ETHEREUM: MỘT SỔ CÁI GIAO DỊCH TỔNG QUÁT AN TOÀN PHI TẬP TRUNG

PARIS VERSION 2f36cf0 – 2024-01-12

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

Abstract. Mô hình chuỗi khối khi kết hợp với giao dịch được bảo vệ bằng mật mã đã chứng minh tính hữu ích của nó thông qua nhiều dự án, trong đó Bitcoin là một trong những dự án đáng chú ý nhất. Mỗi dự án như vậy có thể được xem như một ứng dụng đơn giản trên một nguồn tài nguyên tính toán phi tập trung, nhưng là một máy tính đơn lẻ. Chúng ta có thể gọi mô hình này là một máy đơn với trạng thái giao dịch được chia sẻ.

Ethereum thực hiện mô hình này một cách tổng quát. Hơn nữa, nó cung cấp một loạt các nguồn tài nguyên như vậy, mỗi nguồn có một trạng thái và mã (code) hoạt động riêng biệt, nhưng có khả năng tương tác thông qua một khung gửi tin nhắn với (message-passing framework) các nguồn khác. Chúng ta thảo luận về thiết kế của nó, các vấn đề thực hiện, cơ hội mà nó mang lại và những thách thức tương lai mà chúng ta dự tính.

1. Giới thiệu

Với các kết nối internet phổ biến ở hầu hết các nơi trên thế giới, việc truyền thông tin toàn cầu đã trở nên cực kỳ rẻ. Các phong trào dựa trên công nghệ như Bitcoin đã thể hiện thông qua sức mạnh của các cơ chế mặc định, đồng thuận và sự tôn trọng tự nguyện của hợp đồng xã hội, rằng có thể sử dụng Internet để tạo ra một hệ thống chuyển đổi giá trị phi tập trung có thể được chia sẻ trên toàn thế giới và gần như miễn phí để sử dụng. Hệ thống này có thể được cho là một phiên bản rất chuyên dụng của một máy trạng thái dựa trên giao dịch, an toàn bằng mã hóa. Các hệ thống tiếp theo như Namecoin đã điều chỉnh "Ứng dụng tiền tệ" của công nghệ thành các ứng dụng khác, mặc dù đôi khi khá đơn giản.

Ethereum là một dự án có nỗ lực xây dựng công nghệ tổng quát; công nghệ mà trên đó có thể xây dựng tất cả các khái niệm máy trạng thái dựa trên giao dịch. Hơn nữa, nó nhằm mục tiêu cung cấp cho nhà phát triển cuối cùng một hệ thống tích hợp chặt chẽ từ đầu đến cuối để xây dựng phần mềm trên một mô hình tính toán chưa được khám phá cho đến nay: một đối tượng khung (framework) tính toán nhắn tin đáng tin cậy.

1.1. Các Yếu Tố Thúc Đẩy.

Dự án này có nhiều mục tiêu; một mục tiêu quan trọng là tạo điều kiện cho giao dịch giữa các cá nhân đồng ý mà không có phương tiện nào khác để tin tưởng lẫn nhau. Điều này có thể do sự chia cách địa lý, khó khăn trong việc giao tiếp, hoặc có thể do sự không tương thích, không đủ năng lực, không sẵn lòng, chi phí cao, không chắc chắn, bất tiện hoặc tham nhũng của các hệ thống pháp luật hiện tại. Bằng cách xác định một hệ thống thay đổi trạng thái thông qua một ngôn ngữ phong phú và không mơ hồ, và hơn nữa, thiết kế một hệ thống sao cho chúng ta có thể hợp lý kỳ vọng rằng một thỏa thuận sẽ được thực hiện tự động, chúng ta có thể cung cấp một phương tiện để đạt được mục tiêu này.

Các giao dịch trong hệ thống được đề xuất này sẽ có một số thuộc tính không thường xuất hiện trong thế giới thực. Tính liêm khiết của sự phán xử, thường khó tìm thấy, đến một cách tự nhiên từ một trình thông dịch thuật toán công tâm. Sự minh bạch, hoặc khả năng nhìn thấy chính xác cách một trạng thái hoặc quyết định được hình thành thông qua nhật ký giao dịch và các quy tắc hoặc mã chỉ thị, không bao giờ xảy ra hoàn hảo trong các hệ thống dựa trên con người vì ngôn ngữ tự nhiên thường mơ hồ, thông tin thường thiếu và định kiến cũng khó lòng thay đổi.

Nhìn chung, chúng ta muốn cung cấp một hệ thống sao cho người dùng có thể được đảm bảo rằng bất kể các cá nhân, hệ thống hoặc tổ chức khác mà họ tương tác, họ có thể làm như vậy với lòng tin tuyệt đối về kết quả có thể xảy ra và kết quả đó có thể xảy ra như thế nào.

1.2. Công Trình Trước Đây.

Buterin [2013] lần đầu tiên đã đề xuất ý tưởng cơ bản của công trình này vào cuối tháng 11 năm 2013. Mặc dù đã phát triển theo nhiều cách, nhưng chức năng chính của một chuỗi khối với ngôn ngữ Turing hoàn chỉnh và khả năng lưu trữ giữa các giao dịch hiệu quả vô hạn vẫn không thay đổi.

Dwork and Naor [1992] cung cấp công trình đầu tiên về việc ứng dụng của một bằng chứng mật mã của chi phí tính toán "bằng chứng làm việc" ("proof-of-work") như một phương tiện truyền tải một tín hiệu giá trị qua Internet. Tín hiệu giá trị được sử dụng ở đây như một cơ chế ngăn chặn thư rác thay vì một loại tiền tệ nào đó, nhưng quan trọng là đã chứng minh tiềm năng của một kênh dữ liệu cơ bản để truyền tải một *tín hiệu kinh tế mạnh mẽ*, cho phép người nhận đưa ra một khẳng định vật lý mà không cần phải dựa vào *lòng tin*. Back [2002] sau đó tạo ra một hệ thống theo một hướng tương tự.

Ví dụ đầu tiên về việc sử dụng bằng chứng làm việc như một tín hiệu kinh tế mạnh mẽ để bảo vệ một đơn vị tiền tệ đã được thực hiện bởi Vishnumurthy et al. [2003]. Trong trường hợp này, token được sử dụng để kiểm soát giao dịch tệp ngang hàng, cung cấp khả năng cho "người tiêu dùng" ("consumers") thực hiện thanh toán nhỏ đối với "nhà cung cấp" ("suppliers") cho dịch vụ của họ. Mô hình bảo mật được cung cấp bởi "bằng chứng làm việc" được tăng cường với chữ ký số và một sổ cái để đảm bảo rằng hồ sơ lịch sử không thể bị làm hỏng và các diễn viên ác ý không thể giả mạo thanh toán hoặc phàn nàn một cách bất công về việc cung cấp dịch vụ. Năm năm sau, Nakamoto [2008] giới thiệu một "bằng chứng làm việc được bảo vệ" ("proof-of-work-secured"), có phạm vi rộng hơn một chút. Thành quả của dự án này, Bitcoin, đã trở thành sổ cái

giao dịch phi tập trung toàn cầu đầu tiên được áp dụng rộng rãi.

Các dự án khác được xây dựng dựa trên thành công của Bitcoin; Các "alt-coin" đã giới thiệu nhiều loại tiền tệ khác thông qua thay đổi giao thức. Một số nổi tiếng nhất là Litecoin và Primecoin, được thảo luận bởi Sprankel [2013]. Các dự án khác đã tìm cách lấy cơ chế nội dung giá trị cốt lõi của giao thức và tái sử dụng nó; những thảo luận của Aron [2012] là ví dụ, dự án Namecoin nhằm cung cấp một hệ thống phân giải tên phi tập trung.

Các dự án khác vẫn nhằm mục đích xây dựng trên chính mạng Bitcoin, tận dụng lượng giá trị lớn được đặt trong hệ thống và số lượng lớn tính toán đi vào cơ chế đồng thuận. Dự án Mastercoin, được đề xuất lần đầu tiên bởi Willett [2013], nhằm mục đích xây dựng một giao thức phong phú hơn liên quan đến nhiều tính năng cấp cao bổ sung trên đầu giao thức Bitcoin thông qua việc sử dụng một số phần phụ trợ cho giao thức cốt lõi. Dự án Coloured Coins, được đề xuất bởi Rosenfeld et al. [2012], có một chiến lược tương tự nhưng đơn giản hơn, tô điểm cho các quy tắc của một giao dịch để phá vỡ tính thay thế được của loại tiền cơ bản của Bitcoin và cho phép tạo và theo dõi các mã thông báo thông qua một phần mềm đặc biệt "chroma-wallet" nhận biết giao thức.

Thêm công việc khác đã được thực hiện trong lĩnh vực này với việc từ bỏ cơ sở phi tập trung; Ripple, được thảo luận bởi Boutellier and Heinzen [2014], đã tìm cách tạo ra một hệ thống "liên minh" ("federated") để trao đổi tiền tệ, tạo ra một hệ thống tài chính sạch mới một cách hiệu quả. Nó đã chứng minh rằng có thể đạt được những cải tiến hiệu suất lớn nếu giả định về phi tập trung được từ bỏ.

Công việc sớm về hợp đồng thông minh đã được thực hiện bởi Szabo [1997] và Miller [1997]. Vào khoảng những năm 1990, rõ ràng rằng việc thực thi thỏa thuận thông qua thuật toán có thể trở thành một sức mạnh có ý nghĩa trong sự hợp tác của con người. Mặc dù chưa có hệ thống cụ thể nào được đề xuất để thực hiện một hệ thống như vậy, nhưng người ta đã đề xuất rằng tương lai của pháp luật sẽ bị ảnh hưởng nặng nề bởi các hệ thống đó. Dưới góc nhìn này, Ethereum có thể được coi là một triển khai tổng quát của một hệ thống *luật mật mã* (*crypto-law*).

Để biết danh sách các thuật ngữ được sử dụng trong bài viết này, hãy tham khảo Phụ lục A.

2. Mô hình chuỗi khối

Ethereum, nhìn chung, có thể được xem như một máy trạng thái dựa trên giao dịch: chúng ta bắt đầu với một trạng thái khởi điểm (genesis state) và thực hiện các giao dịch một cách tăng dần để biến nó thành một trạng thái hiện tại nào đó. Đây chính là trạng thái hiện tại mà chúng ta chấp nhận là "phiên bản" chuẩn của thế giới Ethereum. Trạng thái có thể bao gồm thông tin như số dư tài khoản, uy tín, các thỏa thuận tin tưởng, dữ liệu liên quan đến thông tin của thế giới vật lý; nói một cách ngắn gọn, bất cứ điều gì hiện nay có thể được biểu diễn bởi máy tính đều được chấp nhận. Giao dịch do đó đại diện cho một cung đường hợp lệ giữa hai trạng thái; phần 'hợp lệ' là quan trọng—thay đổi trạng thái không hợp lệ tồn tại nhiều hơn nhiều so với những thay đổi trạng thái hợp lệ. Những thay đổi trạng thái không hợp lệ có thể là những điều như giảm số dư một tài khoản mà không có sự gia tăng tương đương và ngược lại ở đâu đó khác. Một chuyển đổi trạng thái hợp lệ là một chuyển đổi mà xuất hiện thông qua một giao dịch. Quy ước:

$$\boldsymbol{\sigma}_{t+1} \equiv \Upsilon(\boldsymbol{\sigma}_t, T) \tag{1}$$

trong đó Υ là hàm chuyển đổi trạng thái Ethereum. Trong Ethereum, Υ, cùng với $\boldsymbol{\sigma}$ mạnh hơn đáng kể so với bất kỳ hệ thống so sánh hiện có nào; Υ cho phép các thành phần thực hiện tính toán tùy ý, trong khi $\boldsymbol{\sigma}$ cho phép các thành phần lưu trữ trạng thái tùy ý giữa các giao dịch.

Giao dịch được tổng hợp thành các khối (blocks); các khối được xích lại với nhau (chained together) bằng cách sử dụng hàm băm mật mã như một phương tiện tham chiếu. Các khối hoạt động như một nhật ký, ghi lại một loạt các giao dịch cùng với khối trước và một định danh cho trạng thái cuối cùng (mặc dù không lưu trữ trạng thái cuối cùng là bản thân nó — điều đó sẽ quá lớn).

Một cách quy ước, chúng ta mở rộng thành:

$$\boldsymbol{\sigma}_{t+1} \equiv \Pi(\boldsymbol{\sigma}_t, B) \tag{2}$$

$$B \equiv (..., (T_0, T_1, ...), ...) \tag{3}$$

$$\Pi(\boldsymbol{\sigma}, B) \equiv \Upsilon(\Upsilon(\boldsymbol{\sigma}, T_0), T_1)... \tag{4}$$

Trong đó B là khối này, bao gồm một loạt các giao dịch bên cạnh một số thành phần khác và Π là hàm chuyển đổi trạng thái cấp khối (block-level).

Đây là cơ sở của mô hình chuỗi khối, một mô hình tạo thành xương sống không chỉ Ethereum, mà tất cả các hệ thống giao dịch dựa trên sự đồng thuận phi tập trung cho đến nay.

2.1. Giá trị (Value).

Để khuyến khích việc tính toán trong mạng, cần có một phương pháp thống nhất để truyền giá trị. Để giải quyết vấn đề này, Ethereum có một đơn vị tiền tệ tích hợp, gọi là Ether, còn được biết đến với tên ETH và đôi khi được đề cập trong tiếng Anh cổ Ð. Đơn vị con nhỏ nhất của Ether, và là đơn vị trong đó tất cả các giá trị tiền tệ được tính trên số nguyên, là Wei. Một Ether được định nghĩa là 10^{18} Wei. Còn các đơn vị con khác của Ether:

Số nhân	Tên
10^0	Wei
10^{12}	Szabo
10^{15}	Finney
10^{18}	Ether

Trong toàn bộ công việc hiện tại, bất kỳ tham chiếu nào đến giá trị, trong ngữ cảnh của Ether, tiền tệ, số dư hoặc thanh toán, đều nên được giả định là được tính bằng Wei.

2.2. Lịch sử Nào (Which History)?

Vì hệ thống được phân tán và tất cả các bên đều có cơ hội để tạo ra một khối mới trên một khối cũ tồn tại trước đó, cấu trúc kết quả là một cây khối. Để đạt được sự nhất quán về đường đi nào, từ gốc (khối khởi điểm (genesis)) đến lá (khối chứa giao dịch mới nhất) qua cấu trúc cây này, được biết đến như là chuỗi khối (blockchain), phải có một kế hoạch được thống nhất. Nếu bao giờ có sự không đồng ý giữa các nút về đường đi từ gốc đến lá xuống cây khối nào là chuỗi khối 'tốt nhất', thì một *sự rẻ nhánh* (*fork*) xảy ra.

Điều này có nghĩa là vượt qua một điểm thời gian nhất định (khối), nhiều trạng thái của hệ thống có thể tồn tại song song: một số nút (nodes) tin rằng một khối chứa các giao dịch chính thức, các nút khác tin rằng một khối khác là chính thức, có thể chứa các giao dịch hoàn toàn khác biệt hoặc không tương thích. Điều này cần tránh bằng mọi

giá vì sự không chắc chắn sẽ có thể đặt dấu chấm hết vào lòng tin trong toàn bộ hệ thống.

Từ *hard fork Paris* trở đi, việc đạt được sự đồng thuận trên các khối mới được quản lý bằng một giao thức được gọi là *Beacon Chain*. Nó được biết đến như là *lớp đồng thuận (consensus layer)* của Ethereum, và nó đặt ra các quy tắc để xác định lịch sử chính thức của các khối Ethereum. Tài liệu này mô tả *lớp thực thi (execution layer)* của Ethereum. Lớp thực thi đặt ra các quy tắc để tương tác và cập nhật trạng thái của máy ảo Ethereum. Lớp đồng thuận được mô tả chi tiết hơn trong các đặc tả đồng thuận. Cách lớp nhất quán được sử dụng để xác định trạng thái chính thức của Ethereum được thảo luận trong phần 11.

Có nhiều phiên bản của Ethereum, vì giao thức đã trải qua nhiều bản cập nhật. Các bản cập nhật này có thể được chỉ định xảy ra:

- tại một số khối cụ thể trong trường hợp của các bản cập nhật trước *Paris*.
- sau khi đạt đến một *tổng độ khó cuối cùng (terminal total difficulty)* trong trường hợp của bản cập nhật *Paris*, hoặc
- tại một dấu thời gian khối (block timestamp) cụ thể trong trường hợp của các bản cập nhật sau *Paris*.

Tài liệu này mô tả phiên bản *Paris*.

Để theo dõi lịch sử của một đường dẫn, người đọc cần tham khảo nhiều phiên bản của tài liệu này. Dưới đây là số khối của các bản cập nhật giao thức trên mạng chính Ethereum:[1]

Tên	Số khối đầu tiên
$F_{\text{Homestead}}$	1150000
$F_{\text{TangerineWhistle}}$	2463000
$F_{\text{SpuriousDragon}}$	2675000
$F_{\text{Byzantium}}$	4370000
$F_{\text{Constantinople}}$	7280000
$F_{\text{Petersburg}}$	7280000
F_{Istanbul}	9069000
$F_{\text{MuirGlacier}}$	9200000
F_{Berlin}	12244000
F_{London}	12965000
$F_{\text{ArrowGlacier}}$	13773000
$F_{\text{GrayGlacier}}$	15050000
F_{Paris}	15537394

Đôi khi các bên không đồng ý với một thay đổi giao thức, và một sự rẽ nhánh (fork) vĩnh viễn xảy ra. Để phân biệt giữa các chuỗi khối rời rạc, EIP-155 được Buterin [2016] giới thiệu khái niệm về Chain ID, được ký hiệu là β. Đối với mạng chính Ethereum

$$\beta = 1 \tag{5}$$

3. Quy ước

Chúng ta sử dụng nhiều quy ước chữ viết để biểu diễn hình thức, một số trong số đó khá đặc biệt cho công việc hiện tại:

Hai tập giá trị trạng thái có cấu trúc cao, 'top-level', được ký hiệu bằng chữ viết thường viết đậm (bold lowercase) của chữ cái Hy Lạp. Chúng thuộc về trạng thái thế giới, được ký hiệu là $\boldsymbol{\sigma}$ (hoặc biến thể tương tự) và trạng thái máy, được ký hiệu là $\boldsymbol{\mu}$.

Các hàm thi hành trên các giá trị có cấu trúc cao được ký hiệu bằng một chữ cái Hy Lạp viết hoa (uppercase), ví dụ như Υ, hàm chuyển trạng thái Ethereum.

Đối với hầu hết các hàm, chúng ta sử dụng một chữ cái viết hoa, ví dụ như C, hàm chi phí chung. Các biến thể chuyên sâu của chúng có thể được ký hiệu bằng chữ dưới, ví dụ như C_{SSTORE}, hàm chi phí cho việc thi hành SSTORE. Đối với các hàm chuyên sâu và có thể được định nghĩa bên ngoài, chúng ta có thể sử dụng văn bản kiểu máy đánh chữ, ví dụ như hàm băm Keccak-256 (theo phiên bản 3 của đề xuất thắng cuộc trong cuộc thi SHA-3 của Bertoni et al. [2011], chứ không phải là đặc tả SHA-3 sau cùng), được ký hiệu là `KEC` (và thường được gọi là Keccak thuần túy). Tương tự, `KEC512` đề cập đến hàm băm Keccak-512.

Bộ dữ liệu (Tuples) thường được ký hiệu bằng một chữ cái viết hoa, ví dụ như T, được sử dụng để biểu thị một giao dịch Ethereum. Ký hiệu này có thể, nếu được định nghĩa một cách phù hợp, được chú thích bằng chữ dưới để chỉ đến một thành phần cụ thể, ví dụ như T_{n}, biểu thị nonce của giao dịch đó. Dạng của chỉ số được sử dụng để biểu thị loại; ví dụ, chữ cái viết hoa dưới chân thì tham chiếu đến bộ ba giá trị có các thành phần có thể được đặt chỉ số.

Các giá tị vô hướng (scalars) và các dãy byte có kích thước cố định (hoặc, một cách đồng nghĩa, các mảng) được ký hiệu bằng một chữ cái viết thường, ví dụ như n được sử dụng trong tài liệu để biểu thị một nonce giao dịch. Các giá trị có ý nghĩa đặc biệt có thể được viết bằng chữ cái Hy Lạp, ví dụ như δ, số lượng các mục được yêu cầu trên ngăn xếp cho một sự thi hành cụ thể.

Dãy có độ dài tùy ý thường được biểu thị bằng một chữ cái viết thường đậm, ví dụ như $\mathbf{o}$ được sử dụng để biểu thị dãy byte đầu ra của một cuộc gọi tin nhắn. Đối với các giá trị quan trọng, có thể sử dụng một chữ cái viết hoa đậm.

Trong toàn bộ, chúng ta giả định rằng các giá trị cố định là số nguyên không âm và do đó thuộc tập hợp $\mathbb{N}$. Tập hợp tất cả các dãy byte là $\mathbb{B}$, được định nghĩa chi tiết trong Phụ lục B. Nếu một tập hợp các dãy được hạn chế đến những dãy có chiều dài cụ thể, nó được biểu thị bằng một chỉ số dưới, vì vậy tập hợp tất cả các dãy byte có chiều dài 32 được đặt tên là $\mathbb{B}_{32}$ và tập hợp tất cả các số nguyên không âm nhỏ hơn 2^{256} được đặt tên là $\mathbb{N}_{256}$. Điều này được định nghĩa chi tiết trong phần 4.3.

Dấu ngoặc vuông được sử dụng để lập chỉ mục và tham chiếu đến các thành phần hoặc các phần con của các dãy, ví dụ như $\boldsymbol{\mu}_{\mathbf{s}}[0]$ biểu thị mục đầu tiên trên ngăn xếp của máy. Đối với các phần con, dấu ba chấm được sử dụng để chỉ định phạm vi dự định, để bao gồm các phần tử ở cả hai đỉnh, ví dụ như $\boldsymbol{\mu}_{\mathbf{m}}[0..31]$ biểu thị 32 mục đầu tiên của bộ nhớ của máy.

Trong trường hợp của trạng thái toàn cầu $\boldsymbol{\sigma}$, một dãy các tài khoản, chúng ta sử dụng dấu ngoặc vuông để tham chiếu đến một tài khoản cụ thể.

