GAS LIMIT, HARDFORK, NETWORK ID, RPC URL, and MINING STATUS. The main area shows a list of accounts with columns for ADDRESS, BALANCE, TX COUNT, and INDEX. Each account entry includes a mnemonic phrase and a HD PATH.

ADDRESS	BALANCE	TX COUNT	INDEX
0x9889f9736585a251157d16e8885FEE45B0DeDDD7	99.76 ETH	4	0
0x7e8539Ae9F7620a4544054795779b7a95778D51f	100.00 ETH	0	1
0x4814C6b45ABcC03429129D49B6F323fECE14DF20	100.00 ETH	0	2
0x6520991c6a8D63d0b9A589a39b6b78F591129AD2	100.00 ETH	0	3
0x47Db4C1fE620d5E8f910F9bd822C562A32aA302F	100.00 ETH	0	4
0x6c89CC98d15518451eB2280ffb9085aAd9C51DF3	100.00 ETH	0	5
0xd1677056525e2d964C9f667ABd9760F7Ed102E36	100.00 ETH	0	6

Các bước viết code tiếp theo có thể đọc trong mã code

Lưu ý:

- ❑ Trong môi trường thực tế, việc triển khai hợp đồng thông minh (Smart Contract) đòi hỏi **phải hiểu rõ về các rủi ro bảo mật, như lỗi logic, overflow, reentrancy attack,...**
- ❑ Ngoài ra, mỗi giao dịch triển khai hay thực thi hợp đồng đều tiêu tốn phí gas, nên việc tối ưu code cũng là yếu tố quan trọng.
- ❑ Sinh viên nên tham khảo thêm các khuyến nghị kiểm thử và tài liệu của nền tảng Ethereum để đảm bảo an toàn khi triển khai thật.

4.5. XÂY DỰNG CHUỖI KHỐI

Phần này nhằm giúp sinh viên tổng hợp và áp dụng các kiến thức lý thuyết đã học trong chương 4 vào xây dựng chuỗi khối, từ đó hình thành tư duy hệ thống về cách một nền tảng Blockchain hoạt động từ cấp độ mạng máy tính cho tới ứng dụng thực tiễn thông qua Smart Contract.

Các bài thực hành được xây dựng với mức độ tăng dần, từ cấu trúc chuỗi khối, mã hóa dữ liệu, xác minh tính toàn vẹn cho đến triển khai hợp đồng thông minh trên mạng thử nghiệm sử dụng Python.

4.5.1. Xây dựng chuỗi Blockchain đơn giản bằng Python

Trong bài thực hành này, sinh viên sẽ tạo một mô hình chuỗi khối đơn giản với các thành phần cơ bản: khối, hàm băm, liên kết khối

Các yêu cầu gồm:

- Thiết kế lớp Block chứa các trường index, timestamp, data, previous_hash và hash.
- Viết hàm create_genesis_block() và next_block() để sinh chuỗi khối.
- In ra toàn bộ chuỗi Blockchain và quan sát tính liên kết qua trường previous_hash.

Kết quả cần đạt: Sinh viên quan sát được mối liên hệ giữa các khối và hiểu vai trò của hàm băm trong việc đảm bảo tính toàn vẹn dữ liệu.

4.5.2. Mã hóa và xác minh dữ liệu trong chuỗi khối

Sinh viên được hướng dẫn sử dụng thuật toán băm SHA-256 để:

- Tạo mã băm (hash) cho một khối dữ liệu bất kỳ.
- Tự kiểm tra tính toàn vẹn bằng cách thay đổi dữ liệu và quan sát thay đổi của mã băm (hash).
- Hiểu cách ứng dụng SHA-256 trong bảo vệ chuỗi khối khỏi hành vi sửa đổi dữ liệu.

Kết quả cần đạt được: Sinh viên nhận thức rõ nguyên lý "bất biến" trong Blockchain và tầm quan trọng của mã hóa một chiều.

4.6. HIỂU ĐÚNG – HIỂU SAI VỀ BLOCKCHAIN

a) **Blockchain đáng tin vì không ai có thể can thiệp?**

- ✓ Hiểu sai: Blockchain hoàn toàn phi tập trung, nên không ai có quyền thay đổi hay can thiệp.
- ✓ Hiểu đúng: Dù blockchain có cấu trúc phi tập trung, nhưng việc xác minh giao dịch và ghi nhận vào chuỗi phụ thuộc vào cơ chế đồng thuận (như PoW, PoS...). Nếu các node thực thi sai mã, các node còn lại vẫn có thể từ chối đồng thuận.

b) **Blockchain vô danh hoàn toàn?**

- ✓ Hiểu sai: Blockchain bí danh nên người dùng hoàn toàn vô danh.
- ✓ Hiểu đúng: Blockchain chỉ cung cấp tính bí danh (pseudonymous), tài khoản với địa chỉ công khai có thể được truy vết và phân tích giao dịch. Việc kết hợp với thông tin ngoại tuyến có thể dẫn đến việc đính danh.