Khi xem xét các biến thể của các giá trị hiện tại, chúng ta tuân theo quy tắc rằng trong một phạm vi xác định, nếu chúng ta giả sử rằng giá trị 'đầu vào' không được sửa đổi được ký hiệu bằng biểu tượng $\square$ thì giá trị đã được sửa đổi và có thể sử dụng được ký hiệu là $\square'$, và giá trị trung gian sẽ là $\square^*$, $\square^{**}$, và các giá trị tương tự. Trong những trường hợp cụ thể, đặc biệt là để tối ưu hóa tính đọc và chỉ khi không mơ hồ về ý nghĩa, chúng ta có thể sử

[1] Lưu ý rằng trong khi fork Paris được kích hoạt tại khối 15,537,394, nhưng điều kích hoạt không phải là số khối, mà là khi đạt được một *tổng độ khó* cụ thể. Chi tiết về điều kích hoạt của hard fork Paris được thảo luận chi tiết hơn trong phần 10.

dụng chữ số và chữ cái để ký hiệu giá trị trung gian, đặc biệt là những giá trị đặc biệt.

Khi xem xét việc sử dụng các hàm hiện có, với một hàm f nào đó, hàm f^* biểu thị một phiên bản tương tự, thực hiện theo từng phần tử của dãy thay vì giữa các dãy. Định nghĩa chi tiết được thực hiện trong phần 4.3.

Chúng ta định nghĩa một số hàm hữu ích trong toàn bộ tài liệu. Một trong những hàm phổ biến là ℓ, mà có giá trị là phần tử cuối cùng trong dãy đã cho:

$$\ell(\mathbf{x}) \equiv \mathbf{x}[\|\mathbf{x}\| - 1] \tag{6}$$

4. Các khối, Trạng thái và Các giao dịch

Sau khi giới thiệu các khái niệm cơ bản của Ethereum, chúng ta sẽ thảo luận chi tiết hơn về ý nghĩa của một giao dịch, một khối và trạng thái.

4.1. Trạng thái thế giới (World State).

Trạng thái thế giới (*state*) là một ánh xạ giữa địa chỉ (nhận dạng 160-bit) và trạng thái tài khoản (một cấu trúc dữ liệu được chuỗi hóa theo RLP, xem Phụ lục B). Mặc dù không được lưu trữ trên chuỗi khối, giả định rằng bản triển khai sẽ duy trì ánh xạ này trong một cây Merkle Patricia được sửa đổi (*trie*, xem Phụ lục D). Trie yêu cầu một backend cơ sở dữ liệu đơn giản duy trì một ánh xạ từ mảng byte đến mảng byte; chúng ta đặt tên cho cơ sở dữ liệu cơ bản này là cơ sở dữ liệu trạng thái. Điều này mang lại nhiều lợi ích; đầu tiên, nút gốc (root node) của cấu trúc này phụ thuộc về mặt mật mã vào tất cả dữ liệu nội bộ và do đó băm của nó có thể được sử dụng như một danh tính an toàn cho toàn bộ trạng thái hệ thống. Thứ hai, với tính chất là một cấu trúc dữ liệu bất biến, nó cho phép lấy lại bất kỳ trạng thái trước đó (mà băm gốc của nó đã được biết) chỉ bằng cách thay đổi băm gốc một cách tương ứng. Vì chúng ta lưu trữ tất cả các băm gốc như vậy trong chuỗi khối, chúng ta có thể dễ dàng quay lại các trạng thái cũ.

Trạng thái tài khoản, $\boldsymbol{\sigma}[a]$, bao gồm bốn trường sau:

- **nonce:** Một giá trị vô hướng (scalar) bằng số lần giao dịch được gửi từ địa chỉ này hoặc, trong trường hợp của các tài khoản có mã (code) liên quan, số lần tạo hợp đồng bởi tài khoản này. Đối với tài khoản có địa chỉ a trong trạng thái $\boldsymbol{\sigma}$, điều này sẽ được biểu diễn là $\boldsymbol{\sigma}[a]_{\text{n}}$.
- **balance:** Giá trị vô hướng bằng số Wei được sở hữu bởi địa chỉ này. Được biểu diễn là $\boldsymbol{\sigma}[a]_{\text{b}}$.
- **storageRoot:** Một băm 256-bit của nút gốc của cây Merkle Patricia mã hóa nội dung lưu trữ của tài khoản (một ánh xạ giữa các giá trị số nguyên 256-bit), được mã hóa vào trie như một ánh xạ từ băm Keccak 256-bit của các khóa số nguyên 256-bit đến các giá trị số nguyên 256-bit được mã hóa bằng RLP. Băm được biểu diễn là $\boldsymbol{\sigma}[a]_{\text{s}}$.
- **codeHash:** Băm của mã (code) EVM của tài khoản này—đây là mã được thực thi nếu địa chỉ này nhận một cuộc Gọi tin nhắn (message call). Tất cả các đoạn mã như vậy được chứa trong cơ sở dữ liệu trạng thái dưới các băm tương ứng của chúng để có thể được gọi lại sau này. Băm này được biểu diễn là $\boldsymbol{\sigma}[a]_{\text{c}}$, và mã (code) có thể được biểu diễn là $\mathbf{b}$, do đó $\texttt{KEC}(\mathbf{b}) = \boldsymbol{\sigma}[a]_{\text{c}}$.

Vì chúng ta thường muốn tham chiếu không phải đến băm gốc của trie mà đến bộ cặp khóa/giá trị cơ bản được lưu trữ bên trong, chúng ta định nghĩa một tương đương thuận tiện:

$$\texttt{TRIE}\big(L_{\text{I}}^*(\boldsymbol{\sigma}[a]_{\mathbf{s}})\big) \equiv \boldsymbol{\sigma}[a]_{\text{s}} \tag{7}$$

Hàm thu gọn cho tập hợp cặp khóa/giá trị trong trie, L_{I}^*, được định nghĩa như sự biến đổi từng phần của hàm cơ sở L_{I}, như sau:

$$L_{\text{I}}\big((k, v)\big) \equiv \big(\texttt{KEC}(k), \texttt{RLP}(v)\big) \tag{8}$$

trong đó:

$$k \in \mathbb{B}_{32} \quad \wedge \quad v \in \mathbb{N} \tag{9}$$

Cần hiểu rằng $\boldsymbol{\sigma}[a]_{\mathbf{s}}$ không phải là một thành phần 'vật lý' của tài khoản và không đóng góp vào việc biểu diễn sau này của nó.

Nếu trường **codeHash** là băm Keccak-256 của chuỗi rỗng, tức là $\boldsymbol{\sigma}[a]_{\text{c}} = \texttt{KEC}\big(()\big)$, thì nút đại diện cho một tài khoản đơn giản, đôi khi được gọi là một tài khoản "không phải hợp đồng".

Do đó, chúng ta có thể định nghĩa một hàm thu gọn trạng thái thế giới: L_{S}:

$$L_{\text{S}}(\boldsymbol{\sigma}) \equiv \{p(a) : \boldsymbol{\sigma}[a] \neq \varnothing\} \tag{10}$$

trong đó

$$p(a) \equiv \big(\texttt{KEC}(a), \texttt{RLP}\big((\boldsymbol{\sigma}[a]_{\text{n}}, \boldsymbol{\sigma}[a]_{\text{b}}, \boldsymbol{\sigma}[a]_{\text{s}}, \boldsymbol{\sigma}[a]_{\text{c}})\big)\big) \tag{11}$$

Hàm này, L_{S}, được sử dụng cùng với hàm trie để cung cấp một định danh ngắn (hash) của trạng thái thế giới. Chúng ta giả định:

$$\forall a : \boldsymbol{\sigma}[a] = \varnothing \ \vee \ (a \in \mathbb{B}_{20} \ \wedge \ v(\boldsymbol{\sigma}[a])) \tag{12}$$

trong đó v là hàm kiểm tra tính hợp lệ của tài khoản:

$$v(x) \equiv x_{\text{n}} \in \mathbb{N}_{256} \wedge x_{\text{b}} \in \mathbb{N}_{256} \wedge x_{\text{s}} \in \mathbb{B}_{32} \wedge x_{\text{c}} \in \mathbb{B}_{32} \tag{13}$$

Một tài khoản được coi là *rỗng (empty)* khi nó không có mã (code), nonce bằng 0 và số dư (balance) bằng 0:

$$\texttt{EMPTY}(\boldsymbol{\sigma}, a) \quad \equiv \quad \boldsymbol{\sigma}[a]_{\text{c}} = \texttt{KEC}\big(()\big) \wedge \boldsymbol{\sigma}[a]_{\text{n}} = 0 \wedge \boldsymbol{\sigma}[a]_{\text{b}} = 0 \tag{14}$$

Ngay cả các hợp đồng được gọi có thể có trạng thái tài khoản rỗng. Điều này là do trạng thái tài khoản của chúng thường không chứa mã mô tả hành vi của chúng.

Một tài khoản được coi là *dead* khi trạng thái tài khoản của nó không tồn tại hoặc rỗng:

$$\texttt{DEAD}(\boldsymbol{\sigma}, a) \quad \equiv \quad \boldsymbol{\sigma}[a] = \varnothing \vee \texttt{EMPTY}(\boldsymbol{\sigma}, a) \tag{15}$$

4.2. Giao dịch (The Transaction).

Một giao dịch (ký hiệu T) là một chỉ thị đơn được ký điện tử được tạo ra bởi một tác nhân bên ngoài phạm vi của Ethereum. Người gửi của một giao dịch không thể là một hợp đồng. Mặc dù người ta cho rằng tác nhân bên ngoài cuối cùng về bản chất sẽ là con người, các công cụ phần mềm sẽ được sử dụng trong quá trình xây dựng và phổ biến[2]. EIP-2718 của Zoltu [2020] giới thiệu khái niệm về các loại giao dịch khác nhau. Kể từ phiên bản giao thức *London*, có ba loại giao dịch: 0 (legacy), 1 (EIP-2930 của Buterin and Swende [2020b]), và 2 (EIP-1559 của Buterin et al. [2019]). Hơn

[2]Chú ý rằng, các 'công cụ' như vậy cuối cùng có thể trở nên xa lạ so với sự khởi đầu dựa trên con người của chúng — hoặc con người có thể trở nên trung lập về mặt nguyên nhân đến mức có thể có một điểm nơi chúng có thể được xem xét đúng là các đại lý tự động. Ví dụ, các hợp đồng có thể cung cấp thưởng cho con người về việc gửi giao dịch để khởi động thực thi của chúng.

nữa, có hai loại con của giao dịch: những giao dịch dẫn đến cuộc gọi tin nhắn và những giao dịch dẫn đến việc tạo ra các tài khoản mới có mã (code) liên quan (gọi một cách không chính thức là 'tạo hợp đồng'). Tất cả các loại giao dịch đều chỉ định một số trường chung:

type: Loại giao dịch theo EIP-2718; quy ước là T_{x}.

nonce: Giá trị vô hướng bằng số lượng giao dịch được gửi bởi người gửi; quy ước là T_{n}.

gasLimit: Giá trị vô hướng bằng số lượng gas tối đa nên được sử dụng trong việc thực thi giao dịch này. Đây là khoản thanh toán trước, trước khi thực hiện bất kỳ tính toán nào và không thể tăng sau này; quy ước là T_{g}.

to: Địa chỉ 160-bit của người nhận cuộc gọi tin nhắn hoặc, đối với một giao dịch tạo hợp đồng, $\varnothing$, được sử dụng ở đây để chỉ định phần tử duy nhất của $\mathbb{B}_0$; quy ước là T_{t}.

value: Giá trị vô hướng bằng số lượng Wei sẽ được chuyển đến người nhận cuộc gọi tin nhắn hoặc, trong trường hợp tạo hợp đồng, như một phúc lợi cho tài khoản mới được tạo; quy ước là T_{v}.

r, s: Các giá trị tương ứng với chữ ký của giao dịch và được sử dụng để xác định người gửi của giao dịch; quy ước là T_{r} và T_{s}. Điều này được mở rộng trong Phụ lục F.

Các giao dịch EIP-2930 (loại 1) và EIP-1559 (loại 2) cũng có:

accessList: Danh sách các mục truy cập để tiếp cận "nóng"; quy ước là $T_{\mathbf{A}}$. Mỗi mục danh sách truy cập E là một bộ dữ liệu (tuple) gồm địa chỉ tài khoản và một danh sách các khóa lưu trữ: $E \equiv (E_{\mathrm{a}}, E_{\mathbf{s}})$.

chainId: Chain ID; quy ước là T_{c}. Phải bằng với Chain ID của mạng β.

yParity: Chẵn lẻ của chữ ký; quy ước là T_{y}.

Các giao dịch Legacy không có **accessList** ($T_{\mathbf{A}} = ()$), trong khi **chainId** và **yParity** cho giao dịch Legacy được kết hợp thành một giá trị duy nhất:

w: Một giá trị vô hướng mã hóa chẵn lẻ Y và có thể là Chain ID; quy ước là T_{w}. $T_{\mathrm{w}} = 27 + T_{\mathrm{y}}$ hoặc $T_{\mathrm{w}} = 2\beta + 35 + T_{\mathrm{y}}$ (xem EIP-155 của Buterin [2016]).

Có sự khác biệt trong cách giá gas chấp nhận được được chỉ định trong các giao dịch loại 2 so với các giao dịch loại 0 và loại 1. Các giao dịch loại 2 tận dụng tốt hơn các cải tiến thị trường gas được giới thiệu trong EIP-1559 bằng cách giới hạn một cách rõ ràng *phí ưu tiên (priority fee)*[3] mà người gửi phải trả. Các giao dịch loại 2 có hai trường sau liên quan đến gas:

maxFeePerGas: Một giá trị vô hướng bằng số Wei tối đa phải trả cho mỗi đơn vị *gas* cho tất cả các chi phí tính toán phát sinh do thực hiện giao dịch này; quy ước là T_{m}.

maxPriorityFeePerGas: Một giá trị vô hướng bằng số Wei tối đa phải trả cho người nhận phí của khối như một động viên để xử lý giao dịch; quy ước là T_{f}.

Ngược lại, các giao dịch loại 0 và loại 1 chỉ định giá gas dưới dạng một giá trị duy nhất:

gasPrice: Một giá trị vô hướng bằng số Wei phải trả cho mỗi đơn vị *gas* cho tất cả các chi phí tính toán phát sinh do thực hiện giao dịch này; quy ước là T_{p}.[4]

Ngoài ra, một giao dịch tạo hợp đồng (bất kể loại giao dịch) chứa:

init: Một mảng byte không giới hạn, chỉ định mã EVM (EVM-code) cho quy trình khởi tạo tài khoản, quy ước là $T_{\mathbf{i}}$.

init là một đoạn mã EVM; nó trả về **body**, một đoạn mã thứ hai thực thi mỗi khi tài khoản nhận một cuộc gọi tin nhắn (dù thông qua một giao dịch hoặc do thực thi nội bộ của mã code). **init** chỉ thực thi một lần duy nhất khi tạo tài khoản và ngay sau đó sẽ bị loại bỏ.

Ngược lại, một giao dịch cuộc gọi tin nhắn chứa:

data: Một mảng byte không giới hạn kích thước, chỉ định dữ liệu đầu vào của cuộc gọi tin nhắn, quy ước là $T_{\mathbf{d}}$.

Phụ lục F chỉ định hàm S, có tác dụng ánh xạ giao dịch đến người gửi, và diễn ra thông qua ECDSA của đường cong SECP-256k1, sử dụng hash của giao dịch (trừ ba trường chữ ký cuối cùng) như là dữ liệu để ký. Để đại diện, chúng ta đơn giản khẳng định rằng người gửi của một giao dịch cụ thể T có thể được biểu diễn bằng $S(T)$.

(16)
$$L_{\mathrm{T}}(T) \equiv \begin{cases} (T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, \mathbf{p}, T_{\mathrm{w}}, T_{\mathrm{r}}, T_{\mathrm{s}}) & \text{nếu } T_{\mathrm{x}} = 0 \\ (T_{\mathrm{c}}, T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, \mathbf{p}, T_{\mathbf{A}}, T_{\mathrm{y}}, T_{\mathrm{r}}, T_{\mathrm{s}}) & \text{nếu } T_{\mathrm{x}} = 1 \\ (T_{\mathrm{c}}, T_{\mathrm{n}}, T_{\mathrm{f}}, T_{\mathrm{m}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, \mathbf{p}, T_{\mathbf{A}}, T_{\mathrm{y}}, T_{\mathrm{r}}, T_{\mathrm{s}}) & \text{nếu } T_{\mathrm{x}} = 2 \end{cases}$$

trong đó

(17)
$$\mathbf{p} \equiv \begin{cases} T_{\mathbf{i}} & \text{nếu } T_{\mathrm{t}} = \varnothing \\ T_{\mathbf{d}} & \text{ngược lại} \end{cases}$$

Ở đây, chúng ta giả định rằng tất cả các thành phần đều được thông dịch bởi RLP như là các giá trị số nguyên, ngoại trừ danh sách truy cập $T_{\mathbf{A}}$ và các mảng byte có độ dài tùy ý $T_{\mathbf{i}}$ và $T_{\mathbf{d}}$.

(18)
$$\begin{array}{lllllll} T_{\mathrm{x}} \in \{0,1,2\} & \wedge & T_{\mathrm{c}} = \beta & \wedge & T_{\mathrm{n}} \in \mathbb{N}_{256} & \wedge \\ T_{\mathrm{p}} \in \mathbb{N}_{256} & \wedge & T_{\mathrm{g}} \in \mathbb{N}_{256} & \wedge & T_{\mathrm{v}} \in \mathbb{N}_{256} & \wedge \\ T_{\mathrm{w}} \in \mathbb{N}_{256} & \wedge & T_{\mathrm{r}} \in \mathbb{N}_{256} & \wedge & T_{\mathrm{s}} \in \mathbb{N}_{256} & \wedge \\ T_{\mathrm{y}} \in \mathbb{N}_{1} & \wedge & T_{\mathbf{d}} \in \mathbb{B} & \wedge & T_{\mathbf{i}} \in \mathbb{B} & \wedge \\ T_{\mathrm{m}} \in \mathbb{N}_{256} & \wedge & T_{\mathrm{f}} \in \mathbb{N}_{256} & & & \end{array}$$

trong đó

(19)
$$\mathbb{N}_{\mathrm{n}} = \{P : P \in \mathbb{N} \wedge P < 2^n\}$$

Băm địa chỉ $T_{\mathbf{t}}$ có một chút khác biệt: nó có thể là một băm địa chỉ 20 byte hoặc, trong trường hợp là một giao dịch tạo hợp đồng (và do đó quy ước bằng $\varnothing$), nó là chuỗi byte trống của RLP và là phần tử của $\mathbb{B}_0$:

(20)
$$T_{\mathbf{t}} \in \begin{cases} \mathbb{B}_{20} & \text{nếu} \quad T_{\mathrm{t}} \neq \varnothing \\ \mathbb{B}_0 & \text{ngược lại} \end{cases}$$

[3] *Phí ưu tiên* được thảo luận chi tiết hơn trong các phần 5 và 6.

[4] Các giao dịch loại 0 và loại 1 sẽ có cùng hành vi giá gas như một giao dịch loại 2 với T_{m} và T_{f} được đặt thành giá trị của T_{p}.

4.3. **Khối (The Block).** Khối trong Ethereum là tổ hợp các thông tin liên quan (được biết đến là *header* của khối), H, cùng với thông tin tương ứng với các giao dịch bao gồm trong khối, $\mathbf{T}$, và một thuộc tính $\mathbf{U}$ hiện tại đã bị loại bỏ, trước *Paris* hard fork, nó chứa các header của các khối mà cha của chúng bằng với cha của cha của khối hiện tại (các khối này được biết đến với tên gọi *ommers*[5]). Header của khối chứa một số thông tin:

parentHash: Hash Keccak 256-bit của header của khối cha, toàn bộ; quy ước là H_{p}.

ommersHash: Một trường hash 256-bit bây giờ đã bị loại bỏ do sự thay thế của đồng thuận bằng chứng làm việc. Giờ đây nó là một hằng số, $\texttt{KEC}(\texttt{RLP}(()))$; quy ước là H_{o}.

beneficiary: Địa chỉ 160-bit mà phí ưu tiên từ khối này sẽ được chuyển đến; quy ước là H_{c}.

stateRoot: Hash Keccak 256-bit của nút gốc của cây trạng thái (state trie), sau khi tất cả các giao dịch được thực thi và các sự hoàn thành (finalisations) được áp dụng; quy ước là H_{r}.

transactionsRoot: Hash Keccak 256-bit của nút gốc của cấu trúc cây (trie) với mỗi giao dịch được điền vào trong phần danh sách giao dịch của khối; quy ước là H_{t}.

receiptsRoot: Hash Keccak 256-bit của nút gốc của cấu trúc cây (trie) với các biên nhận của mỗi giao dịch được điền vào trong phần danh sách giao dịch của khối; quy ước là H_{e}.

logsBloom: Bộ lọc Bloom được tạo ra từ thông tin có thể lập chỉ mục (địa chỉ và chủ đề (logger address and log topics)) chứa trong mỗi mục log từ biên nhận của mỗi giao dịch trong phần danh sách giao dịch; quy ước là H_{b}.

difficulty: Một trường vô hướng hiện tại đã bị loại bỏ do sự thay thế của đồng thuận bằng chứng làm việc. Nó được đặt là 0; quy ước là H_{d}.

number: Một giá trị vô hướng bằng số lượng khối tổ tiên. Khối khởi đầu có số là 0; quy ước là H_{i}.

gasLimit: Một giá trị vô hướng bằng với giới hạn chi tiêu gas hiện tại của mỗi khối; quy ước là H_{l}.

gasUsed: Một giá trị vô hướng bằng với tổng lượng gas đã sử dụng trong các giao dịch trong khối này; quy ước là H_{g}.

timestamp: Một giá trị vô hướng bằng với đầu ra hợp lý của time() của Unix tại thời điểm bắt đầu của khối này; quy ước là H_{s}.

extraData: Một mảng byte tùy ý chứa dữ liệu liên quan đến khối này. Phải có 32 byte hoặc ít hơn; quy ước là H_{x}.

prevRandao: mix RANDAO mới nhất[6] của trạng thái post beacon của khối trước; quy ước là H_{a}.

nonce: Giá trị 64-bit hiện tại đã bị loại bỏ do sự thay thế của đồng thuận bằng chứng làm việc. Nó được đặt là $\texttt{0x0000000000000000}$; quy ước là H_{n}.

baseFeePerGas: Một giá trị vô hướng bằng với số lượng wei được đốt cháy cho mỗi đơn vị gas tiêu thụ; quy ước là H_{f}.

Hai thành phần khác trong khối là một loạt các giao dịch, $B_{\mathbf{T}}$, và một mảng trống trước đây dành cho các header khối ommer, $B_{\mathbf{U}}$. Quy ước là, chúng ta có thể tham chiếu đến một khối B:

$$B \equiv (B_{\mathrm{H}}, B_{\mathbf{T}}, B_{\mathbf{U}}) \tag{21}$$

4.3.1. *Biên nhận Giao dịch (Transaction Receipt).* Để mã hóa thông tin về một giao dịch mà có thể hữu ích để tạo một bằng chứng không tiết lộ (zero-knowledge proof), hoặc lập chỉ mục và tìm kiếm, chúng ta mã hóa một biên nhận của mỗi giao dịch chứa một số thông tin từ quá trình thực thi của nó. Mỗi biên nhận, được ký hiệu là $B_{\mathbf{R}}[i]$ cho giao dịch thứ i, được đặt trong một trie được chỉ định bởi khóa chỉ mục cây (index-keyed trie) và gốc (root) được ghi lại trong header là H_{e}.

Biên nhận giao dịch, R, là một bộ năm mục bao gồm: loại của giao dịch, R_{x}, mã tình trạng (status code) của giao dịch, R_{z}, tổng lượng gas đã sử dụng trong khối chứa biên nhận giao dịch ngay sau khi giao dịch đã diễn ra, R_{u}, tập hợp các log được tạo ra thông qua thực thi của giao dịch, R_{l} và bộ lọc Bloom được tạo ra từ thông tin trong những log đó, R_{b}:

$$R \equiv (R_{\mathrm{x}}, R_{\mathrm{z}}, R_{\mathrm{u}}, R_{\mathrm{b}}, R_{\mathrm{l}}) \tag{22}$$

R_{x} bằng với loại của giao dịch tương ứng.

Hàm L_{R} chuẩn bị một biên nhận giao dịch để chuyển đổi thành một mảng byte được tuần tự hóa RLP (RLP-serialised):

$$L_{\mathrm{R}}(R) \equiv (R_{\mathrm{z}}, R_{\mathrm{u}}, R_{\mathrm{b}}, R_{\mathrm{l}}) \tag{23}$$

Chúng ta khẳng định rằng mã tình trạng R_{z} là một số nguyên không âm:

$$R_{\mathrm{z}} \in \mathbb{N} \tag{24}$$

Chúng ta khẳng định rằng R_{u}, tổng lượng gas đã sử dụng, là một số nguyên không âm và rằng log Bloom, R_{b}, là một hash có kích thước là 2048 bit (256 byte):

$$R_{\mathrm{u}} \in \mathbb{N} \quad \wedge \quad R_{\mathrm{b}} \in \mathbb{B}_{256} \tag{25}$$

Dãy R_{l} là một loạt các mục log, $(O_0, O_1, ...)$. Một mục log, O, là một bộ (tuple) gồm địa chỉ của logger, O_{a}, một loạt có thể trống các chủ đề log 32 byte, $O_{\mathbf{t}}$, và một số byte dữ liệu, $O_{\mathbf{d}}$:

$$O \equiv (O_{\mathrm{a}}, (O_{\mathbf{t}0}, O_{\mathbf{t}1}, ...), O_{\mathbf{d}}) \tag{26}$$

$$O_{\mathrm{a}} \in \mathbb{B}_{20} \quad \wedge \quad \forall x \in O_{\mathbf{t}} : x \in \mathbb{B}_{32} \quad \wedge \quad O_{\mathbf{d}} \in \mathbb{B} \tag{27}$$

Chúng ta định nghĩa hàm bộ lọc Bloom, M, để chuyển đổi một mục log thành một hash đơn 256 byte duy nhất:

$$M(O) \equiv \bigvee_{x \in \{O_{\mathrm{a}}\} \cup O_{\mathbf{t}}} \left(M_{3:2048}(x)\right) \tag{28}$$

trong đó $M_{3:2048}$ là một bộ lọc Bloom chuyên biệt được thiết lập với ba bit trong tổng số 2048 bit, dựa trên một chuỗi byte tùy ý. Điều này được thực hiện bằng cách lấy

[5] *ommer* là một thuật ngữ giới tính không phân biệt để chỉ "anh chị em của cha mẹ"; xem `https://nonbinary.miraheze.org/wiki/Gender_neutral_language_in_English#Aunt/Uncle`

[6] RANDAO là một giá trị giả ngẫu nhiên được tạo ra bởi các nhà xác thực trên lớp đồng thuận Ethereum. Tham khảo các quy tắc của lớp đồng thuận (`https://github.com/ethereum/consensus-specs`) để biết thêm chi tiết về RANDAO.

[7] $2048 = 2^{11}$ (11 bits), và 11 bit thấp nhất là phần dư 2048 của toán hạng, trong trường hợp này là "mỗi cặp đầu tiên của byte trong hash Keccak-256 của chuỗi byte."

11 bit thấp nhất của mỗi cặp đầu tiên của byte trong hash Keccak-256 của chuỗi byte.[7]. Quy ước:

$$M_{3:2048}(\mathbf{x} : \mathbf{x} \in \mathbb{B}) \equiv \mathbf{y} : \mathbf{y} \in \mathbb{B}_{256} \quad \text{trong đó:} \tag{29}$$

$$\mathbf{y} = (0, 0, ..., 0) \quad \text{ngoại trừ:} \tag{30}$$

$$\forall i \in \{0, 2, 4\} : \mathcal{B}_{2047-m(\mathbf{x},i)}(\mathbf{y}) = 1 \tag{31}$$

$$m(\mathbf{x}, i) \equiv \texttt{KEC}(\mathbf{x})[i, i+1] \bmod 2048 \tag{32}$$

trong đó $\mathcal{B}$ là hàm tham chiếu bit sao cho $\mathcal{B}_{\mathrm{j}}(\mathbf{x})$ bằng với bit có chỉ mục j (đánh chỉ mục từ 0) trong mảng byte $\mathbf{x}$. Đáng chú ý, nó xử lý $\mathbf{x}$ như là big-endian (các bit có ý nghĩa lớn hơn sẽ có chỉ mục nhỏ hơn).

4.3.2. *Tính hợp lý toàn diện (Holistic Validity).* Chúng ta có thể khẳng định tính hợp lý của một khối nếu và chỉ nếu nó đáp ứng một số điều kiện: trường ommers ($B_{\mathbf{U}}$) của khối phải là một mảng rỗng và tiêu đề của khối phải nhất quán với các giao dịch được cung cấp ($B_{\mathbf{T}}$). Để header nhất quán với các giao dịch $B_{\mathbf{T}}$, **stateRoot** (H_{r}) phải khớp với trạng thái kết quả sau khi thực hiện tất cả các giao dịch theo thứ tự trên trạng thái cơ sở $\boldsymbol{\sigma}$ (như được chỉ định trong phần 12), và **transactionsRoot** (H_{t}), **receiptsRoot** (H_{e}), và **logsBloom** (H_{b}) phải được tạo ra chính xác từ chính các giao dịch, các biên nhận kết quả từ quá trình thực thi và các logs kết quả, tương ứng.

$$\begin{aligned}
B_{\mathbf{U}} &\equiv () &\wedge\\
H_{\mathrm{r}} &\equiv \texttt{TRIE}(L_S(\Pi(\boldsymbol{\sigma}, B))) &\wedge\\
H_{\mathrm{t}} &\equiv \texttt{TRIE}(\{\forall i < \|B_{\mathbf{T}}\|, i \in \mathbb{N} :\\
&\qquad p_{\mathrm{T}}(i, B_{\mathbf{T}}[i])\}) &\wedge\\
H_{\mathrm{e}} &\equiv \texttt{TRIE}(\{\forall i < \|B_{\mathbf{R}}\|, i \in \mathbb{N} :\\
&\qquad p_{\mathrm{R}}(i, B_{\mathbf{R}}[i])\}) &\wedge\\
H_{\mathrm{b}} &\equiv \bigvee_{\mathbf{r} \in B_{\mathbf{R}}} (\mathbf{r}_{\mathrm{b}})
\end{aligned} \tag{33}$$

trong đó $p_{\mathrm{T}}(k, v)$ và $p_{\mathrm{R}}(k, v)$ là các phép biến đổi RLP theo cặp, nhưng với một phương pháp xử lý đặc biệt cho các giao dịch EIP-2718:

$$p_{\mathrm{T}}(k, T) \equiv \left(\texttt{RLP}(k), \begin{cases} \texttt{RLP}(L_{\mathrm{T}}(T)) & \text{nếu} \quad T_{\mathrm{x}} = 0 \\ (T_{\mathrm{x}}) \cdot \texttt{RLP}(L_{\mathrm{T}}(T)) & \text{ngược lại} \end{cases}\right) \tag{34}$$

và

$$p_{\mathrm{R}}(k, R) \equiv \left(\texttt{RLP}(k), \begin{cases} \texttt{RLP}(L_{\mathrm{R}}(R)) & \text{nếu} \quad R_{\mathrm{x}} = 0 \\ (R_{\mathrm{x}}) \cdot \texttt{RLP}(L_{\mathrm{R}}(R)) & \text{ngược lại} \end{cases}\right) \tag{35}$$

($\cdot$ là việc nối chuỗi của các mảng byte).

Hơn nữa:

$$\texttt{TRIE}(L_{\mathrm{S}}(\boldsymbol{\sigma})) = P(B_H)_{H_{\mathrm{r}}} \tag{36}$$

Do đó, $\texttt{TRIE}(L_{\mathrm{S}}(\boldsymbol{\sigma}))$ là giá trị hash của nút gốc của cấu trúc cây Merkle Patricia chứa các cặp khóa-giá trị của trạng thái $\boldsymbol{\sigma}$ với giá trị được mã hóa bằng RLP, và $P(B_{\mathrm{H}})$ là khối cha của B, được định nghĩa trực tiếp.

Các giá trị phát sinh từ việc tính toán các giao dịch, cụ thể là biên nhận giao dịch, $B_{\mathbf{R}}$, và được định nghĩa thông qua hàm tích lũy trạng thái, Π của giao dịch, sẽ được hình thành chi tiết hơn trong phần 12.2.

4.3.3. *Tuần tự hóa (Serialisation).* Hàm L_{B} và L_{H} là các hàm chuẩn bị cho một khối và header khối tương ứng. Chúng ta khẳng định các kiểu và thứ tự của cấu trúc khi sự chuyển đổi RLP được yêu cầu:

$$\begin{aligned}
L_{\mathrm{H}}(H) \equiv (\ & H_{\mathrm{p}}, H_{\mathrm{o}}, H_{\mathrm{c}}, H_{\mathrm{r}}, H_{\mathrm{t}}, H_{\mathrm{e}}, H_{\mathrm{b}}, H_{\mathrm{d}},\\
& H_{\mathrm{i}}, H_{\mathrm{l}}, H_{\mathrm{g}}, H_{\mathrm{s}}, H_{\mathrm{x}}, H_{\mathrm{a}}, H_{\mathrm{n}}, H_{\mathrm{f}}\ )
\end{aligned} \tag{37}$$

$$L_{\mathrm{B}}(B) \equiv \big(L_{\mathrm{H}}(B_{\mathrm{H}}), \widetilde{L}^*_{\mathrm{T}}(B_{\mathbf{T}}), L^*_{\mathrm{H}}(B_{\mathbf{U}})\big) \tag{38}$$

trong đó $\widetilde{L}_{\mathrm{T}}$ quan tâm đặc biệt của các giao dịch EIP-2718:

$$\widetilde{L}_{\mathrm{T}}(T) = \begin{cases} L_{\mathrm{T}}(T) & \text{nếu} \quad T_{\mathrm{x}} = 0 \\ (T_{\mathrm{x}}) \cdot \texttt{RLP}(L_{\mathrm{T}}(T)) & \text{ngược lại} \end{cases} \tag{39}$$

với $\widetilde{L}^*_{\mathrm{T}}$ và L^*_{H} là các biến đổi chuỗi theo phần tử, do đó:

$$f^*\big((x_0, x_1, ...)\big) \equiv \big(f(x_0), f(x_1), ...\big) \quad \text{cho bất kỳ hàm } f \tag{40}$$

Các loại thành phần được định nghĩa như sau:

$$\begin{aligned}
& H_{\mathrm{p}} \in \mathbb{B}_{32} \wedge H_{\mathrm{o}} \in \mathbb{B}_{32} \wedge H_{\mathrm{c}} \in \mathbb{B}_{20} \wedge\\
& H_{\mathrm{r}} \in \mathbb{B}_{32} \wedge H_{\mathrm{t}} \in \mathbb{B}_{32} \wedge H_{\mathrm{e}} \in \mathbb{B}_{32} \wedge\\
& H_{\mathrm{b}} \in \mathbb{B}_{256} \wedge H_{\mathrm{d}} \in \mathbb{N} \wedge H_{\mathrm{i}} \in \mathbb{N} \wedge\\
& H_{\mathrm{l}} \in \mathbb{N} \wedge H_{\mathrm{g}} \in \mathbb{N} \wedge H_{\mathrm{s}} \in \mathbb{N}_{256} \wedge\\
& H_{\mathrm{x}} \in \mathbb{B} \wedge H_{\mathrm{a}} \in \mathbb{B}_{32} \wedge H_{\mathrm{n}} \in \mathbb{B}_{8} \wedge\\
& H_{\mathrm{f}} \in \mathbb{N}
\end{aligned} \tag{41}$$

trong đó

$$\mathbb{B}_{\mathrm{n}} = \{B : B \in \mathbb{B} \wedge \|B\| = n\} \tag{42}$$

Bây giờ chúng ta có một đặc tả chặt chẽ cho việc xây dựng một cấu trúc khối chính thức. Hàm RLP `RLP` (xem Phụ lục B) cung cấp phương pháp cổ điển để biến đổi cấu trúc này thành một chuỗi byte sẵn sàng được truyền qua mạng hoặc lưu trữ cục bộ.

4.3.4. *Sự Hợp Lệ của Tiêu Đề Khối (Block Header Validity).* Chúng ta định nghĩa $P(B_{\mathrm{H}})$ là khối cha của B, quy ước là:

$$P(H) \equiv B' : \texttt{KEC}(\texttt{RLP}(B'\mathrm{H})) = H\mathrm{p} \tag{43}$$

Số khối là số khối của khối cha tăng lên một:

$$H_{\mathrm{i}} \equiv P(H)\mathrm{Hi} + 1 \tag{44}$$

Bản phát hành *London* giới thiệu thuộc tính khối *phí cơ sở trên mỗi gas* H_{f} (xem EIP-1559 của Buterin et al. [2019]). Phí cơ sở là lượng wei được đốt cháy cho mỗi đơn vị gas tiêu thụ trong quá trình thực hiện giao dịch trong khối. Giá trị của phí cơ sở là một hàm của sự khác biệt giữa gas được sử dụng bởi khối cha và *mục tiêu gas* của khối cha.

Phí cơ sở dự kiến trên mỗi gas được định nghĩa như sau $F(H)$:

$$F(H) \equiv \begin{cases} 1000000000 & \text{nếu} \quad H_{\mathrm{i}} = F_{\mathrm{London}} \\ P(H)_{\mathrm{H_f}} & \text{nếu} \quad P(H)_{\mathrm{H_g}} = \tau \\ P(H)_{\mathrm{H_f}} - \nu & \text{nếu} \quad P(H)_{\mathrm{H_g}} < \tau \\ P(H)_{\mathrm{H_f}} + \nu & \text{nếu} \quad P(H)_{\mathrm{H_g}} > \tau \end{cases} \tag{45}$$

trong đó:

$$\tau \equiv \lfloor \frac{P(H)_{\mathrm{H_l}}}{\rho} \rfloor \tag{46}$$

$$\rho \equiv 2 \tag{47}$$

$$\nu^* \equiv \begin{cases} \lfloor \frac{P(H)_{\mathrm{H_f}} \times (\tau - P(H)_{\mathrm{H_g}})}{\tau} \rfloor & \text{nếu} \quad P(H)_{\mathrm{H_g}} < \tau \\ \lfloor \frac{P(H)_{\mathrm{H_f}} \times (P(H)_{\mathrm{H_g}} - \tau)}{\tau} \rfloor & \text{nếu} \quad P(H)_{\mathrm{H_g}} > \tau \end{cases} \tag{48}$$

$$\nu \equiv \begin{cases} \lfloor \frac{\nu^*}{\xi} \rfloor & \text{nếu} \quad P(H)_{\mathrm{H_g}} < \tau \\ max(\lfloor \frac{\nu^*}{\xi} \rfloor, 1) & \text{nếu} \quad P(H)_{\mathrm{H_g}} > \tau \end{cases} \tag{49}$$

$$\xi \equiv 8 \tag{50}$$

Mục tiêu gas, τ, được định nghĩa là giới hạn gas H_l chia cho *hệ số đàn hồi*, ρ, một hằng số toàn cục được đặt là 2. Vì vậy, mặc dù các khối có thể tiêu thụ nhiều gas bằng giới hạn gas, nhưng phí cơ sở được điều chỉnh sao cho trung bình các khối tiêu thụ nhiều gas bằng mục tiêu gas. Phí cơ sở tăng trong khối hiện tại khi sử dụng gas của khối cha vượt quá mục tiêu gas, và ngược lại, phí cơ sở giảm trong khối hiện tại khi sử dụng gas của khối cha ít hơn mục tiêu gas.

Cường độ của sự tăng hoặc giảm phí cơ sở, được định nghĩa là ν, tỉ lệ với sự khác biệt giữa lượng gas mà khối cha tiêu thụ và mục tiêu gas của khối cha. Ảnh hưởng đến phí cơ sở được làm dịu bởi một hằng số toàn cục được gọi là *mẫu số tối đa biến đổi phí cơ sở*, quy ước là ξ, được đặt là 8. Một giá trị là 8 đồng nghĩa với việc phí cơ sở có thể tăng hoặc giảm tối đa 12.5% từ một khối sang khối tiếp theo.

Giới hạn gas chuẩn H_l của một khối có header H phải đáp ứng mối quan hệ:

$$\begin{aligned} & H_\mathrm{l} < P(H)_{\mathrm{H_{l'}}} + \left\lfloor \frac{P(H)_{\mathrm{H_{l'}}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} > P(H)_{\mathrm{H_{l'}}} - \left\lfloor \frac{P(H)_{\mathrm{H_{l'}}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} \geqslant 5000 \end{aligned} \tag{51}$$

trong đó:

$$P(H)_{\mathrm{H_{l'}}} \equiv \begin{cases} P(H)_{\mathrm{H_l}} \times \rho & \text{nếu} \quad H_\mathrm{i} = F_\mathrm{London} \\ P(H)_{\mathrm{H_l}} & \text{nếu} \quad H_\mathrm{i} > F_\mathrm{London} \end{cases} \tag{52}$$

Để tránh sự gián đoạn trong việc sử dụng gas, giá trị giới hạn gas của khối cha cho mục đích xác nhận giới hạn gas của khối hiện tại được sửa đổi tại khối *fork London* bằng cách nhân nó với *hệ số đàn hồi*, ρ. Chúng ta gọi giá trị sửa đổi này là $P(H)_{\mathrm{H_{l'}}}$. Điều này đảm bảo rằng mục tiêu gas cho các khối sau *fork London* có thể được đặt xấp xỉ với giới hạn gas của các khối trước *fork London*.

H_s là dấu thời gian (theo thời gian Unix's) của khối H và phải đáp ứng mối quan hệ:

$$H_\mathrm{s} > P(H)_{\mathrm{H_s}} \tag{53}$$

Hard fork *Paris* đã thay đổi cơ chế đồng thuận của Ethereum từ bằng chứng công việc sang bằng chứng cổ phần, và do đó đã không còn sử dụng nhiều thuộc tính trong phần đầu khối (block header) liên quan đến bằng chứng công việc. Những thuộc tính đã không còn sử dụng này bao gồm **nonce** ($H_\mathbf{n}$), **ommersHash** ($H_\mathbf{o}$), **difficulty** ($H_\mathbf{d}$), và **mixHash** ($H_\mathbf{m}$).

mixHash đã được thay thế bằng một trường mới **prevRandao** ($H_\mathbf{a}$). Các trường header khác liên quan đến bằng chứng công việc đã được thay thế bằng các hằng số:

$$H_\mathrm{o} \equiv \texttt{KEC}(\texttt{RLP}(())) \tag{54}$$

$$H_\mathrm{d} \equiv 0 \tag{55}$$

$$H_\mathrm{n} \equiv \texttt{0x0000000000000000} \tag{56}$$

Giá trị của **prevRandao** phải được xác định bằng cách sử dụng thông tin từ Beacon Chain. Mặc dù chi tiết về cách tạo giá trị *RANDAO* trên Beacon Chain vượt quá phạm vi của bài báo này, nhưng chúng ta có thể tham chiếu đến giá trị *RANDAO* dự kiến cho khối trước đó là `PREVRANDAO`().

Do đó, chúng ta có thể định nghĩa hàm kiểm tra tính hợp lệ của tiêu đề khối (block header) $V(H)$:

$$\begin{aligned} V(H) \equiv \; & H_\mathrm{g} \leq H_\mathrm{l} \quad \wedge \\ & H_\mathrm{l} < P(H)_{\mathrm{H_{l'}}} + \left\lfloor \frac{P(H)_{\mathrm{H_{l'}}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} > P(H)_{\mathrm{H_{l'}}} - \left\lfloor \frac{P(H)_{\mathrm{H_{l'}}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} \geqslant 5000 \quad \wedge \\ & H_\mathrm{s} > P(H)_{\mathrm{H_s}} \quad \wedge \\ & H_\mathrm{i} = P(H)_{\mathrm{H_i}} + 1 \quad \wedge \\ & \|H_\mathrm{x}\| \leq 32 \quad \wedge \\ & H_\mathrm{f} = F(H) \wedge \\ & H_\mathrm{o} = \texttt{KEC}(\texttt{RLP}(())) \quad \wedge \\ & H_\mathrm{d} = 0 \quad \wedge \\ & H_\mathrm{n} = \texttt{0x0000000000000000} \quad \wedge \\ & H_\mathrm{a} = \texttt{PREVRANDAO}() \end{aligned} \tag{57}$$

Lưu ý thêm rằng **extraData** phải có tối đa 32 byte.

5. Gas và Thanh toán

Để tránh vấn đề lạm dụng mạng và để né tránh các câu hỏi không tránh khỏi xuất phát từ tính hoàn chỉnh của Turing, tất cả các tính toán có thể lập trình trong Ethereum đều phải trả phí. Biểu phí được quy định theo đơn vị *gas* (xem Phụ lục G để biết các phí liên quan đến các loại tính toán khác nhau). Do đó, bất kỳ đoạn tính toán có thể lập trình nào (điều này bao gồm việc tạo hợp đồng, thực hiện cuộc gọi tin nhắn, sử dụng và truy cập vào bộ nhớ tài khoản cũng như thực thi các hoạt động trên máy ảo) có một thỏa thuận chung chi phí về gas.