c) **Blockchain đồng nghĩa với Bitcoin?**

- ✓ Hiểu sai: Blockchain chỉ dùng để giao dịch tiền ảo như Bitcoin.
- ✓ Hiểu đúng: Blockchain là công nghệ nền tảng, có thể dùng để lưu trữ dữ liệu và đồng bộ thông tin trong nhiều lĩnh vực: chuỗi cung ứng, y tế, giáo dục, bảo hiểm, hợp đồng thông minh...

d) **Blockchain là bất biến, vô hiểu hoá và vĩnh viễn?**

- ✓ Hiểu sai: Blockchain không thể bị thay đổi, nên dữ liệu được lưu trữ mãi mãi.
- ✓ Hiểu đúng: Về nguyên lý, Blockchain được thiết kế để không thay đổi (immutability), nhưng vẫn có khả năng fork (chia nhánh) hoặc can thiệp bằng đa số node nửa (51% attack), hoặc theo quyết định đồng thuận của cộng đồng.

CÂU HỎI ÔN TẬP

- Câu 1. Trình bày định nghĩa công nghệ Blockchain. Nêu rõ các đặc điểm chính giúp Blockchain trở nên tin cậy và khác biệt so với cơ sở dữ liệu truyền thống.
- Câu 2. So sánh mô hình Client/Server với mô hình Peer-to-Peer (P2P) trong bối cảnh ứng dụng Blockchain. Ưu nhược điểm của từng mô hình là gì?
- Câu 3. Mô tả cấu trúc của một khối trong Blockchain. Giải thích vai trò của phần header và phần dữ liệu giao dịch.
- Câu 4. Các giao thức đồng thuận như PoW, PoS và PBFT hoạt động như thế nào? Phân tích ưu nhược điểm của từng loại giao thức trong việc bảo vệ tính nhất quán của chuỗi khối.
- Câu 5. Vì sao Blockchain có thể đảm bảo tính toàn vẹn dữ liệu? Trình bày vai trò của thuật toán SHA-256 trong việc mã hóa và xác thực giao dịch.
- Câu 6. Smart Contract là gì? Trong thực tế, hợp đồng thông minh có thể được sử dụng trong những lĩnh vực nào? Lấy ví dụ minh họa.
- Câu 7. Khi triển khai hợp đồng thông minh bằng Python với thư viện Web3.py, cần thực hiện những bước kỹ thuật nào? Nêu các bước chính.
- Câu 8. Trong một dự án Khoa học dữ liệu có yêu cầu xác minh nguồn gốc dữ liệu đầu vào, Anh/Chị sẽ tích hợp Blockchain như thế nào? Trình bày đề xuất quy trình ứng dụng cụ thể.
- Câu 9. Một sinh viên đã viết thành công một Smart Contract để lưu trữ thông điệp chuỗi văn bản, nhưng khi triển khai gặp lỗi "invalid opcode". Theo Anh/Chị, nguyên nhân có thể là gì? Hãy đề xuất các bước kiểm tra khắc phục.

BÀI TẬP VẬN DỤNG

Bài 1. Viết chương trình Python để mô phỏng một chuỗi Blockchain gồm 3 khối. Mỗi khối chứa một thông điệp và mã băm (hash) của khối trước.

Gợi ý:

- Sử dụng `hashlib.sha256()` để tạo mã băm (hash).
- Mỗi khối gồm: thông điệp, thời gian tạo, mã băm (hash) khối trước.
- In ra chuỗi khối và thử thay đổi thông tin khối giữa → quan sát hậu quả.

Bài 2. Viết chương trình Python nhập một chuỗi bất kỳ (ví dụ: "Học phần Mạng máy tính") và tính mã băm (hash) bằng SHA-256. Sau đó, thay đổi một ký tự và quan sát sự thay đổi mã băm (hash).

Gợi ý:

- Dùng thư viện `hashlib`.
- Trình bày nhận xét: chỉ một thay đổi nhỏ → mã băm (hash) thay đổi toàn bộ.

Bài 3. Viết và triển khai Smart Contract đơn giản bằng Python

Mục tiêu: Làm quen với Smart Contract và công cụ `Web3.py` trong phạm vi đơn giản nhất.

Yêu cầu:

- Dùng mã Solidity đã trình bày trong chương 4 để viết một hợp đồng lưu thông điệp.
- Biên dịch hợp đồng bằng `solcx`.
- Dùng `Web3.py` kết nối Ganache và triển khai hợp đồng.

Gửi thông điệp mới và đọc lại thông điệp đã lưu.

Vi sao Bitcoin / Ethereum có giá trị?



Tính khan hiếm

- Tổng cung giới hạn hoặc giảm phát



Phi tập trung và tin cậy

- Giao dịch công khai, bảo mật



Tính hữu dụng

- Phương tiện thanh toán, vận hành ứng dụng



Hiệu ứng mạng

- Càng nhiều người dùng, giá trị càng tăng