Mỗi giao dịch đều có một lượng gas cụ thể đi kèm: **gasLimit**. Đây là lượng gas được ngầm mua từ số dư tài khoản của người gửi. Việc mua này diễn ra ở mức *giá gas hiệu quả* được xác định trong phần 6. Giao dịch được coi là không hợp lệ nếu số dư tài khoản không thể hỗ trợ việc mua này. Nó được đặt tên là **gasLimit** vì bất kỳ gas không sử dụng nào ở cuối giao dịch đều được hoàn trả (với cùng mức giá mua) vào số dư tài khoản của người gửi. Gas không tồn tại ngoài việc thực hiện một giao dịch. Do đó, đối với các tài khoản có mã nguồn đáng tin cậy được liên kết, một giới hạn gas tương đối cao có thể được đặt và giữ nguyên.

Kể từ khi giới thiệu EIP-1559 bởi Buterin et al. [2019] trong hard fork *London*, mỗi giao dịch được đưa vào trong một khối phải trả một *phí cơ sở*, được xác định là wei cho mỗi đơn vị gas tiêu thụ và là hằng số cho mỗi giao dịch trong một khối. Ether được trả để đáp ứng phí cơ sở này sẽ được đốt cháy (rút khỏi lưu thông). phí cơ sở điều chỉnh động như là một hàm của lượng gas tiêu thụ trong khối

trước đó so với *mục tiêu gas* của nó (một giá trị hiện tại là một nửa giới hạn gas của khối, có thể được điều chỉnh bởi các trình xác nhận (validators)). Nếu tổng mức tiêu thụ gas của khối trước vượt quá mục tiêu gas, điều này cho thấy nhu cầu vượt quá về không gian khối ở mức phí cơ sở hiện tại và phí cơ sở sẽ tăng lên. Ngược lại, nếu lượng gas tiêu thụ trong khối trước đó thấp hơn mục tiêu gas, nhu cầu về không gian khối thấp hơn mục tiêu gas ở giá cơ bản hiện tại, và do đó phí cơ sở sẽ giảm. Quá trình điều chỉnh phí cơ sở này giúp đưa lượng gas tiêu thụ trung bình của các khối vào đúng với mục tiêu gas. Xem phần 4.3 để biết chi tiết hơn về cách phí cơ sở được đặt.

Để khuyến khích các trình xác nhận (validators) xử lý các giao dịch, có một khoản phí bổ sung được biết đến là *phí ưu tiên* (*priority fee*), cũng được xác định dưới dạng wei cho mỗi đơn vị gas tiêu thụ. Tổng phí mà người tạo giao dịch (transactor) phải trả là tổng của phí cơ sở trên mỗi đơn vị gas và phí ưu tiên trên mỗi đơn vị gas, nhân với tổng lượng gas tiêu thụ. Ether được sử dụng để thanh toán phí ưu tiên được chuyển đến địa chỉ *người thụ hưởng* (*beneficiary*), địa chỉ của một tài khoản thường nằm dưới sự kiểm soát của trình xác nhận (validator).

Người tạo giao dịch (transactors) sử dụng giao dịch loại 2 có thể chỉ định phí ưu tiên tối đa mà họ sẵn lòng trả (**maxPriorityFeePerGas**), cũng như tổng phí tối đa mà họ sẵn lòng trả (**maxFeePerGas**), bao gồm cả phí ưu tiên và phí cơ sở. **maxFeePerGas** phải ít nhất bằng phí cơ sở để giao dịch được đưa vào trong một khối. Giao dịch loại 0 và loại 1 chỉ có một trường để chỉ định giá gas–**gasPrice**–cũng phải ít nhất bằng phí cơ sở để được đưa vào một khối. Số lượng bởi **gasPrice** vượt quá phí cơ sở chính là phí ưu tiên trong trường hợp của một giao dịch loại 0 hoặc loại 1.

Người gửi giao dịch có thể tự do chọn bất kỳ phí ưu tiên nào mà họ mong muốn, tuy nhiên trình xác nhận (validator) cũng có quyền bỏ qua giao dịch theo cách họ chọn. Do đó, một phí ưu tiên cao hơn trên một giao dịch sẽ tốn nhiều hơn về mặt Ether đối với người gửi và mang lại giá trị lớn hơn cho trình xác nhận (validator), và do đó có khả năng được chọn để xử lý. Vì sẽ có một phân phối (có trọng số) của phí ưu tiên tối thiểu chấp nhận được, người gửi giao dịch sẽ phải đưa ra một sự cân nhắc giữa việc giảm phí ưu tiên và tối đa hóa cơ hội giao dịch của họ được đưa vào một khối một cách kịp thời.

6. Thực thi giao dịch (Transaction Execution)

Quá trình thực thi của một giao dịch là phần phức tạp nhất của giao thức Ethereum: nó định nghĩa hàm chuyển trạng thái Υ. Giả định rằng mọi giao dịch được thực hiện đều trải qua các kiểm tra ban đầu về tính hợp lệ nội tại. Các kiểm tra này bao gồm:

(1) Giao dịch là định dạng RLP chính xác, không có các byte thừa ở cuối;
(2) Chữ ký của giao dịch là hợp lệ;
(3) Nonce của giao dịch là hợp lệ (tương đương với nonce hiện tại của tài khoản người gửi (sender));
(4) Tài khoản người gửi không có mã hợp đồng triển khai (xem EIP-3607 của Feist et al. [2021]);
(5) Giới hạn gas không nhỏ hơn gas nội tại, g_0, được sử dụng bởi giao dịch;
(6) Số dư của tài khoản người gửi chứa ít nhất số chi phí cần thiết, v_0, được yêu cầu trong khoản thanh toán trước;
(7) **maxFeePerGas**, T_m, trong trường hợp của giao dịch loại 2, hoặc **gasPrice**, T_p, trong trường hợp của giao dịch loại 0 và loại 1, lớn hơn hoặc bằng phí cơ sở của khối, H_f; và
(8) đối với giao dịch loại 2, **maxPriorityFeePerGas**, T_f, phải không lớn hơn **maxFeePerGas**, T_m.

Một cách chính thức, chúng ta xem xét hàm Υ, với T là một giao dịch và $\boldsymbol{\sigma}$ là trạng thái:

$$\boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T) \tag{58}$$

Như vậy, $\boldsymbol{\sigma}'$ là trạng thái sau giao dịch. Chúng ta cũng định nghĩa Υ^g để đánh giá lượng gas được sử dụng trong quá trình thực hiện một giao dịch, $\Upsilon^\mathbf{l}$ để đánh giá các mục log được tích lũy của giao dịch và Υ^z để đánh giá mã tình trạng kết quả từ giao dịch. Những điều này sẽ được định nghĩa chính thức sau này.

6.1. Trạng thái con (Substate).

Trong suốt quá trình thực thi giao dịch, chúng ta tích lũy một số thông tin được thực hiện ngay theo giao dịch. Chúng ta gọi đó là *trạng thái con giao dịch đã tích lũy*, hoặc ngắn gọn là *trạng thái con tích lũy*, và biểu diễn nó như là A, là một bộ (tuple):

$$A \equiv (A_\mathbf{s}, A_\mathbf{l}, A_\mathbf{t}, A_\mathrm{r}, A_\mathbf{a}, A_\mathbf{K}) \tag{59}$$

Phần tử của bộ bao gồm $A_\mathbf{s}$, tập hợp tự phá huỷ (self-destruct): một tập hợp các tài khoản sẽ bị loại bỏ sau khi giao dịch hoàn tất. $A_\mathbf{l}$ là dãy log: đây là một dãy các 'điểm tham chiếu' ('checkpoints') được lưu trữ và có thể được lập chỉ mục trong việc thực thi mã VM, giúp theo dõi dễ dàng các cuộc gọi hợp đồng bởi những người quan sát bên ngoài thế giới Ethereum (như giao diện người dùng của ứng dụng phi tập trung). $A_\mathbf{t}$ là tập hợp các tài khoản đã chạm vào, trong đó những tài khoản trống sẽ bị xóa vào cuối giao dịch. A_r là số dư hoàn trả, tăng lên thông qua việc sử dụng chỉ thị SSTORE để đặt lại bộ nhớ lưu trữ của hợp đồng về không từ giá trị khác không. Mặc dù không được hoàn trả ngay lập tức, nó được phép một phần để làm giảm bớt chi phí thực thi tổng cộng. Cuối cùng, EIP-2929 của Buterin and Swende [2020a] giới thiệu $A_\mathbf{a}$, tập hợp địa chỉ tài khoản đã được truy cập, và $A_\mathbf{K}$, tập hợp các khóa bộ nhớ được truy cập (chính xác hơn, mỗi phần tử của $A_\mathbf{K}$ là một bộ (tuple) giữa một địa chỉ tài khoản 20 byte và một khe bộ nhớ 32 byte).

Chúng ta định nghĩa trạng thái con đã tích lũy rỗng A^0 thành không có tự phá huỷ (self-destruct), không có log, không có tài khoản đã chạm vào, số dư hoàn trả là không, tất cả hợp đồng đã được biên dịch trước trong địa chỉ đã được truy cập, và không có bộ nhớ nào đã được truy cập:

$$A^0 \equiv (\varnothing, (), \varnothing, 0, \pi, \varnothing) \tag{60}$$

trong đó π là tập hợp của tất cả địa chỉ được biên dịch trước.

6.2. Thực thi (Execution).

Chúng ta định nghĩa gas nội tại g_0, lượng gas mà giao dịch này yêu cầu phải trả

trước khi thực thi, như sau:

(61)
$$\begin{aligned} g_0 \equiv & \sum_{i \in T_{\mathbf{i}}, T_{\mathbf{d}}} \begin{cases} G_{\text{txdatazero}} & \text{nếu} \quad i = 0 \\ G_{\text{txdatanonzero}} & \text{ngược lại} \end{cases} \\ & + \begin{cases} G_{\text{txcreate}} & \text{nếu} \quad T_{\text{t}} = \varnothing \\ 0 & \text{ngược lại} \end{cases} \\ & + G_{\text{transaction}} \\ & + \sum_{j=0}^{\|T_{\mathbf{A}}\|-1} \left(G_{\text{accesslistaddress}} + \|T_{\mathbf{A}}[j]_{\mathbf{s}}\| G_{\text{accessliststorage}} \right) \end{aligned}$$

trong đó $T_{\mathbf{i}}, T_{\mathbf{d}}$ là chuỗi byte của dữ liệu và EVM-code khởi tạo liên quan đến giao dịch, tùy thuộc vào việc giao dịch là để tạo hợp đồng hay để gọi tin nhắn. G_{txcreate} được thêm vào nếu giao dịch là để tạo hợp đồng, nhưng không được thêm nếu là kết quả của EVM-code. $G_{\text{accesslistaddress}}$ và $G_{\text{accessliststorage}}$ là chi phí của quá trình khởi động quyền truy cập tài khoản và bộ nhớ, tương ứng. G được định nghĩa đầy đủ trong Phụ lục G.

Chúng ta định nghĩa *giá gas hiệu quả*, quy ước là p, là số wei mà người ký gửi giao dịch sẽ trả cho mỗi đơn vị gas tiêu thụ trong quá trình thực thi giao dịch. Nó được tính như sau:

$$p \equiv \begin{cases} T_{\text{p}} & \text{nếu } T_{\text{x}} = 0 \vee T_{\text{x}} = 1 \\ f + H_{\text{f}} & \text{nếu } T_{\text{x}} = 2 \end{cases} \tag{62}$$

trong đó f là *phí ưu tiên*—số wei mà địa chỉ người thụ hưởng của khối sẽ nhận được cho mỗi đơn vị gas tiêu thụ trong quá trình thực hiện giao dịch. Nó được tính như sau:

$$f \equiv \begin{cases} T_{\text{p}} - H_{\text{f}} & \text{nếu } T_{\text{x}} = 0 \vee T_{\text{x}} = 1 \\ min(T_{\text{f}}, T_{\text{m}} - H_{\text{f}}) & \text{nếu } T_{\text{x}} = 2 \end{cases} \tag{63}$$

Chi phí ban đầu v_0 được tính như sau:

$$v_0 \equiv \begin{cases} T_{\text{g}} T_{\text{p}} + T_{\text{v}} & \text{nếu } T_{\text{x}} = 0 \vee T_{\text{x}} = 1 \\ T_{\text{g}} T_{\text{m}} + T_{\text{v}} & \text{nếu } T_{\text{x}} = 2 \end{cases} \tag{64}$$

Độ hợp lệ được xác định như sau:

$$\begin{aligned} S(T) &\neq \varnothing \quad \wedge \\ \boldsymbol{\sigma}[S(T)]_{\text{c}} &= \texttt{KEC}(()) \quad \wedge \\ T_{\text{n}} &= \boldsymbol{\sigma}[S(T)]_{\text{n}} \quad \wedge \\ g_0 &\leqslant T_{\text{g}} \quad \wedge \\ v_0 &\leqslant \boldsymbol{\sigma}[S(T)]_{\text{b}} \quad \wedge \\ m &\geqslant H_{\text{f}} \quad \wedge \\ T_{\text{g}} &\leqslant B_{\text{Hl}} - \ell(B_{\mathbf{R}})_{\text{u}} \end{aligned} \tag{65}$$

trong đó

$$m \equiv \begin{cases} T_{\text{p}} & \text{nếu } T_{\text{x}} = 0 \vee T_{\text{x}} = 1 \\ T_{\text{m}} & \text{nếu } T_{\text{x}} = 2 \end{cases} \tag{66}$$

Lưu ý điều kiện cuối cùng; tổng của giới hạn gas của giao dịch, T_{g}, và gas đã sử dụng trong khối này trước đó, cho bởi $\ell(B_{\mathbf{R}})_{\text{u}}$, phải không lớn hơn giới hạn gas của khối, B_{Hl}. Ngoài ra, với một cách sử dụng ký hiệu một chút, chúng ta giả định rằng $\boldsymbol{\sigma}[S(T)]_{\text{c}} = \texttt{KEC}(())$, $\boldsymbol{\sigma}[S(T)]_{\text{n}} = 0$, và $\boldsymbol{\sigma}[S(T)]_{\text{b}} = 0$ nếu $\boldsymbol{\sigma}[S(T)] = \varnothing$.

Đối với giao dịch loại 2, chúng ta thêm một kiểm tra bổ sung rằng **maxPriorityFeePerGas** không lớn hơn **maxFeePerGas**:

$$T_{\text{m}} \geqslant T_{\text{f}} \tag{67}$$

Việc thực hiện một giao dịch hợp lệ bắt đầu với một thay đổi không thể hoàn nguyên được thực hiện trên trạng thái: nonce của tài khoản của người gửi, $S(T)$, được tăng lên một và số dư giảm đi một phần của chi phí trả trước, $T_{\text{g}}p$. Gas có sẵn cho quá trình tính toán tiếp theo, g, được định nghĩa là $T_{\text{g}} - g_0$. Quá trình tính toán, có thể là tạo hợp đồng hoặc cuộc gọi tin nhắn, dẫn đến một trạng thái cuối cùng (có thể hợp pháp giống với trạng thái hiện tại), sự thay đổi mang tính quyết định và không bao giờ không hợp lệ: không thể có giao dịch không hợp lệ từ điểm này trở đi.

Chúng ta định nghĩa trạng thái điểm tham chiếu (checkpoint state) $\boldsymbol{\sigma}_0$:

$$\boldsymbol{\sigma}_0 \equiv \boldsymbol{\sigma} \quad \text{ngoại trừ:} \tag{68}$$
$$\boldsymbol{\sigma}_0[S(T)]_{\text{b}} \equiv \boldsymbol{\sigma}[S(T)]_{\text{b}} - T_{\text{g}}p \tag{69}$$
$$\boldsymbol{\sigma}_0[S(T)]_{\text{n}} \equiv \boldsymbol{\sigma}[S(T)]_{\text{n}} + 1 \tag{70}$$

Việc xác định $\boldsymbol{\sigma}_{\text{P}}$ từ $\boldsymbol{\sigma}_0$ phụ thuộc vào loại giao dịch; có thể là tạo hợp đồng (contract creation) hoặc gọi tin nhắn (message call); chúng ta định nghĩa bộ giá trị của trạng thái dự kiến sau khi thực thi $\boldsymbol{\sigma}_{\text{P}}$, gas còn lại g', trạng thái con tích lũy A và mã tình trạng (status code) z:

(71)
$$(\boldsymbol{\sigma}_{\text{P}}, g', A, z) \equiv \begin{cases} \Lambda_4(\boldsymbol{\sigma}_0, A^*, S(T), S(T), g, \\ \quad p, T_{\text{v}}, T_{\mathbf{i}}, 0, \varnothing, \top) & \text{nếu} \quad T_{\text{t}} = \varnothing \\ \Theta_4(\boldsymbol{\sigma}_0, A^*, S(T), S(T), T_{\text{t}}, \\ \quad T_{\text{t}}, g, p, T_{\text{v}}, T_{\text{v}}, T_{\mathbf{d}}, 0, \top) & \text{ngược lại} \end{cases}$$

trong đó

$$A^* \equiv A^0 \quad \text{ngoại trừ} \tag{72}$$
$$A^*_{\mathbf{a}} \equiv A^0_{\mathbf{a}} \cup \{S(T)\} \cup_{E \in T_{\mathbf{A}}} \{E_{\text{a}}\} \tag{73}$$
$$A^*_{\mathbf{K}} \equiv \bigcup_{E \in T_{\mathbf{A}}} \left\{ \forall i < \|E_{\mathbf{s}}\|, i \in \mathbb{N} : \ (E_{\text{a}}, E_{\mathbf{s}}[i]) \right\} \tag{74}$$

và g là lượng gas còn lại sau khi trừ số gas cơ sở cần phải trả cho sự tồn tại của giao dịch:

$$g \equiv T_{\text{g}} - g_0 \tag{75}$$

Lưu ý rằng chúng ta sử dụng Θ_4 và Λ_4 để biểu thị thực tế là chỉ có bốn thành phần đầu tiên của các giá trị của hàm là được lấy; giá trị cuối cùng biểu thị giá trị đầu ra của cuộc gọi tin nhắn (một mảng byte) và không được sử dụng trong ngữ cảnh đánh giá giao dịch.

Sau đó, trạng thái được hoàn tất bằng cách xác định số lượng cần hoàn trả, g^* từ gas còn lại, g', cộng với một số lượng cho phép từ bộ đếm hoàn trả, cho người gửi theo tỷ lệ ban đầu (original rate).

$$g^* \equiv g' + \min\left\{ \left\lfloor \frac{T_{\text{g}} - g'}{5} \right\rfloor, A_{\text{r}} \right\} \tag{76}$$

Số lượng hoàn trả tổng cộng là gas hợp lệ còn lại g', cộng thêm A_{r}, với thành phần sau được giữ lại tối đa một phần năm[8] (làm tròn xuống) của tổng số gas đã sử dụng $T_{\text{g}} - g'$. Do đó, g^* là tổng gas còn lại sau khi giao dịch đã được thực hiện.

[8]Tỷ lệ hoàn trả tối đa của gas đã được giảm từ một nửa xuống một phần năm bởi EIP-3529 bởi Buterin and Swende [2021] trong phiên bản *London*

Trình xác nhận (validator), người mà địa chỉ của họ được chỉ định làm người thụ hưởng của khối hiện tại B, nhận được gas đã tiêu thụ nhân với *lệ phí ưu tiên mỗi gas* của giao dịch, được xác định là f trong phần này. Số ether được người giao dịch thanh toán để tính phí cơ sở sẽ bị ghi nợ từ tài khoản của họ nhưng không được ghi có vào tài khoản nào khác, vì vậy nó sẽ bị đốt cháy.

Chúng ta định nghĩa trước trạng thái cuối (pre-final state) $\boldsymbol{\sigma}^*$ dựa trên trạng thái tạm thời $\boldsymbol{\sigma}_{\mathrm{P}}$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma}_{\mathrm{P}} \quad \text{ngoại trừ} \tag{77}$$

$$\boldsymbol{\sigma}^*[S(T)]_{\mathrm{b}} \equiv \boldsymbol{\sigma}_{\mathrm{P}}[S(T)]_{\mathrm{b}} + g^* p \tag{78}$$

$$\boldsymbol{\sigma}^*[B_{\mathrm{Hc}}]_{\mathrm{b}} \equiv \boldsymbol{\sigma}_{\mathrm{P}}[B_{\mathrm{Hc}}]_{\mathrm{b}} + (T_{\mathrm{g}} - g^*) f \tag{79}$$

Trạng thái cuối (final state), $\boldsymbol{\sigma}'$, được đạt được sau khi xóa tất cả các tài khoản xuất hiện trong tập tự hủy (self-destruct) hoặc được chạm vào và trống rỗng:

$$\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma}^* \quad \text{ngoại trừ} \tag{80}$$

$$\forall i \in A_{\mathbf{s}} : \boldsymbol{\sigma}'[i] = \varnothing \tag{81}$$

$$\forall i \in A_{\mathbf{t}} : \boldsymbol{\sigma}'[i] = \varnothing \quad \text{nếu} \quad \texttt{DEAD}(\boldsymbol{\sigma}^*, i) \tag{82}$$

Và cuối cùng, chúng ta xác định Υ^{g}, tổng lượng gas sử dụng trong giao dịch này Υ^{l}, các nhật ký (logs) được tạo ra bởi giao dịch này và Υ^{z}, mã tình trạng (status code) của giao dịch này:

$$\Upsilon^{\mathrm{g}}(\boldsymbol{\sigma}, T) \equiv T_{\mathrm{g}} - g^* \tag{83}$$

$$\Upsilon^{\mathrm{l}}(\boldsymbol{\sigma}, T) \equiv A_{\mathrm{l}} \tag{84}$$

$$\Upsilon^{\mathrm{z}}(\boldsymbol{\sigma}, T) \equiv z \tag{85}$$

Các giá trị này được sử dụng để giúp định nghĩa biên nhận giao dịch (transaction receipt) và cũng được sử dụng sau này để xác minh trạng thái.

7. Tạo hợp đồng (Contract Creation)

Có một số tham số nội tại được sử dụng khi tạo một tài khoản: người gửi (s), người giao dịch ban đầu[9] (o), lượng gas khả dụng (g), giá gas hiệu quả (p), giá trị tài trợ (v) cùng với một mảng byte có độ dài tùy ý, $\mathbf{i}$, mã khởi tạo EVM, độ sâu hiện tại của ngăn xếp cuộc gọi tin nhắn/tạo hợp đồng (e), muối (salt) cho địa chỉ tài khoản mới (ζ) và cuối cùng là quyền được phép thực hiện sửa đổi trạng thái (w). Muối (salt) ζ có thể thiếu ($\zeta = \varnothing$); theo quy ước,

$$\zeta \in \mathbb{B}_{32} \cup \mathbb{B}_0 \tag{86}$$

Nếu việc tạo được gây ra bởi CREATE2, thì $\zeta \neq \varnothing$.

Chúng ta định nghĩa hàm tạo, quy ước là hàm Λ, mà đánh giá từ những giá trị này, cùng với trạng thái $\boldsymbol{\sigma}$ và trạng thái con được tích lũy A, đến bộ (tuple) bao gồm trạng thái mới, gas còn lại, trạng thái con mới được tích lũy, mã tình trạng (status code) và đầu ra $(\boldsymbol{\sigma}', g', A', z, \mathbf{o})$:

$$(\boldsymbol{\sigma}', g', A', z, \mathbf{o}) \equiv \Lambda(\boldsymbol{\sigma}, A, s, o, g, p, v, \mathbf{i}, e, \zeta, w) \tag{87}$$

Địa chỉ của tài khoản mới được xác định là 160 bit bên phải của hash Keccak-256 của mã hóa RLP của cấu trúc chỉ chứa người gửi và số nonce của tài khoản. Đối với CREATE2, quy tắc là khác và được mô tả trong EIP-1014 của Buterin [2018]. Kết hợp hai trường hợp, chúng ta định nghĩa địa chỉ kết quả cho tài khoản mới là a:

$$a \equiv \texttt{ADDR}(s, \boldsymbol{\sigma}[s]_{\mathrm{n}} - 1, \zeta, \mathbf{i}) \tag{88}$$

$$\texttt{ADDR}(s, n, \zeta, \mathbf{i}) \equiv \mathcal{B}_{96..255}\Big(\texttt{KEC}\big(L_{\mathrm{A}}(s, n, \zeta, \mathbf{i})\big)\Big) \tag{89}$$

$$L_{\mathrm{A}}(s, n, \zeta, \mathbf{i}) \equiv \begin{cases} \texttt{RLP}\big(\ (s, n)\ \big) & \text{nếu } \zeta = \varnothing \\ (255) \cdot s \cdot \zeta \cdot \texttt{KEC}(\mathbf{i}) & \text{ngược lại} \end{cases} \tag{90}$$

Ở đây, $\cdot$ đại diện cho việc nối các mảng byte, $\mathcal{B}_{a..b}(X)$ đánh giá thành một giá trị nhị phân chứa các bit có chỉ số trong khoảng $[a, b]$ của dữ liệu nhị phân X, và $\boldsymbol{\sigma}[x]$ là trạng thái địa chỉ của x, hoặc $\varnothing$ nếu không tồn tại. Lưu ý rằng chúng ta sử dụng giá trị nonce của người gửi ít hơn một; chúng ta khẳng định rằng chúng ta đã tăng giá trị nonce của tài khoản người gửi trước cuộc gọi này, và vì vậy, giá trị được sử dụng là giá trị nonce của người gửi ở đầu giao dịch hoặc hoạt động VM.

Địa chỉ của tài khoản mới được thêm vào tập hợp các tài khoản được truy cập:

$$A^* \equiv A \quad \text{ngoại trừ} \quad A^*_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{a\} \tag{91}$$

Nonce của tài khoản được định nghĩa ban đầu bằng một, số dư là giá trị được truyền vào, kho lưu trữ là trống và mã hash là băm Keccak 256-bit của chuỗi rỗng; số dư của người gửi cũng giảm đi giá trị được truyền vào. Do đó, trạng thái biến đổi trở thành $\boldsymbol{\sigma}^*$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma} \quad \text{ngoại trừ:} \tag{92}$$

$$\boldsymbol{\sigma}^*[a] = \big(1, v + v', \texttt{TRIE}(\varnothing), \texttt{KEC}(())\big) \tag{93}$$

$$\boldsymbol{\sigma}^*[s] = \begin{cases} \varnothing & \text{nếu } \boldsymbol{\sigma}[s] = \varnothing \ \wedge \ v = 0 \\ \mathbf{a}^* & \text{ngược lại} \end{cases} \tag{94}$$

$$\mathbf{a}^* \equiv (\boldsymbol{\sigma}[s]_{\mathrm{n}}, \boldsymbol{\sigma}[s]_{\mathrm{b}} - v, \boldsymbol{\sigma}[s]_{\mathbf{s}}, \boldsymbol{\sigma}[s]_{\mathrm{c}}) \tag{95}$$

trong đó v' là giá trị tồn tại trước đó của tài khoản, trong trường hợp nó đã tồn tại trước đó:

$$v' \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\sigma}[a] = \varnothing \\ \boldsymbol{\sigma}[a]_{\mathrm{b}} & \text{ngược lại} \end{cases} \tag{96}$$

Cuối cùng, tài khoản được khởi tạo thông qua việc thực thi mã EVM khởi tạo $\mathbf{i}$ theo mô hình thực thi (xem phần 9). Quá trình thực thi mã có thể tác động đến một số sự kiện không nằm trong trạng thái thực thi: kho lưu trữ của tài khoản có thể bị thay đổi, có thể tạo thêm tài khoản và có thể thực hiện thêm cuộc gọi tin nhắn.

Do đó, hàm thực thi mã Ξ đánh giá thành một bộ giá trị (tuple) bao gồm trạng thái kết quả $\boldsymbol{\sigma}^{**}$, gas khả dụng còn lại g^{**}, trạng thái con tích lũy A^{**} và nội dung mã nguồn (body code) của tài khoản $\mathbf{o}$.

$$(\boldsymbol{\sigma}^{**}, g^{**}, A^{**}, \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}^*, g, A^*, I) \tag{97}$$

trong đó I chứa các tham số của môi trường thực thi, tức là:

[9] có thể khác người gửi trong trường hợp của một cuộc gọi tin nhắn hoặc tạo hợp đồng không phải do một giao dịch kích hoạt trực tiếp mà đến từ việc thực thi mã EVM

(98) $$I_{\mathrm{a}} \equiv a$$

(99) $$I_{\mathrm{o}} \equiv o$$

(100) $$I_{\mathrm{p}} \equiv p$$

(101) $$I_{\mathbf{d}} \equiv ()$$

(102) $$I_{\mathrm{s}} \equiv s$$

(103) $$I_{\mathrm{v}} \equiv v$$

(104) $$I_{\mathbf{b}} \equiv \mathbf{i}$$

(105) $$I_{\mathrm{e}} \equiv e$$

(106) $$I_{\mathrm{w}} \equiv w$$

$I_{\mathbf{d}}$ đánh giá thành bộ (tuple) giá trị rỗng vì không có dữ liệu đầu vào cho cuộc gọi này. I_{H} không có xử lý đặc biệt và được xác định từ chuỗi khối.

Việc thực thi mã sẽ tiêu thụ gas, và gas không nên xuống dưới 0, do đó quá trình thực thi có thể kết thúc trước khi mã đạt đến một trạng thái dừng tự nhiên. Trong trường hợp ngoại lệ này (và một số trường hợp ngoại lệ khác), chúng ta nói rằng một ngoại lệ hết gas out-of-gas (OOG) đã xảy ra: Trạng thái được đánh giá là tập hợp rỗng, $\varnothing$, và toàn bộ quá trình tạo không ảnh hưởng đến trạng thái, một cách hiệu quả để lại nó như nó đã từng ngay trước khi thực hiện quá trình tạo.

Nếu mã khởi tạo hoàn tất thành công, một chi phí sau cùng của việc tạo hợp đồng được thanh toán, chi phí đặt cọc mã, c, tỷ lệ thuận với kích thước của mã hợp đồng đã tạo:

(107) $$c \equiv G_{\mathrm{codedeposit}} \times \|\mathbf{o}\|$$

Nếu không có đủ gas còn lại để thanh toán, tức là $g^{**} < c$, chúng ta cũng tuyên bố có một ngoại lệ hết gas.

Gas còn lại sẽ bằng không trong bất kỳ điều kiện ngoại lệ nào như vậy, tức là nếu quá trình tạo ra được thực hiện như là sự tiếp nhận của một giao dịch, thì điều này không ảnh hưởng đến việc thanh toán chi phí nội tại của việc tạo hợp đồng; chi phí này sẽ được thanh toán bất kể. Tuy nhiên, giá trị của giao dịch không được chuyển đến địa chỉ của hợp đồng đã bị hủy khi chúng ta hết gas, do đó mã của hợp đồng không được lưu trữ.

Nếu một ngoại lệ như vậy không xảy ra, thì gas còn lại sẽ được hoàn trả cho người tạo và trạng thái đã biến đổi bây giờ được phép tồn tại. Do đó, theo quy ước, chúng ta có thể xác định trạng thái kết quả, gas, trạng thái con đã tích lũy và mã tình trạng là $(\boldsymbol{\sigma}', g', A', z)$ trong đó:

(108)
$$g' \equiv \begin{cases} 0 & \text{nếu} \quad F \\ g^{**} - c & \text{ngược lại} \end{cases}$$

(109)
$$\boldsymbol{\sigma}' \equiv \begin{cases} \boldsymbol{\sigma} & \text{nếu} \quad F \ \vee \ \boldsymbol{\sigma}^{**} = \varnothing \\ \boldsymbol{\sigma}^{**} \quad \text{ngoại trừ:} & \\ \quad \boldsymbol{\sigma}'[a] = \varnothing & \text{nếu} \quad \texttt{DEAD}(\boldsymbol{\sigma}^{**}, a) \\ \boldsymbol{\sigma}^{**} \quad \text{ngoại trừ:} & \\ \quad \boldsymbol{\sigma}'[a]_{\mathrm{c}} = \texttt{KEC}(\mathbf{o}) & \text{ngược lại} \end{cases}$$

(110)
$$A' \equiv \begin{cases} A^{*} & \text{nếu} \quad F \ \vee \ \boldsymbol{\sigma}^{**} = \varnothing \\ A^{**} & \text{ngược lại} \end{cases}$$

(111)
$$z \equiv \begin{cases} 0 & \text{nếu} \quad F \ \vee \ \boldsymbol{\sigma}^{**} = \varnothing \\ 1 & \text{ngược lại} \end{cases}$$

trong đó

(112)
$$\begin{aligned} F \equiv & \big(\boldsymbol{\sigma}[a] \neq \varnothing \ \wedge \ \big(\boldsymbol{\sigma}[a]_c \neq \texttt{KEC}(()) \vee \boldsymbol{\sigma}[a]_n \neq 0\big)\big) \quad \vee \\ & (\boldsymbol{\sigma}^{**} = \varnothing \ \wedge \ \mathbf{o} = \varnothing) \quad \vee \\ & g^{**} < c \quad \vee \\ & \|\mathbf{o}\| > 24576 \quad \vee \\ & \mathbf{o}[0] = \texttt{0xef} \end{aligned}$$

Lưu ý điều kiện cuối cùng của F chỉ ra rằng mã hợp đồng không thể bắt đầu bằng byte `0xef` (xem EIP-3541 của Beregszaszi et al. [2021]).

Ngoại lệ trong xác định của $\boldsymbol{\sigma}'$ quy định rằng $\mathbf{o}$, dãy byte kết quả từ việc thực thi mã khởi tạo, chỉ định nội dung mã nguồn sau cùng (final body code) cho tài khoản mới được tạo ra.

Lưu ý rằng ý định kết quả sẽ là một hợp đồng mới được tạo thành công với tài sản của nó (its endowment) được chuyển đến, hoặc không có hợp đồng mới và không có chuyển khoản giá trị nào cả. Ngoài ra, quan sát rằng nếu việc thực thi mã khởi tạo reverts ($\boldsymbol{\sigma}^{**} = \varnothing \ \wedge \ \mathbf{o} \neq \varnothing$), kết quả gas g' không bị tiêu hao (miễn là không có ngoại lệ khác), nhưng không có tài khoản mới được tạo ra.

7.1. **Sự tinh tế.** Lưu ý rằng trong khi mã khởi tạo đang thực thi, địa chỉ mới được tạo ra nhưng không có nội dung mã nguồn nội tại[10]. Do đó, bất kỳ cuộc gọi tin nhắn nào được nhận vào nó trong thời gian này đều không gây ra việc thực thi mã. Nếu việc thực hiện mã khởi tạo kết thúc bằng một lệnh SELFDESTRUCT, thì vấn đề không có ý nghĩa vì tài khoản sẽ bị xóa trước khi giao dịch được hoàn thành. Đối với một mã STOP bình thường, hoặc nếu mã trả về rỗng, thì trạng thái sẽ bị để lại với một tài khoản zombie và bất kỳ số dư còn lại sẽ bị khóa vào tài khoản mãi mãi.

[10]Trong quá trình thực thi mã khởi tạo, `EXTCODESIZE` trên địa chỉ nên trả về 0, tức là độ dài của mã của tài khoản trong khi `CODESIZE` sẽ trả về độ dài của mã khởi tạo (như được định nghĩa trong H.2).

8. Cuộc Gọi Tin Nhắn (Message Call)

Trong trường hợp thực hiện cuộc gọi tin nhắn, một số tham số được yêu cầu: người gửi (s), người khởi tạo giao dịch (transaction originator) (o), người nhận (r), tài khoản có mã (code) sẽ được thực thi (c, thường giống với người nhận), lượng gas khả dụng (g), giá trị (v) và giá gas hiệu quả (p) cùng với một mảng byte có độ dài tùy ý, $\mathbf{d}$, dữ liệu đầu vào của cuộc gọi, độ sâu hiện tại của ngăn xếp cuộc gọi tin nhắn/tạo hợp đồng (e) và cuối cùng là quyền được phép thực hiện sửa đổi trạng thái (w).

Ngoài việc đánh giá sang trạng thái mới và trạng thái con tích lũy của giao dịch, cuộc gọi tin nhắn cũng có một thành phần phụ thêm - dữ liệu đầu ra được biểu thị bằng mảng byte $\mathbf{o}$. Điều này được bỏ qua khi thực hiện các giao dịch, tuy nhiên cuộc gọi tin nhắn có thể được bắt đầu do việc thực thi mã VM và trong trường hợp này thông tin này được sử dụng.

$$(113)\quad (\boldsymbol{\sigma}', g', A', z, \mathbf{o}) \equiv \Theta(\boldsymbol{\sigma}, A, s, o, r, c, g, p, v, \tilde{v}, \mathbf{d}, e, w)$$

Lưu ý rằng chúng ta cần phải phân biệt giữa giá trị cần chuyển, v, và giá trị xuất hiện trong ngữ cảnh thực thi, $\tilde{v}$, cho lệnh DELEGATECALL.

Chúng ta định nghĩa $\boldsymbol{\sigma}_1$, trạng thái chuyển tiếp đầu tiên, như là trạng thái ban đầu nhưng có giá trị được chuyển từ người gửi tới người nhận:

$$(114)\qquad \boldsymbol{\sigma}_1[r]_{\mathrm{b}} \equiv \boldsymbol{\sigma}[r]_{\mathrm{b}} + v \quad \wedge \quad \boldsymbol{\sigma}_1[s]_{\mathrm{b}} \equiv \boldsymbol{\sigma}[s]_{\mathrm{b}} - v$$

trừ khi $s = r$.

Trong toàn bộ công việc hiện tại, giả sử nếu $\boldsymbol{\sigma}_1[r]$ ban đầu không được định nghĩa, nó sẽ được tạo ra như một tài khoản không có mã (code) hoặc trạng thái, có số dư và nonce bằng không. Do đó, phương trình trước đó nên được hiểu như:

$$(115)\qquad \boldsymbol{\sigma}_1 \equiv \boldsymbol{\sigma}'_1 \quad \text{ngoại trừ:}$$

$$(116)\qquad \boldsymbol{\sigma}_1[s] \equiv \begin{cases} \varnothing & \text{nếu } \boldsymbol{\sigma}'_1[s] = \varnothing \ \wedge \ v = 0 \\ \mathbf{a}_1 & \text{ngược lại} \end{cases}$$

$$(117)\qquad \mathbf{a}_1 \equiv \left(\boldsymbol{\sigma}'_1[s]_{\mathrm{n}}, \boldsymbol{\sigma}'_1[s]_{\mathrm{b}} - v, \boldsymbol{\sigma}'_1[s]_{\mathbf{s}}, \boldsymbol{\sigma}'_1[s]_{\mathrm{c}}\right)$$

$$(118)\qquad \text{và} \quad \boldsymbol{\sigma}'_1 \equiv \boldsymbol{\sigma} \quad \text{ngoại trừ:}$$

$$(119)\qquad \begin{cases} \boldsymbol{\sigma}'_1[r] \equiv (0, v, \mathtt{TRIE}(\varnothing), \mathtt{KEC}(())) & \text{nếu} \quad \boldsymbol{\sigma}[r] = \varnothing \wedge v \neq 0 \\ \boldsymbol{\sigma}'_1[r] \equiv \varnothing & \text{nếu} \quad \boldsymbol{\sigma}[r] = \varnothing \wedge v = 0 \\ \boldsymbol{\sigma}'_1[r] \equiv \mathbf{a}'_1 & \text{ngược lại} \end{cases}$$

$$(120)\qquad \mathbf{a}'_1 \equiv (\boldsymbol{\sigma}[r]_{\mathrm{n}}, \boldsymbol{\sigma}[r]_{\mathrm{b}} + v, \boldsymbol{\sigma}[r]_{\mathbf{s}}, \boldsymbol{\sigma}[r]_{\mathrm{c}})$$

Mã (code) liên kết với tài khoản (được xác định là đoạn mã mà băm Keccak-256 của nó là $\boldsymbol{\sigma}[c]_{\mathrm{c}}$) được thực thi theo mô hình thực thi (xem phần 9). Giống như việc tạo hợp đồng, nếu quá trình thực hiện kết thúc một cách ngoại lệ (tức là do hết nguồn gas, tràn ngăn xếp, đích nhảy không hợp lệ hoặc chỉ thị không hợp lệ), thì không có gas nào được hoàn trả cho người gọi và trạng thái sẽ quay về điểm ngay trước khi chuyển số dư (tức là $\boldsymbol{\sigma}$).

$$(121)\qquad \boldsymbol{\sigma}' \equiv \begin{cases} \boldsymbol{\sigma} & \text{nếu} \quad \boldsymbol{\sigma}^{**} = \varnothing \\ \boldsymbol{\sigma}^{**} & \text{ngược lại} \end{cases}$$

$$(122)\qquad g' \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\sigma}^{**} = \varnothing \wedge \\ & \quad \mathbf{o} = \varnothing \\ g^{**} & \text{ngược lại} \end{cases}$$

$$(123)\qquad A' \equiv \begin{cases} A & \text{nếu} \quad \boldsymbol{\sigma}^{**} = \varnothing \\ A^{**} & \text{ngược lại} \end{cases}$$

$$(124)\qquad z \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\sigma}^{**} = \varnothing \\ 1 & \text{ngược lại} \end{cases}$$

$$(125)\qquad (\boldsymbol{\sigma}^{**}, g^{**}, A^{**}, \mathbf{o}) \equiv \Xi$$

$$(126)\qquad I_{\mathrm{a}} \equiv r$$

$$(127)\qquad I_{\mathrm{o}} \equiv o$$

$$(128)\qquad I_{\mathrm{p}} \equiv p$$

$$(129)\qquad I_{\mathbf{d}} \equiv \mathbf{d}$$

$$(130)\qquad I_{\mathrm{s}} \equiv s$$

$$(131)\qquad I_{\mathrm{v}} \equiv \tilde{v}$$

$$(132)\qquad I_{\mathrm{e}} \equiv e$$

$$(133)\qquad I_{\mathrm{w}} \equiv w$$

trong đó

$$(134)\qquad \Xi \equiv \begin{cases} \Xi_{\mathtt{ECREC}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 1 \\ \Xi_{\mathtt{SHA256}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 2 \\ \Xi_{\mathtt{RIP160}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 3 \\ \Xi_{\mathtt{ID}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 4 \\ \Xi_{\mathtt{EXPMOD}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 5 \\ \Xi_{\mathtt{BN\_ADD}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 6 \\ \Xi_{\mathtt{BN\_MUL}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 7 \\ \Xi_{\mathtt{SNARKV}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 8 \\ \Xi_{\mathtt{BLAKE2\_F}}(\boldsymbol{\sigma}_1, g, A, I) & \text{nếu} \quad c = 9 \\ \Xi(\boldsymbol{\sigma}_1, g, A, I) & \text{ngược lại} \end{cases}$$

và

$$(135)\qquad \mathtt{KEC}(I_{\mathbf{b}}) = \boldsymbol{\sigma}[c]_{\mathrm{c}}$$

Giả định rằng người dùng sẽ lưu trữ cặp $(\mathtt{KEC}(I_{\mathbf{b}}), I_{\mathbf{b}})$ tại một thời điểm trước để làm cho việc xác định $I_{\mathbf{b}}$ trở nên khả thi.

Có thể thấy, có chín trường hợp ngoại lệ đối với việc sử dụng khung thực thi chung Ξ để đánh giá cuộc gọi tin nhắn: đây là các hợp đồng gọi là 'được biên dịch trước (precompiled)', được thiết kế như một phần kiến trúc sơ bộ có thể sau này trở thành *tiện ích mở rộng gốc (native extensions)*. Các hợp đồng tại các địa chỉ từ 1 đến 9 thực thi hàm khôi phục khóa công khai của đường cong elliptic, giải thuật băm 256-bit SHA2, giải thuật băm 160-bit RIPEMD, hàm nhận diện, lũy thừa modular chính xác tùy ý, phép cộng đường cong elliptic, phép nhân vô hướng đường elliptic, kiểm tra cặp đường elliptic và hàm nén BLAKE2 `F` tương ứng. Định nghĩa đầy đủ của chúng được trình bày trong Phụ lục E. Chúng ta ký hiệu tập hợp địa chỉ của các hợp đồng 'được biên dịch trước (precompiled)' là π:

$$(136)\qquad \pi \equiv \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

9. Mô Hình Thực Thi (Execution Model)

Mô hình thực thi chỉ định cách trạng thái hệ thống thay đổi dựa trên một loạt các chỉ thị mã bytecode và một bộ (tuple) nhỏ dữ liệu môi trường. Điều này được chỉ định thông qua mô hình chính thức của máy trạng thái ảo, được gọi là Máy ảo Ethereum (EVM). Đây là một máy *gần như* Turing-complete; việc *gần như* này đến từ việc tính toán được giới hạn tự nhiên thông qua một tham số, *gas*, giới hạn tổng lượng tính toán được thực hiện.

9.1. **Cơ bản.** EVM là một kiến trúc đơn giản dựa trên ngăn xếp. Kích thước từ (word) của máy (và do đó kích thước của các phần tử trong ngăn xếp) là 256-bit. Lựa chọn này được chọn để hỗ trợ giải thuật băm Keccak-256 và tính toán đường cong elliptic. Mô hình bộ nhớ là một mảng byte được địa chỉ theo từng từ (word). Ngăn xếp có kích thước tối đa là 1024. Máy cũng có một mô hình lưu trữ độc lập; điều này tương tự trong khái niệm với bộ nhớ, nhưng thay vì một mảng byte, nó là một mảng từ (word) có thể được truy cập theo từng từ (word). Không giống như bộ nhớ (memory) dễ thay đổi, kho lưu trữ (storage) không dễ thay đổi và được duy trì như một phần của trạng thái hệ thống. Tất cả các vị trí trong cả bộ lưu trữ và bộ nhớ ban đầu đều được xác định rõ ràng bằng 0

Máy không tuân theo kiến trúc von Neumann tiêu chuẩn. Thay vì lưu trữ mã chương trình trong bộ nhớ hoặc kho lưu trữ có thể truy cập chung, nó được lưu trữ riêng biệt trong một ROM ảo chỉ có thể tương tác thông qua một chỉ thị chuyên biệt.

Máy có thể có quá trình thực thi ngoại lệ vì một số lý do, bao gồm việc tràn ngăn xếp và chỉ thị không hợp lệ. Giống như ngoại lệ hết gas, chúng không giữ nguyên các thay đổi trạng thái. Thay vào đó, máy sẽ dừng ngay lập tức và báo cáo vấn đề cho tác nhân thực thi (bộ xử lý giao dịch hoặc, theo cách đệ quy, môi trường thực thi tạo ra (spawning)) cái mà sẽ giải quyết nó một cách riêng biệt.

9.2. **Tổng quan về phí.** Các phí (được tính bằng gas) được thu trong ba trường hợp khác nhau, tất cả đều là điều kiện tiên quyết cho việc thực hiện một hoạt động. Trường hợp đầu tiên và phổ biến nhất là phí nội tại cho việc tính toán của hoạt động (xem Phụ lục G). Thứ hai, gas có thể bị trừ để tạo thành thanh toán cho một cuộc gọi tin nhắn phụ hoặc tạo hợp đồng; điều này là một phần của thanh toán cho CREATE, CREATE2, CALL và CALLCODE. Cuối cùng, gas có thể được thanh toán do sự tăng trong việc sử dụng bộ nhớ.

Trong quá trình thực thi của một tài khoản, tổng phí cho việc sử dụng bộ nhớ phải thanh toán tỷ lệ với bội số nhỏ nhất của 32 byte cần thiết để bao gồm tất cả các chỉ số bộ nhớ (cho việc đọc hoặc ghi) trong phạm vi. Điều này được thanh toán theo cơ sở just-in-time; do đó, tham chiếu đến một khu vực bộ nhớ ít nhất là 32 byte lớn hơn bất kỳ bộ nhớ nào đã được lập chỉ mục trước đó sẽ chắc chắn dẫn đến một phí sử dụng bộ nhớ bổ sung. Do phí này, khả năng cao là địa chỉ sẽ không bao giờ vượt quá giới hạn 32-bit. Tuy nhiên, các triển khai phải có khả năng quản lý tình huống này.

Phí lưu trữ có một hành vi có chút tinh tế — để khuyến khích việc giảm thiểu việc sử dụng bộ nhớ (tương ứng trực tiếp với cơ sở dữ liệu trạng thái lớn hơn trên tất cả các nút), phí thực hiện cho một hoạt động xóa một mục trong bộ nhớ không chỉ được miễn phí, mà còn được cấp một khoản hoàn trả đủ điều kiện; thực tế, hoàn trả này được thanh toán trước vì việc sử dụng ban đầu của một vị trí bộ nhớ có giá đắt hơn nhiều so với sử dụng bình thường.

Xem Phụ lục H để biết định nghĩa chặt chẽ về chi phí gas của EVM.

9.3. **Môi trường Thực thi.** Ngoài trạng thái hệ thống $\boldsymbol{\sigma}$, gas còn lại cho việc tính toán g, và substate đã được tích lũy A, có một số thông tin quan trọng được sử dụng trong môi trường thực thi mà tác nhân thực thi phải cung cấp; những thông tin này được chứa trong bộ (tuple) I:

- I_{a}, địa chỉ của tài khoản sở hữu mã nguồn đang thực thi.
- I_{o}, địa chỉ người gửi giao dịch gốc của quá trình thực thi này.
- I_{p}, giá gas đã thanh toán bởi người ký của giao dịch gốc của quá trình thực thi. Đây được định nghĩa như là giá gas hiệu quả p trong phần 6.
- $I_{\mathbf{d}}$, mảng byte là dữ liệu đầu vào của quá trình thực thi này; nếu tác nhân thực thi là một giao dịch, điều này sẽ là dữ liệu của giao dịch.
- I_{s}, địa chỉ của tài khoản gây ra mã nguồn đang thực thi; nếu tác nhân thực thi là một giao dịch, điều này sẽ là người gửi giao dịch.
- I_{v}, giá trị, tính bằng Wei, được chuyển đến tài khoản này như là một phần của cùng một quy trình với việc thực thi; nếu tác nhân thực thi là một giao dịch, điều này sẽ là giá trị của giao dịch.
- $I_{\mathbf{b}}$, mảng byte là mã máy để thực thi.
- I_{H}, tiêu đề khối của khối hiện tại.
- I_{e}, độ sâu của cuộc gọi tin nhắn hiện tại hoặc tạo hợp đồng (tức là số lượng CALL hoặc CREATE(2) đang được thực thi hiện tại).
- I_{w}, quyền được phép thực hiện sửa đổi vào trạng thái.

Mô hình thực thi định nghĩa hàm Ξ, có thể tính toán trạng thái kết quả $\boldsymbol{\sigma}'$, gas còn lại g', substate đã tích lũy A', và kết quả đầu ra, $\mathbf{o}$, dựa trên những định nghĩa này. Trong ngữ cảnh hiện tại, chúng ta sẽ định nghĩa nó như sau:

$$(\boldsymbol{\sigma}', g', A', \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}, g, A, I) \tag{137}$$

nơi chúng ta sẽ nhớ rằng A, trạng thái con đã tích lũy, được định nghĩa trong phần 6.1.

9.4. **Tổng Quan về Thực Thi.** Chúng ta phải định nghĩa hàm Ξ bây giờ. Trong hầu hết các triển khai thực tế, điều này sẽ được mô hình như là một tiến triển lặp của cặp bao gồm trạng thái hệ thống đầy đủ, $\boldsymbol{\sigma}$, và trạng thái máy, $\boldsymbol{\mu}$. Theo quy ước, chúng ta định nghĩa nó theo cách đệ quy với một hàm X. Điều này sử dụng một hàm lặp O (định nghĩa kết quả của một chu kỳ duy nhất của máy trạng thái) cùng với các hàm Z xác định xem trạng thái hiện tại có phải là trạng thái dừng ngoại lệ của máy không và H, chỉ định dữ liệu đầu ra của chỉ thị nếu và chỉ nếu trạng thái hiện tại là trạng thái dừng bình thường của máy.

Dãy trống, được ký hiệu là (), không bằng với tập trống, được ký hiệu là $\varnothing$; điều này quan trọng khi giải thích đầu ra của H, nó được đánh giá là $\varnothing$ khi việc thực thi sẽ tiếp tục nhưng là một loạt (có thể là trống) khi việc thực thi

nên dừng lại.

$$\Xi(\boldsymbol{\sigma}, g, A, I) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}'_{\mathrm{g}}, A', \mathbf{o}) \tag{138}$$

$$(\boldsymbol{\sigma}', \boldsymbol{\mu}', A', ..., \mathbf{o}) \equiv X\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) \tag{139}$$

$$\boldsymbol{\mu}_{\mathrm{g}} \equiv g \tag{140}$$

$$\boldsymbol{\mu}_{\mathrm{pc}} \equiv 0 \tag{141}$$

$$\boldsymbol{\mu}_{\mathbf{m}} \equiv (0, 0, ...) \tag{142}$$

$$\boldsymbol{\mu}_{\mathrm{i}} \equiv 0 \tag{143}$$

$$\boldsymbol{\mu}_{\mathbf{s}} \equiv () \tag{144}$$

$$\boldsymbol{\mu}_{\mathbf{o}} \equiv () \tag{145}$$

(146)
$$X\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) \equiv \begin{cases} \big(\varnothing, \boldsymbol{\mu}, A, I, \varnothing\big) & \text{nếu} \quad Z(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \\ \big(\varnothing, \boldsymbol{\mu}', A, I, \mathbf{o}\big) & \text{nếu} \quad w = \text{REVERT} \\ O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \cdot \mathbf{o} & \text{nếu} \quad \mathbf{o} \neq \varnothing \\ X\big(O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) & \text{ngược lại} \end{cases}$$

where

$$\mathbf{o} \equiv H(\boldsymbol{\mu}, I) \tag{147}$$

$$(a, b, c, d) \cdot e \equiv (a, b, c, d, e) \tag{148}$$

$$\boldsymbol{\mu}' \equiv \boldsymbol{\mu} \text{ ngoại trừ:} \tag{149}$$

$$\boldsymbol{\mu}'_{\mathrm{g}} \equiv \boldsymbol{\mu}_{\mathrm{g}} - C(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \tag{150}$$

Lưu ý rằng khi chúng ta đánh giá Ξ, chúng ta bỏ đi phần tử thứ tư I' và trích xuất gas còn lại $\boldsymbol{\mu}'_{\mathrm{g}}$ từ kết quả trạng thái máy $\boldsymbol{\mu}'$.

Vì vậy, X được lặp (đệ quy ở đây, nhưng thông thường các triển khai sẽ sử dụng một vòng lặp đơn giản) cho đến khi Z trở thành đúng (true), chỉ ra rằng trạng thái hiện tại là ngoại lệ và máy phải bị dừng lại và mọi thay đổi được loại bỏ hoặc cho đến khi H trở thành một loạt (thay vì tập hợp trống) chỉ ra rằng máy đã đạt đến một sự dừng được kiểm soát.

9.4.1. *Trạng Thái Máy.* Trạng thái máy $\boldsymbol{\mu}$ được định nghĩa như là bộ (tuple) $(g, pc, \mathbf{m}, i, \mathbf{s})$ là gas sẵn có, bộ đếm chương trình $pc \in \mathbb{N}_{256}$, nội dung bộ nhớ, số lượng từ (word) hoạt động trong bộ nhớ (đếm liên tục từ vị trí 0), và nội dung ngăn xếp. Nội dung bộ nhớ $\boldsymbol{\mu}_{\mathbf{m}}$ là một loạt các số 0 có kích thước 2^{256}.

Để dễ đọc, các chỉ thị ghi nhớ (mnemonic), viết bằng chữ hoa nhỏ (vd: ADD), nên được hiểu như là các giá trị số tương ứng của chúng; bảng đầy đủ của các chỉ thị và chi tiết của chúng được cho trong Phụ Lục H.

Đối với mục đích định nghĩa Z, H, và O, chúng ta định nghĩa w là hoạt động hiện tại sẽ được thực thi:

$$w \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\mathrm{pc}}] & \text{nếu} \quad \boldsymbol{\mu}_{\mathrm{pc}} < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{ngược lại} \end{cases} \tag{151}$$

Chúng ta cũng giả sử các giá trị cố định của δ và α, chỉ thị số phần tử ngăn xếp bị loại bỏ và được thêm vào, cả hai có thể đăng ký được trên chỉ thị và một hàm chi phí chỉ thị ước tính toàn bộ chi phí C, theo gas, của việc thực hiện chỉ thị đã cho.

9.4.2. *Dừng ngoại lệ (Exceptional Halting).* Hàm dừng ngoại lệ Z được định nghĩa như sau:

(152)
$$\begin{aligned} Z(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \equiv \; & \boldsymbol{\mu}_g < C(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \quad \vee \\ & \delta_w = \varnothing \quad \vee \\ & \|\boldsymbol{\mu}_{\mathbf{s}}\| < \delta_w \quad \vee \\ & (w = \text{JUMP} \ \wedge \ \boldsymbol{\mu}_{\mathbf{s}}[0] \notin D(I_{\mathbf{b}})) \quad \vee \\ & (w = \text{JUMPI} \ \wedge \ \boldsymbol{\mu}_{\mathbf{s}}[1] \neq 0 \ \wedge \\ & \quad \boldsymbol{\mu}_{\mathbf{s}}[0] \notin D(I_{\mathbf{b}})) \quad \vee \\ & (w = \text{RETURNDATACOPY} \ \wedge \\ & \quad \boldsymbol{\mu}_{\mathbf{s}}[1] + \boldsymbol{\mu}_{\mathbf{s}}[2] > \|\boldsymbol{\mu}_{\mathbf{o}}\|) \quad \vee \\ & \|\boldsymbol{\mu}_{\mathbf{s}}\| - \delta_w + \alpha_w > 1024 \quad \vee \\ & (\neg I_{\mathrm{w}} \ \wedge \ W(w, \boldsymbol{\mu})) \quad \vee \\ & (w = \text{SSTORE} \ \wedge \ \boldsymbol{\mu}_g \leqslant G_{\text{callstipend}}) \end{aligned}$$

trong đó

(153)
$$\begin{aligned} W(w, \boldsymbol{\mu}) \equiv \; & w \in \{\text{CREATE}, \text{CREATE2}, \text{SSTORE}, \\ & \text{SELFDESTRUCT}\} \ \vee \\ & \text{LOG0} \leq w \ \wedge \ w \leq \text{LOG4} \quad \vee \\ & w = \text{CALL} \ \wedge \ \boldsymbol{\mu}_{\mathbf{s}}[2] \neq 0 \end{aligned}$$

Điều này cho biết rằng việc thực thi đang ở trong trạng thái dừng ngoại lệ nếu không đủ gas, nếu chỉ thị không hợp lệ (và do đó chỉ số dưới δ của nó là không xác định), nếu không đủ mục ngăn xếp, nếu một đích JUMP/JUMPI không hợp lệ, kích thước ngăn xếp mới lớn hơn 1024 hoặc cố thay đổi trạng thái trong một cuộc gọi tĩnh (static call). Người đọc tinh tế sẽ nhận ra rằng điều này ngụ ý rằng không có chỉ thị nào, thông qua việc thực thi nó, có thể gây ra một sự dừng đặc biệt. Ngoài ra, việc thực thi đang ở trong trạng thái dừng ngoại lệ nếu gas còn lại trước khi thực hiện một chỉ thị SSTORE ít hơn hoặc bằng số gas ưu đãi cuộc gọi (call stipend) $G_{\text{callstipend}}$ – xem EIP-2200 của Tang [2019] để biết thêm thông tin.

9.4.3. *Độ hợp lệ của Đích Nhảy (Jump Destination Validity).* Chúng ta trước đây đã sử dụng D làm hàm để xác định tập hợp các đích nhảy hợp lệ dựa trên mã nguồn (code) đang chạy. Chúng ta định nghĩa điều này như là bất kỳ vị trí nào trong mã nguồn được chiếm bởi một chỉ thị JUMPDEST.

Tất cả các vị trí như vậy phải nằm trên các ranh giới chỉ thị hợp lệ, thay vì nằm trong phần dữ liệu của các thao tác PUSH, và phải xuất hiện trong phần mã (code) được định nghĩa một cách rõ ràng (thay vì trong thao tác STOP được định nghĩa ngầm theo nó).

Quy ước:

$$D(\mathbf{c}) \equiv D_{\mathrm{J}}(\mathbf{c}, 0) \tag{154}$$

trong đó:

(155)
$$D_{\mathrm{J}}(\mathbf{c}, i) \equiv \begin{cases} \{\} & \text{nếu} \quad i \geqslant \|\mathbf{c}\| \\ \{i\} \cup D_{\mathrm{J}}(\mathbf{c}, N(i, \mathbf{c}[i])) & \\ \qquad \text{nếu} \quad \mathbf{c}[i] = \text{JUMPDEST} & \\ D_{\mathrm{J}}(\mathbf{c}, N(i, \mathbf{c}[i])) & \text{ngược lại} \end{cases}$$

trong đó N là vị trí lệnh hợp lệ tiếp theo trong mã nguồn, bỏ qua dữ liệu của một lệnh PUSH nếu có:

(156)
$$N(i, w) \equiv \begin{cases} i + w - \text{PUSH1} + 2 & \\ \qquad \text{nếu} \quad w \in [\text{PUSH1}, \text{PUSH32}] & \\ i + 1 & \text{ngược lại} \end{cases}$$

9.4.4. *Dừng bình thường (Normal Halting).* Hàm dừng bình thường H được định nghĩa như sau:

(157)
$$H(\boldsymbol{\mu}, I) \equiv \begin{cases} H_{\text{RETURN}}(\boldsymbol{\mu}) & \text{nếu} \quad w \in \{\text{RETURN}, \text{REVERT}\} \\ () & \text{nếu} \quad w \in \{\text{STOP}, \text{SELFDESTRUCT}\} \\ \varnothing & \text{ngược lại} \end{cases}$$

Các hoạt động dừng trả dữ liệu, RETURN và REVERT, có một hàm đặc biệt H_{RETURN}. Lưu ý sự khác biệt giữa dãy (sequence) trống và tập hợp (set) trống như đã thảo luận ở đây.

9.5. **Chu Kỳ Thực Thi (The Execution Cycle).** Các phần tử ngăn xếp được thêm hoặc loại bỏ từ phần bên trái, phần được lập chỉ mục thấp hơn của loạt phần tử; tất cả các phần tử khác không thay đổi:

$$O\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}', A', I) \tag{158}$$
$$\Delta \equiv \alpha_w - \delta_w \tag{159}$$
$$\|\boldsymbol{\mu}'_{\mathbf{s}}\| \equiv \|\boldsymbol{\mu}_{\mathbf{s}}\| + \Delta \tag{160}$$
$$\forall x \in [\alpha_w, \|\boldsymbol{\mu}'_{\mathbf{s}}\|) : \boldsymbol{\mu}'_{\mathbf{s}}[x] \equiv \boldsymbol{\mu}_{\mathbf{s}}[x - \Delta] \tag{161}$$

Gas được giảm trừ bởi chi phí gas của chỉ thị và đối với hầu hết các chỉ thị, bộ đếm chương trình sẽ tăng lên trong mỗi chu kỳ, với ba ngoại lệ, chúng ta giả định một hàm J, được đăng ký bởi một trong hai chỉ thị, cái sẽ đánh giá giá trị tương ứng:

$$\boldsymbol{\mu}'_{\text{g}} \equiv \boldsymbol{\mu}_{\text{g}} - C(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \tag{162}$$
$$\boldsymbol{\mu}'_{\text{pc}} \equiv \begin{cases} J_{\text{JUMP}}(\boldsymbol{\mu}) & \text{nếu} \quad w = \text{JUMP} \\ J_{\text{JUMPI}}(\boldsymbol{\mu}) & \text{nếu} \quad w = \text{JUMPI} \\ N(\boldsymbol{\mu}_{\text{pc}}, w) & \text{ngược lại} \end{cases} \tag{163}$$

Nói chung, chúng ta giả sử bộ nhớ, trạng thái con tích lũy và trạng thái hệ thống không thay đổi:

$$\boldsymbol{\mu}'_{\mathbf{m}} \equiv \boldsymbol{\mu}_{\mathbf{m}} \tag{164}$$
$$\boldsymbol{\mu}'_{\text{i}} \equiv \boldsymbol{\mu}_{\text{i}} \tag{165}$$
$$A' \equiv A \tag{166}$$
$$\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma} \tag{167}$$

Tuy nhiên, các chỉ thị thường thay đổi một hoặc vài thành phần của các giá trị này. Các thành phần đã được sửa đổi được liệt kê theo từng chỉ thị trong Phụ lục H, cùng với giá trị cho α và δ và mô tả chính thức về yêu cầu gas.

10. Chuyển đổi sang bằng chứng cổ phần

Hard fork *Paris* đã thay đổi cơ chế đồng thuận cơ bản của Ethereum từ bằng chứng làm việc sang bằng chứng cổ phần.

Không giống như tất cả các hard fork trước đó của Ethereum, *Paris* không được xác định để xảy ra ở một độ cao cụ thể của khối, mà thay vào đó là sau khi đạt *tổng độ khó trạm cuối* được chỉ định. Độ khó tổng được dùng thay vì độ cao khối để tránh tình huống mà một phần nhỏ của sức mạnh hash có thể tạo ra một fork độc hại có thể đua để đáp ứng yêu cầu độ cao khối và đưa ra khối bằng chứng cổ phần đầu tiên.

Do đó, *khối trạm cuối*, là khối chứng bằng chứng làm việc cuối cùng trước khi hard fork *Paris* có hiệu lực, được xác định có:

$$B_{\text{t}} \geqslant 58750000000000000000000 \tag{168}$$
$$P(B_{\text{H}})_{\text{t}} < 58750000000000000000000 \tag{169}$$

trong đó B_{t} là độ khó tổng của khối B và $P(B_{\text{H}})_{\text{t}}$ là độ khó tổng của khối cha của nó.

Độ khó tổng cho một khối bằng chứng làm việc (trước *Paris*) được định nghĩa theo đệ quy như sau:

$$B_{\text{t}} \equiv P(B_{\text{H}})_{\text{t}} + H_{\text{d}} \tag{170}$$

trong đó H_{d} là độ khó của khối hiện tại B.

Khi đạt đến khối trạm cuối, các khối mới sẽ được xử lý bởi Beacon Chain.

11. Cây Khối Đến Chuỗi Khối (Blocktree to Blockchain)

Trước khi chuyển đổi sang bằng chứng cổ phần tại hard fork *Paris*, chuỗi khối cổ điển được định nghĩa là đường dẫn khối có độ khó tổng lớn nhất, được định nghĩa trong phần 10 như B_{t}.

Sau khi đạt đến *khối kết thúc* được mô tả trong phần 10, quy tắc về *độ khó tổng* lớn nhất phải được loại bỏ để ủng hộ một quy tắc được biết đến là *LMD Ghost*.[11]

Lưu ý rằng để xác định những khối nào thuộc chuỗi khối Ethereum cổ điển sau hard fork *Paris*, người ta cần có thông tin bổ sung từ Beacon Chain, điều này không được mô tả trong văn bản này. Chúng ta ký hiệu các sự kiện được phát ra bởi Beacon Chain với tiền tố `POS_`.

Mỗi khi có một sự kiện `POS_FORKCHOICE_UPDATED` xuất hiện, bắt đầu từ sự kiện đầu tiên tại *khối chuyển đổi* mô tả trong phần 10, chuỗi khối cổ điển được định nghĩa là chuỗi bắt đầu từ khối genensis và kết thúc tại khối được đề cử bởi sự kiện làm đầu của chuỗi.

Đầu của chuỗi nên được cập nhật nếu và chỉ nếu một sự kiện `POS_FORKCHOICE_UPDATED` được phát ra, trong trường hợp đó, đầu nên được thiết lập thành khối được chỉ định bởi sự kiện đó. Không nên thực hiện cập nhật lạc quan cho đầu của chuỗi.

Sự kiện `POS_FORKCHOICE_UPDATED` cũng tham chiếu đến một khối đã được xác nhận. Khối đã được xác nhận gần đây nhất nên được đặt thành khối này.

Chuỗi khối cổ điển cũng phải chứa một khối với hash và số của *khối kết thúc* được định nghĩa trong phần 10.

12. Hoàn Thiện Khối (Block Finalisation)

Quá trình hoàn thiện một khối bao gồm hai giai đoạn:

(1) xác thực giao dịch;
(2) xác minh trạng thái.

12.1. **Xác thực Giao dịch. gasUsed** đã cho phải tương ứng trung thực với các giao dịch được liệt kê: B_{Hg}, tổng lượng gas sử dụng trong khối, phải bằng với lượng gas đã tích luỹ theo giao dịch cuối cùng:

$$B_{\text{Hg}} = \ell(\mathbf{R})_{\text{u}} \tag{171}$$

[11] *LMD GHOST* bao gồm hai từ viết tắt, "Latest Message Driven" và "Greedy Heaviest-Observed Sub-Tree".

12.2. **Xác thực Trạng thái.** Ta có thể định nghĩa hàm Γ, ánh xạ một khối B vào trạng thái khởi tạo của nó:

(172) $$\Gamma(B) \equiv \begin{cases} \boldsymbol{\sigma}_0 & \text{nếu} \quad P(B_{\mathrm{H}}) = \varnothing \\ \boldsymbol{\sigma}_{\mathrm{i}} : \mathtt{TRIE}(L_{\mathrm{S}}(\boldsymbol{\sigma}_{\mathrm{i}})) = P(B_{\mathrm{H}})_{\mathrm{H_r}} & \text{ngược lại} \end{cases}$$

Ở đây, $\mathtt{TRIE}(L_{\mathrm{S}}(\boldsymbol{\sigma}_{\mathrm{i}}))$ là giá trị hash của nút gốc của một trie trạng thái $\boldsymbol{\sigma}_{\mathrm{i}}$; giả sử rằng các triển khai sẽ lưu trữ giá trị này trong cơ sở dữ liệu trạng thái, điều này là đơn giản và hiệu quả vì cây trie theo bản chất của nó là một cấu trúc dữ liệu không thay đổi.

Cuối cùng, ta định nghĩa Φ, hàm chuyển đổi khối, ánh xạ một khối chưa đầy đủ B thành một khối đầy đủ B':

(173) $$\Phi(B) \equiv B' : \quad B' = B \quad \text{ngoại trừ:}$$

(174) $$B'_{\mathrm{H_r}} = \mathtt{TRIE}(L_{\mathrm{S}}(\Pi(\Gamma(B), B)))$$

Như đã xác định ở đầu công việc hiện tại, Π là hàm chuyển trạng thái, được định nghĩa dựa trên Υ, hàm đánh giá giao dịch.

Như đã chi tiết trước đó, $\mathbf{R}[n]_{\mathrm{z}}$, $\mathbf{R}[n]_{\mathrm{l}}$ và $\mathbf{R}[n]_{\mathrm{u}}$ là mã tình trạng thứ n, logs và tổng gas đã sử dụng sau mỗi giao dịch ($\mathbf{R}[n]_{\mathrm{b}}$, thành phần thứ tư trong bộ ba, đã được định nghĩa dựa trên logs). Ta cũng định nghĩa trạng thái thứ n $\boldsymbol{\sigma}[n]$, đơn giản là trạng thái kết quả từ việc áp dụng giao dịch tương ứng vào trạng thái kết quả từ giao dịch trước đó (hoặc trạng thái khởi tạo của khối trong trường hợp giao dịch đầu tiên):

(175) $$\boldsymbol{\sigma}[n] = \begin{cases} \Gamma(B) & \text{nếu} \quad n < 0 \\ \Upsilon(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n]) & \text{ngược lại} \end{cases}$$

Trong trường hợp của $B_{\mathbf{R}}[n]_{\mathrm{u}}$, ta tiếp cận một cách tương tự, xác định mỗi mục như là gas đã sử dụng trong việc đánh giá giao dịch tương ứng cộng với mục trước đó (hoặc bằng không, nếu đó là giao dịch đầu tiên), cho chúng ta một tổng số đang chạy:

(176) $$\mathbf{R}[n]_{\mathrm{u}} = \begin{cases} 0 & \text{nếu} \quad n < 0 \\ \Upsilon^g(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n]) & \\ \quad +\mathbf{R}[n-1]_{\mathrm{u}} & \text{ngược lại} \end{cases}$$

Đối với $\mathbf{R}[n]_{\mathrm{l}}$, ta sử dụng hàm $\Upsilon^{\mathbf{l}}$ mà ta đã định nghĩa một cách thuận tiện trong hàm thực hiện giao dịch.

(177) $$\mathbf{R}[n]_{\mathrm{l}} = \Upsilon^{\mathbf{l}}(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n])$$

Ta định nghĩa $\mathbf{R}[n]_{\mathrm{z}}$ theo cách tương tự.

(178) $$\mathbf{R}[n]_{\mathrm{z}} = \Upsilon^{\mathrm{z}}(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n])$$

Cuối cùng, ta định nghĩa Π là trạng thái kết quả cuối cùng của giao dịch, $\ell(\boldsymbol{\sigma})$:

(179) $$\Pi(\boldsymbol{\sigma}, B) \equiv \ell(\boldsymbol{\sigma})$$

Như vậy, cơ chế chuyển đổi khối hoàn chỉnh (trước đồng thuận) đã được định nghĩa.

13. Triển Khai Hợp Đồng (Implementing Contracts)

Có một số mô hình kỹ thuật hợp đồng cho phép các hành vi hữu ích cụ thể; hai trong số những điều này chúng ta sẽ thảo luận ngắn gọn là nguồn cấp dữ liệu (data feeds) và số ngẫu nhiên (random numbers).

13.1. **Nguồn Cấp Dữ liệu (Data Feeds).** Một hợp đồng nguồn cấp dữ liệu là một hợp đồng cung cấp một dịch vụ duy nhất: nó cung cấp quyền truy cập vào thông tin từ thế giới bên ngoài Ethereum. Độ chính xác và tính kịp thời của thông tin này không được đảm bảo và đó là nhiệm vụ của một tác giả hợp đồng thứ cấp - hợp đồng sử dụng nguồn cấp dữ liệu - để xác định mức độ tin cậy có thể đặt vào nguồn cấp dữ liệu một cách nào đó.

Mô hình chung bao gồm một hợp đồng duy nhất trong Ethereum, khi được gọi, sẽ trả lời với một số thông tin kịp thời liên quan đến một hiện tượng bên ngoài. Một ví dụ có thể là nhiệt độ hiện tại của thành phố New York. Điều này có thể được triển khai như một hợp đồng trả về giá trị của một số điểm đã biết trong bộ nhớ. Tất nhiên, điểm này trong bộ nhớ phải được duy trì với nhiệt độ chính xác, và do đó, phần thứ hai của mô hình sẽ là một máy chủ bên ngoài chạy một nút Ethereum, và ngay sau khi phát hiện một khối mới, tạo một giao dịch hợp lệ mới, gửi đến hợp đồng, cập nhật giá trị đó trong kho lưu trữ. Mã của hợp đồng sẽ chỉ chấp nhận các cập nhật như vậy từ định danh chứa trên máy chủ đó.

13.2. **Số Ngẫu Nhiên (Random Numbers).** Việc cung cấp số ngẫu nhiên trong một hệ thống xác định là, tất nhiên, một nhiệm vụ không thể thực hiện được. Tuy nhiên, chúng ta có thể xấp xỉ với các số ngẫu nhiên giả tưởng bằng cách sử dụng dữ liệu mà thường không thể biết trước vào thời điểm giao dịch. Dữ liệu như vậy có thể bao gồm hash của khối và địa chỉ người thụ hưởng của khối. Để làm cho việc kiểm soát những giá trị này trở nên khó khăn đối với các trình xác thực (validators) ác ý, bạn nên sử dụng phép toán BLOCKHASH để sử dụng hash của 256 khối trước đó như là số ngẫu nhiên giả tưởng. Đối với một loạt các số như vậy, một giải pháp đơn giản có thể là thêm một lượng hằng số nào đó và băm kết quả.

14. Định hướng tương lai (Future Directions)

Cơ sở dữ liệu trạng thái sẽ không bị buộc phải duy trì tất cả các cấu trúc trie trạng thái quá khứ vào tương lai. Nó sẽ duy trì một tuổi cho mỗi nút và cuối cùng loại bỏ các nút không đủ mới hoặc không phải là điểm tham chiếu. Các điểm tham chiếu, hoặc một tập hợp các nút trong cơ sở dữ liệu cho phép duyệt qua trie trạng thái của một khối cụ thể, có thể được sử dụng để đặt một giới hạn tối đa về lượng tính toán cần thiết để truy xuất bất kỳ trạng thái nào trong toàn bộ chuỗi khối.

Hợp nhất chuỗi khối có thể được sử dụng để giảm lượng khối mà một nút đầy đủ cần phải tải xuống. Một bản lưu trữ nén của cấu trúc trie tại các điểm thời gian cụ thể (có thể là mỗi 10,000 khối) có thể được duy trì bởi mạng ngang hàng, hiệu quả làm lại khối khởi tạo. Điều này sẽ giảm lượng dữ liệu cần tải xuống chỉ thành một bản lưu trữ cộng với một giới hạn tối đa cứng về khối.

Cuối cùng, có thể thực hiện việc nén chuỗi khối: các nút trong trie trạng thái mà không gửi/nhận một giao dịch trong một lượng khối hằng số có thể bị loại bỏ, giảm cả sự rò rỉ Ether và sự tăng trưởng của cơ sở dữ liệu trạng thái.

15. Kết luận (Conclusion)

Chúng ta đã giới thiệu, thảo luận và định nghĩa chính thức giao thức Ethereum. Thông qua giao thức này, đọc giả có thể triển khai một nút trên mạng Ethereum và tham gia cùng những người khác trong một hệ điều hành xã hội

phi tập trung và an toàn. Các hợp đồng có thể được viết để mô tả theo thuật toán và thực hiện tự động các quy tắc tương tác.

16. Sự ghi nhận (Acknowledgements)

Rất cảm ơn Aeron Buchanan đã viết bản sửa đổi *Homestead*, Christoph Jentzsch đã viết thuật toán Ethash và Yoichi Hirai đã thực hiện hầu hết các thay đổi của EIP-150. Bảo trì quan trọng, sửa lỗi hữu ích và đề xuất đã được cung cấp bởi nhiều người khác từ tổ chức Ethereum DEV và cộng đồng Ethereum nói chung, bao gồm Gustav Simonsson, Paweł Bylica, Jutta Steiner, Nick Savers, Viktor Trón, Marko Simovic, Giacomo Tazzari và, tất nhiên, Vitalik Buterin.

17. Tính khả dụng (Availability)

Nguồn của bài báo này được duy trì tại `https://github.com/ethereum/yellowpaper/`. PDF được tạo tự động được đặt tại `https://ethereum.github.io/yellowpaper/paper.pdf`.

References

Jacob Aron. BitCoin software finds new life. *New Scientist*, 213(2847):20, 2012. URL `http://www.sciencedirect.com/science/article/pii/S0262407912601055`.

Adam Back. Hashcash - Amortizable Publicly Auditable Cost-Functions, 2002. URL `http://www.hashcash.org/papers/amortizable.pdf`.

Alex Beregszaszi, Paweł Bylica, Andrei Maiboroda, Alexey Akhunov, Christian Reitwiessner, and Martin Swende. EIP-3541: Reject new contract code starting with the 0xef byte, March 2021. URL `https://eips.ethereum.org/EIPS/eip-3541`.

Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The KECCAK SHA-3 submission, 2011. URL `https://keccak.team/files/Keccak-submission-3.pdf`.

Roman Boutellier and Mareike Heinzen. Pirates, Pioneers, Innovators and Imitators. In *Growth Through Innovation*, pages 85–96. Springer, 2014. URL `https://www.springer.com/gb/book/9783319040158`.

Vitalik Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, 2013. URL `https://github.com/ethereum/wiki/wiki/White-Paper`.

Vitalik Buterin. EIP-2: Homestead hard-fork changes, 2015. URL `https://eips.ethereum.org/EIPS/eip-2`.

Vitalik Buterin. EIP-155: Simple replay attack protection, October 2016. URL `https://eips.ethereum.org/EIPS/eip-155`.

Vitalik Buterin. EIP-1014: Skinny CREATE2, April 2018. URL `https://eips.ethereum.org/EIPS/eip-1014`.

Vitalik Buterin and Martin Swende. EIP-2929: Gas cost increases for state access opcodes, September 2020a. URL `https://eips.ethereum.org/EIPS/eip-2929`.

Vitalik Buterin and Martin Swende. EIP-2930: Optional access lists, August 2020b. URL `https://eips.ethereum.org/EIPS/eip-2930`.

Vitalik Buterin and Martin Swende. EIP-3529: Reduction in refunds, April 2021. URL `https://eips.ethereum.org/EIPS/eip-3529`.

Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. EIP-1559: Fee market change for eth 1.0 chain, 2019. URL `https://eips.ethereum.org/EIPS/eip-1559`.

Nicolas T. Courtois, Marek Grajek, and Rahul Naik. *Optimizing SHA256 in Bitcoin Mining*, pages 131–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44893-9. doi: 10.1007/978-3-662-44893-9_12. URL `https://doi.org/10.1007/978-3-662-44893-9_12`.

B.A. Davey and H.A. Priestley. *Introduction to lattices and order. 2nd ed.* Cambridge: Cambridge University Press, 2nd ed. edition, 2002. ISBN 0-521-78451-4/pbk.

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *In 12th Annual International Cryptology Conference*, pages 139–147, 1992. URL `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf`.

Dankrad Feist, Dmitry Khovratovich, and Marius van der Wijden. EIP-3607: Reject transactions from senders with deployed code, June 2021. URL `https://eips.ethereum.org/EIPS/eip-3607`.

Phong Vo Glenn Fowler, Landon Curt Noll. Fowler–Noll–Vo hash function, 1991. URL `http://www.isthe.com/chongo/tech/comp/fnv/index.html`.

Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 119–132. Springer, 2004. URL `https://www.iacr.org/archive/ches2004/31560117/31560117.pdf`.

Tjaden Hess, Matt Luongo, Piotr Dyraga, and James Hancock. EIP-152: Add BLAKE2 compression function 'F' precompile, October 2016. URL `https://eips.ethereum.org/EIPS/eip-152`.

Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA), 2001. URL `https://web.archive.org/web/20170921160141/http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf`. Accessed 21 September 2017, but the original link was inaccessible on 19 October 2017. Refer to section 6.2 for ECDSAPUBKEY, and section 7 for ECDSASIGN and ECDSARECOVER.

Sergio Demian Lerner. Strict Memory Hard Hashing Functions, 2014. URL `http://www.hashcash.org/papers/memohash.pdf`.

Mark Miller. The Future of Law. In *paper delivered at the Extro 3 Conference (August 9)*, 1997. URL `https://drive.google.com/file/d/0Bw0VXJKBgYPMS0J2VGIyWWlocms/edit?usp=sharing`.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL `http://www.bitcoin.org/bitcoin.pdf`.

Meni Rosenfeld, Yoni Assia, Vitalik Buterin, m liorhakiLior, Oded Leiba, Assaf Shomer, and Eliran Zach. Colored Coins Protocol Specification, 2012. URL `https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification`.

Markku-Juhani Saarinen and Jean-Philippe Aumasson. RFC 7693: The BLAKE2 cryptographic hash and message authentication code (MAC), November 2015. URL `https://tools.ietf.org/html/rfc7693`.

Simon Sprankel. Technical Basis of Digital Currencies, 2013. URL http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf.

Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997. URL http://firstmonday.org/ojs/index.php/fm/article/view/548.

Wei Tang. EIP-2200: Structured definitions for net gas metering, 2019. URL https://eips.ethereum.org/EIPS/eip-2200.

Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gün Sirer. KARMA: A secure economic framework for peer-to-peer resource sharing, 2003. URL https://www.cs.cornell.edu/people/egs/papers/karma.pdf.

J. R. Willett. MasterCoin Complete Specification, 2013. URL https://github.com/mastercoin-MSC/spec.

Micah Zoltu. EIP-2718: Typed transaction envelope, June 2020. URL https://eips.ethereum.org/EIPS/eip-2718.

Appendix A. Thuật ngữ (Terminology)

Diễn viên bên ngoài (External Actor): Người hoặc thực thể khác có khả năng tương tác với một nút Ethereum, nhưng nằm ngoài thế giới của Ethereum. Có thể tương tác với Ethereum thông qua việc gửi giao dịch được ký và kiểm tra chuỗi khối cùng trạng thái liên quan. Có một (hoặc nhiều) Tài khoản nội tại.

Địa chỉ (Address): Mã 160 bit được sử dụng để xác định Tài khoản.

Tài khoản (Account): Tài khoản có một số dư và số giao dịch nội tại được duy trì như một phần của trạng thái Ethereum. Chúng cũng có một EVM Code (có thể là rỗng) và một Trạng thái Lưu trữ (có thể là rỗng) liên quan. Mặc dù là đồng nhất, nhưng có ý nghĩa khi phân biệt giữa hai loại tài khoản thực tế: những tài khoản có EVM Code liên quan rỗng (do đó, số dư của tài khoản được kiểm soát, nếu có, bởi một thực thể bên ngoại) và những tài khoản có EVM Code liên quan không rỗng (do đó, tài khoản đại diện cho một Đối tượng Tự trị). Mỗi Tài khoản có một Địa chỉ duy nhất để xác định nó.

Giao dịch (Transaction): Một phần dữ liệu, được ký bởi một External Actor. Nó đại diện cho một Message hoặc một đối tượng Tự trị mới. Giao dịch được ghi vào mỗi khối của chuỗi khối.

Đối tượng Tự trị (Autonomous Object): Một đối tượng khả tưởng tồn tại chỉ trong trạng thái giả định của Ethereum. Có một địa chỉ nội tại và do đó có một tài khoản liên quan; tài khoản sẽ có EVM Code liên quan không rỗng. Được tích hợp chỉ là Trạng thái Lưu trữ của tài khoản đó.

Trạng thái Lưu trữ (Storage State): Thông tin đặc biệt đối với một Tài khoản cụ thể được duy trì giữa các thời điểm mà EVM Code liên quan của Tài khoản chạy.

Tin nhắn (Message): Dữ liệu (dưới dạng một tập hợp byte) và Giá trị (được chỉ định dưới dạng Ether) được truyền giữa hai Tài khoản, hoặc thông qua hoạt động xác định của một Đối tượng Tự trị hoặc chữ ký mật mã bảo đảm của Giao dịch.

Cuộc gọi tin nhắn (Message Call): Hành động truyền một thông điệp từ một Tài khoản sang một Tài khoản khác. Nếu tài khoản đích liên quan đến EVM Code không rỗng, thì VM sẽ được bắt đầu với trạng thái của Đối tượng đó và Thông điệp được thực hiện. Nếu người gửi thông điệp là một Đối tượng Tự trị, thì Cuộc gọi sẽ truyền bất kỳ dữ liệu nào được trả về từ hoạt động VM.

Gas: Đơn vị chi phí mạng cơ bản. Chỉ thanh toán bằng Ether (tính đến PoC-4), có thể chuyển đổi tự do thành và từ Gas khi cần. Gas không tồn tại bên ngoài của động cơ tính toán nội tại của Ethereum; giá của nó được đặt bởi Giao dịch và các máy xác minh có quyền bỏ qua các Giao dịch có giá Gas quá thấp.

Hợp đồng (Contract): Thuật ngữ không chính thức được sử dụng để chỉ cả một phần mã EVM có thể được liên kết với một Tài khoản hoặc một Đối tượng Tự trị.

Đối tượng (Object): Đồng nghĩa với Đối tượng Tự trị.

Ứng dụng (App): Một ứng dụng có thể nhìn thấy được bởi người dùng cuối được lưu trữ trong Trình duyệt Ethereum.

Trình duyệt Ethereum (Ethereum Browser): (còn được gọi là Khách hàng Tham chiếu Ethereum) Một giao diện đồ họa người dùng chéo nền tảng giống như một trình duyệt đơn giản (giống như Chrome) có khả năng chứa các ứng dụng được bảo vệ trong môi trường đặc biệt của giao thức Ethereum.

Máy ảo Ethereum (Ethereum Virtual Machine): (còn được gọi là EVM) Máy ảo tạo thành phần chính của mô hình thực thi cho mã nguồn EVM được liên kết của tài khoản.

Môi trường Thực thi Ethereum (Ethereum Runtime Environment): (còn được gọi là ERE) Môi trường được cung cấp cho một Đối tượng Tự trị thực hiện trong EVM. Bao gồm EVM nhưng cũng bao gồm cấu trúc của trạng thái thế giới mà EVM phụ thuộc vào cho một số chỉ thị I/O cụ thể bao gồm CALL & CREATE.

Mã EVM (EVM Code): Mã bytecode mà EVM có thể thực thi trực tiếp. Được sử dụng để đặc tả chính thức ý nghĩa và hậu quả của một thông điệp đối với một Tài khoản.

Hợp ngữ EVM (EVM Assembly): (còn được gọi là EVM Assembly) Biểu diễn dạng đọc được của mã EVM.

LLL: Ngôn ngữ Thấp cấp giống Lisp, một ngôn ngữ có thể viết được sử dụng để soạn thảo hợp đồng đơn giản và bộ công cụ ngôn ngữ cấp thấp chung để biên dịch qua.

Appendix B. Tiền Tố Độ Dài Đệ Quy (Recursive Length Prefix)

Đây là một phương pháp tuần tự hóa để mã hóa dữ liệu nhị phân có cấu trúc tùy ý (mảng byte).

Chúng ta xác định tập hợp các cấu trúc có thể $\mathbb{T}$:

$$\mathbb{T} \equiv \mathbb{L} \uplus \mathbb{B} \tag{180}$$

$$\mathbb{L} \equiv \{\mathbf{t} : \mathbf{t} = (\mathbf{t}[0], \mathbf{t}[1], ...) \ \wedge \ \forall n < \|\mathbf{t}\| : \mathbf{t}[n] \in \mathbb{T}\} \tag{181}$$

$$\mathbb{B} \equiv \{\mathbf{b} : \mathbf{b} = (\mathbf{b}[0], \mathbf{b}[1], ...) \ \wedge \ \forall n < \|\mathbf{b}\| : \mathbf{b}[n] \in \mathbb{O}\} \tag{182}$$

Trong đó, $\mathbb{O}$ là tập hợp của các byte (8-bit). Do đó, $\mathbb{B}$ là tập hợp của tất cả các dãy byte (còn được biết đến là mảng byte, và một lá nếu được tưởng tượng như một cây), $\mathbb{L}$ là tập hợp của tất cả các cấu trúc (hoặc cấu trúc con) giống cây không phải là một lá duy nhất (một nút nhánh nếu được tưởng tượng như một cây) và $\mathbb{T}$ là tập hợp của tất cả các mảng byte và các dãy cấu trúc như vậy. Hợp nhất không chồng lấp $\uplus$ chỉ để phân biệt giữa mảng byte trống $() \in \mathbb{B}$ từ danh sách trống $() \in \mathbb{L}$, thứ sẽ được mã hóa một cách khác biệt như được định nghĩa dưới đây; như thường lệ, chúng ta sẽ lạm dụng ký hiệu (sử dụng ký hiệu một cách không chính thức) và để các chỉ số hợp nhất không chồng lấp là ngầm định, có thể suy ra được từ ngữ cảnh (kỳ vọng đọc giả phải suy luận ý nghĩa các chỉ số từ ngữ cảnh chung của văn bản).

Chúng ta định nghĩa hàm RLP là $\texttt{RLP}$ thông qua hai hàm con, hàm đầu tiên xử lý trường hợp khi giá trị là một mảng byte, hàm thứ hai khi nó là một chuỗi các giá trị khác nhau:

$$\texttt{RLP}(\mathbf{x}) \equiv \begin{cases} R_{\text{b}}(\mathbf{x}) & \text{nếu} \quad \mathbf{x} \in \mathbb{B} \\ R_{\text{l}}(\mathbf{x}) & \text{ngược lại} \end{cases} \tag{183}$$

Nếu giá trị cần được chuỗi hóa là một mảng byte, quá trình chuỗi hóa RLP có ba dạng:

- Nếu mảng byte chỉ chứa một byte duy nhất và byte đó nhỏ hơn 128, thì đầu vào hoàn toàn bằng đầu ra.
- Nếu mảng byte có ít hơn 56 byte, thì đầu ra bằng đầu vào được gắn thêm tiền tố bởi byte có giá trị là độ dài của mảng byte cộng thêm 128.
- Ngược lại, cho rằng nó có ít hơn 2^{64} byte, thì đầu ra bằng đầu vào được gắn tiền tố bởi giá trị chiều dài tối thiểu của mảng byte mà khi được biểu diễn như một big-endian là độ dài của mảng byte đầu vào, tiếp theo chính nó (chính tiền tố vừa nêu) lại được gắn thêm tiền tố bởi số byte cần thiết để mã hóa giá trị độ dài này một cách trung thực cộng thêm 183.

Mảng byte chứa 2^{64} byte hoặc nhiều hơn không thể được mã hóa. Ràng buộc này đảm bảo rằng byte đầu tiên của việc mã hóa một mảng byte luôn dưới 192, và do đó nó có thể dễ dàng phân biệt được khỏi việc mã hóa của các chuỗi trong $\mathbb{L}$.

Một cách chính thức, chúng ta định nghĩa R_{b}:

$$R_{\text{b}}(\mathbf{x}) \equiv \begin{cases} \mathbf{x} & \text{nếu} \quad \|\mathbf{x}\| = 1 \wedge \mathbf{x}[0] < 128 \\ (128 + \|\mathbf{x}\|) \cdot \mathbf{x} & \text{còn nếu} \quad \|\mathbf{x}\| < 56 \\ \left(183 + \left\|\texttt{BE}(\|\mathbf{x}\|)\right\|\right) \cdot \texttt{BE}(\|\mathbf{x}\|) \cdot \mathbf{x} & \text{còn nếu} \quad \|\mathbf{x}\| < 2^{64} \\ \varnothing & \text{ngược lại} \end{cases} \tag{184}$$

$$\texttt{BE}(x) \equiv (b_0, b_1, ...) : b_0 \neq 0 \wedge x = \sum_{n=0}^{\|\mathbf{b}\|-1} b_n \cdot 256^{\|\mathbf{b}\|-1-n} \tag{185}$$

$$(x_1, ..., x_n) \cdot (y_1, ..., y_m) = (x_1, ..., x_n, y_1, ..., y_m) \tag{186}$$

Vậy nên, $\texttt{BE}$ là hàm mở rộng một giá trị số nguyên không âm thành một mảng byte big-endian có chiều dài tối thiểu và toán tử dấu chấm '.' thực hiện việc nối chuỗi.

Nếu giá trị cần được chuỗi hóa là một chuỗi các mục khác nhau thì quá trình chuỗi hóa RLP có hai dạng:

- Nếu *chuỗi hóa đã nối (concatenated serialised)* của mỗi mục chứa có độ dài ít hơn 56 byte, thì đầu ra bằng *chuỗi hóa đã nối* đó gắn tiền tố bởi byte có giá trị bằng với độ dài của mảng byte *chuỗi hóa đã nối* này cộng thêm 192.
- Ngược lại, cho rằng chúng có ít hơn 2^{64} byte, đầu ra bằng *chuỗi hóa đã nối* đó gắn tiền tố bởi giá trị chiều dài tối thiểu của mảng byte mà khi được biểu diễn như một big-endian là độ dài của mảng byte *chuỗi hóa đã nối* đầu vào, tiếp theo chính nó (chính tiền tố vừa nêu) lại được gắn thêm tiền tố bởi số byte cần thiết để mã hóa giá trị độ dài này một cách trung thực cộng thêm 247.

Dãy mà *chuỗi hóa đã nối* các mục chứa 2^{64} byte hoặc nhiều hơn không thể được mã hóa. Ràng buộc này đảm bảo rằng byte đầu tiên của việc mã hóa không vượt quá 255 (ngược lại nó sẽ không phải là một byte).

Cuối cùng, chúng ta định nghĩa một cách chính thức R_{l}:

$$R_{\text{l}}(\mathbf{x}) \equiv \begin{cases} (192 + \|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{nếu} \quad s(\mathbf{x}) \neq \varnothing \wedge \|s(\mathbf{x})\| < 56 \\ \left(247 + \left\|\texttt{BE}(\|s(\mathbf{x})\|)\right\|\right) \cdot \texttt{BE}(\|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{còn nếu} \quad s(\mathbf{x}) \neq \varnothing \wedge \|s(\mathbf{x})\| < 2^{64} \\ \varnothing & \text{ngược lại} \end{cases} \tag{187}$$

$$s(\mathbf{x}) \equiv \begin{cases} \texttt{RLP}(\mathbf{x}[0]) \cdot \texttt{RLP}(\mathbf{x}[1]) \cdot ... & \text{nếu} \quad \forall i : \texttt{RLP}(\mathbf{x}[i]) \neq \varnothing \\ \varnothing & \text{ngược lại} \end{cases} \tag{188}$$

Nếu RLP được sử dụng để mã hóa một giá trị vô hướng (scalar), chỉ được định nghĩa là một số nguyên không âm (thuộc $\mathbb{N}$ hoặc thuộc $\mathbb{N}_x$ với mọi x), nó phải được mã hóa dưới dạng mảng byte ngắn nhất khi được biểu diễn là giá trị vô hướng. Do đó, RLP của một số nguyên không âm i được định nghĩa như sau:

$$\texttt{RLP}(i : i \in \mathbb{N}) \equiv \texttt{RLP}(\texttt{BE}(i)) \tag{189}$$

Khi biểu diễn dữ liệu RLP, nếu một đoạn dự kiến được giải mã là một giá trị vô hướng và có các chữ số 0 đứng đầu trong chuỗi byte, các client được yêu cầu coi nó không kinh điển (non-canonical) và xử lý nó giống như dữ liệu RLP không hợp lệ khác, loại bỏ nó hoàn toàn.

Không có định dạng mã hóa cụ thể cho giá trị có dấu hoặc giá trị dấu chấm động (số thực).

Appendix C. Mã Hóa Tiền Tố HEX (Hex-Prefix Encoding)

Mã hóa tiền tố hex là một phương pháp hiệu quả để mã hóa một số lượng nipples bất kỳ thành một mảng byte. Nó có thể lưu trữ một cờ hiệu bổ sung, khi được sử dụng trong ngữ cảnh của trie (duy nhất trong ngữ cảnh này nó được sử dụng), làm rõ ràng giữa các loại nút.

Nó được định nghĩa như là hàm $\texttt{HP}$ ánh xạ từ một dãy các nibble (nữa byte/4 bits) (được biểu diễn bởi tập hợp $\mathbb{Y}$) cùng với một giá trị boolean sang một dãy byte (được biểu diễn bởi tập hợp $\mathbb{B}$):

$$\texttt{HP}(\mathbf{x}, t) : \mathbf{x} \in \mathbb{Y} \quad \equiv \quad \begin{cases} (16f(t), 16\mathbf{x}[0] + \mathbf{x}[1], 16\mathbf{x}[2] + \mathbf{x}[3], ...) & \text{nếu} \quad \|\mathbf{x}\| \text{ là chẵn} \\ (16(f(t) + 1) + \mathbf{x}[0], 16\mathbf{x}[1] + \mathbf{x}[2], 16\mathbf{x}[3] + \mathbf{x}[4], ...) & \text{ngược lại} \end{cases} \tag{190}$$

$$f(t) \quad \equiv \quad \begin{cases} 2 & \text{nếu} \quad t \neq 0 \\ 0 & \text{ngược lại} \end{cases} \tag{191}$$

Do đó, nibble cao của byte đầu tiên chứa hai cờ hiệu; bit thấp nhất mã hóa tính chẵn lẻ của độ dài và bit thấp thứ hai mã hóa cờ t. Nibble thấp của byte đầu tiên là không trong trường hợp số lượng nibbles là chẵn và là nibble đầu tiên trong trường hợp số lượng nibbles là lẻ. Tất cả các nibble còn lại (bây giờ là số lẻ) vừa vặn đúng vào các byte còn lại.

Appendix D. Cây Merkle Patricia sửa đổi

Cây Merkle Patricia sửa đổi (trie) cung cấp một cấu trúc dữ liệu bền để ánh xạ giữa dữ liệu nhị phân có độ dài tùy ý (mảng byte). Nó được định nghĩa dưới dạng một cấu trúc dữ liệu có thể thay đổi để ánh xạ giữa các đoạn nhị phân 256 bit và dữ liệu nhị phân có độ dài tùy ý, thường được thực hiện dưới dạng một cơ sở dữ liệu. Lõi của trie, và yêu cầu duy nhất của nó trong kịch bản đặc tả giao thức, là cung cấp một giá trị duy nhất xác định một bộ cặp khóa-giá trị đã cho, có thể là một chuỗi 32 byte hoặc chuỗi byte trống. Để hiện thực giao thức một cách hiệu quả và hiệu quả, cách lưu trữ và duy trì cấu trúc của cây được để lại là một yếu tố cần xem xét khi triển khai.

Hình thức, chúng ta giả sử giá trị đầu vào $\mathfrak{I}$, một tập chứa các cặp dãy byte có khóa duy nhất:

$$\mathfrak{I} = \{(\mathbf{k}_0 \in \mathbb{B}, \mathbf{v}_0 \in \mathbb{B}), (\mathbf{k}_1 \in \mathbb{B}, \mathbf{v}_1 \in \mathbb{B}), ...\} \tag{192}$$

Khi xem xét một chuỗi như vậy, chúng ta sử dụng ký hiệu số học chung để tham chiếu đến khóa hoặc giá trị của một bộ, như sau:

$$\forall I \in \mathfrak{I} : I \equiv (I_0, I_1) \tag{193}$$

Bất kỳ loạt byte nào cũng có thể được xem là một loạt các nữa byte/4 bít (nibble), được đưa ra một ký hiệu đặc trưng cho endian; Ở đây chúng ta giả sử Big-Endian. Do đó:

$$y(\mathfrak{I}) \quad = \quad \{(\mathbf{k}'_0 \in \mathbb{Y}, \mathbf{v}_0 \in \mathbb{B}), (\mathbf{k}'_1 \in \mathbb{Y}, \mathbf{v}_1 \in \mathbb{B}), ...\} \tag{194}$$

$$\forall n : \quad \forall i < 2\|\mathbf{k}_n\| : \quad \mathbf{k}'_n[i] \quad \equiv \quad \begin{cases} \lfloor \mathbf{k}_n[i \div 2] \div 16 \rfloor & \text{nếu } i \text{ là chẵn} \\ \mathbf{k}_n[\lfloor i \div 2 \rfloor] \bmod 16 & \text{ngược lại} \end{cases} \tag{195}$$

Chúng ta định nghĩa hàm $\texttt{TRIE}$, mà khi đánh giá sẽ trả về gốc của cây trie đại diện tập hợp này khi được mã hóa trong cấu trúc này:

$$\texttt{TRIE}(\mathfrak{I}) \equiv \texttt{KEC}\big(\texttt{RLP}(c(\mathfrak{I}, 0))\big) \tag{196}$$

Chúng ta cũng giả định có một hàm n, là hàm giới hạn số nút của cây trie. Khi tạo một nút, chúng ta sử dụng RLP để mã hóa cấu trúc. Nhằm giảm độ phức tạp về lưu trữ, chúng ta lưu trữ các nút mà RLP tạo ra có độ dài ít hơn 32 byte trực tiếp; đối với những nút có kích thước lớn hơn, chúng ta khẳng định sự biết trước về dãy byte mà băm Keccak-256 của nó đánh giá đến tham chiếu của chúng ta. Do đó, chúng ta định nghĩa dựa trên c, hàm tạo nút:

$$n(\mathfrak{I}, i) \equiv \begin{cases} () \in \mathbb{B} & \text{nếu} \quad \mathfrak{I} = \varnothing \\ c(\mathfrak{I}, i) & \text{nếu} \quad \|\texttt{RLP}\big(c(\mathfrak{I}, i)\big)\| < 32 \\ \texttt{KEC}\big(\texttt{RLP}(c(\mathfrak{I}, i))\big) & \text{ngược lại} \end{cases} \tag{197}$$

Một cách tương tự như cây radix, khi trie được duyệt từ gốc đến lá, ta có thể xây dựng một cặp key-value duy nhất. Key được tích luỹ qua quá trình duyệt, thu thập một nibble duy nhất từ mỗi nút nhánh (giống như cây radix). Khác với cây radix, trong trường hợp nhiều key chia sẻ cùng một tiền tố hoặc trong trường hợp chỉ có một key có một hậu tố duy nhất, hai nút tối ưu hóa được cung cấp. Do đó, trong quá trình duyệt, có thể có thể lấy được nhiều nibble từ mỗi loại nút khác nhau, bao gồm nút mở rộng và lá. Có ba loại nút trong trie:

Leaf (Lá):: Một cấu trúc hai mục, mục đầu tiên tương ứng với các nibble trong key chưa được tính đến bởi việc tích luỹ của keys và các nhánh đã được duyệt từ gốc. Phương pháp mã hóa hex-prefix được sử dụng và tham số thứ hai cho hàm phải là 1.

Extension (Mở rộng):: Một cấu trúc hai mục, mục đầu tiên tương ứng với một chuỗi nibble có kích thước lớn hơn một, được chia sẻ bởi ít nhất hai key khác nhau sau khi tích lũy các nibble và nhánh khi duyệt từ gốc. Phương pháp mã hóa hex-prefix được sử dụng và tham số thứ hai cho hàm phải là 0.

Branch (Nhánh):: Một cấu trúc 17 mục, trong đó 16 mục tương ứng với mỗi giá trị nibble có thể có cho các key tại thời điểm này trong quá trình duyệt nó. Mục thứ 17 được sử dụng trong trường hợp này là một nút kết thúc và do đó một key kết thúc tại điểm này trong quá trình duyệt nó.

Một nhánh chỉ được sử dụng khi cần thiết; không có nút nhánh nào có thể tồn tại chỉ chứa một mục khác không. Chúng ta có thể định nghĩa cấu trúc này một cách chính thức bằng hàm sắp xếp cấu trúc c.

(198)
$$c(\mathfrak{J},i) \equiv \begin{cases} \big(\texttt{HP}(I_0[i..(\|I_0\|-1)],1),I_1\big) & \text{nếu} \quad \|\mathfrak{J}\|=1 \quad \text{trong đó } \exists I: I\in\mathfrak{J} \\ \big(\texttt{HP}(I_0[i..(j-1)],0),n(\mathfrak{J},j)\big) & \text{nếu} \quad i\neq j \quad \text{trong đó } j=\max\{x:\exists \mathbf{l}:\|\mathbf{l}\|=x\wedge\forall I\in\mathfrak{J}:I_0[0..(x-1)]=\mathbf{l}\} \\ (u(0),u(1),...,u(15),v) & \text{ngược lại} \quad \text{trong đó} \quad u(j) \equiv n(\{I:I\in\mathfrak{J}\wedge I_0[i]=j\},i+1) \\ & \qquad\qquad\qquad\qquad v = \begin{cases} I_1 & \text{nếu} \quad \exists I:I\in\mathfrak{J}\wedge\|I_0\|=i \\ () \in \mathbb{B} & \text{ngược lại} \end{cases} \end{cases}$$

D.1. **Cơ sở dữ liệu Trie (Trie Database).** Do không có giả định cụ thể nào được đưa ra về dữ liệu được lưu trữ và dữ liệu không được lưu trữ, vì đó là một xem xét cụ thể của việc triển khai; chúng ta đơn giản chỉ định hàm đồng nhất ánh xạ tập hợp key-value $\mathfrak{J}$ thành một hash 32-byte và khẳng định rằng chỉ có một hash như vậy tồn tại cho bất kỳ $\mathfrak{J}$ nào, điều này mặc dù không hoàn toàn chính xác nhưng đúng với độ chính xác chấp nhận được dựa trên khả năng chống va chạm của hàm hash Keccak. Trong thực tế, một triển khai hợp lý sẽ không tính toán lại toàn bộ hash gốc của trie cho mỗi bộ.

Một triển khai hợp lý sẽ duy trì một cơ sở dữ liệu của các nút xác định từ việc tính toán của các trie khác nhau hoặc, một cách chính thức hơn, nó sẽ ghi nhớ hàm c. Chiến lược này sử dụng tính chất của trie để dễ dàng gọi lại nội dung của bất kỳ tập hợp key-value trước đó nào và lưu trữ nhiều bộ như vậy một cách hiệu quả. Do mối quan hệ phụ thuộc, chứng minh Merkle có thể được xây dựng với yêu cầu không gian $O(\log N)$ mà có thể chứng minh một lá cụ thể phải tồn tại trong một trie có một root hash cụ thể.

Appendix E. Hợp Đồng Được Biên Dịch Trước (Precompiled Contracts)

Đối với mỗi hợp đồng được biên dịch trước, chúng ta sử dụng một hàm mẫu, $\Xi_{\texttt{PRE}}$, thực hiện việc kiểm tra hết gas (out-of-gas).

(199)
$$\Xi_{\texttt{PRE}}(\boldsymbol{\sigma},g,A,I) \equiv \begin{cases} (\varnothing,0,A,()) & \text{nếu} \quad g<g_{\text{r}} \\ (\boldsymbol{\sigma},g-g_{\text{r}},A,\mathbf{o}) & \text{ngược lại} \end{cases}$$

Các hợp đồng được biên dịch trước mỗi lần sử dụng các định nghĩa này và cung cấp các đặc tả cho $\mathbf{o}$ (dữ liệu đầu ra) và g_{r}, gas yêu cầu.

Chúng ta định nghĩa $\Xi_{\texttt{ECREC}}$ là một hợp đồng được biên dịch trước cho chức năng khôi phục khóa công khai của thuật toán chữ ký số đường cong elliptic (ECDSA) (ecrecover). Xem Phụ lục F để biết định nghĩa của hàm `ECDSARECOVER` và hằng số `secp256k1n`. Chúng ta cũng định nghĩa $\mathbf{d}$ là dữ liệu đầu vào, được định nghĩa rõ ràng cho một độ dài vô hạn bằng cách thêm các số không cần thiết. Trong trường hợp chữ ký không hợp lệ, chúng ta trả về không có đầu ra.

(200) $\Xi_{\texttt{ECREC}} \equiv \Xi_{\texttt{PRE}}$ trong đó:

(201) $g_{\text{r}} = 3000$

(202)
$$\|\mathbf{o}\| = \begin{cases} 0 & \text{nếu} \quad v\notin\{27,28\}\vee r=0\vee r\geq\texttt{secp256k1n}\vee s=0\vee s\geq\texttt{secp256k1n} \\ 0 & \text{nếu} \quad \texttt{ECDSARECOVER}(h,v-27,r,s)=\varnothing \\ 32 & \text{ngược lại} \end{cases}$$

(203) nếu $\|\mathbf{o}\|=32$:

(204) $\mathbf{o}[0..11] = 0$

(205) $\mathbf{o}[12..31] = \texttt{KEC}\big(\texttt{ECDSARECOVER}(h,v-27,r,s)\big)[12..31]$ trong đó:

(206) $\mathbf{d}[0..(\|I_{\mathbf{d}}\|-1)] = I_{\mathbf{d}}$

(207) $\mathbf{d}[\|I_{\mathbf{d}}\|..] = (0,0,...)$

(208) $h = \mathbf{d}[0..31]$

(209) $v = \mathbf{d}[32..63]$

(210) $r = \mathbf{d}[64..95]$

(211) $s = \mathbf{d}[96..127]$

Chúng ta định nghĩa $\Xi_{\texttt{SHA256}}$ và $\Xi_{\texttt{RIP160}}$ là các hợp đồng được biên dịch trước triển khai các hàm băm SHA2-256 và RIPEMD-160 tương ứng. Việc sử dụng gas của chúng phụ thuộc vào kích thước dữ liệu đầu vào, một yếu tố được làm tròn lên đến số từ (words) gần nhất.

$$\Xi_{\texttt{SHA256}} \equiv \Xi_{\texttt{PRE}} \quad \text{trong đó:} \tag{212}$$

$$g_{\text{r}} = 60 + 12\left\lceil \frac{\|I_{\mathbf{d}}\|}{32} \right\rceil \tag{213}$$

$$\mathbf{o}[0..31] = \texttt{SHA256}(I_{\mathbf{d}}) \tag{214}$$

$$\Xi_{\texttt{RIP160}} \equiv \Xi_{\texttt{PRE}} \quad \text{trong đó:} \tag{215}$$

$$g_{\text{r}} = 600 + 120\left\lceil \frac{\|I_{\mathbf{d}}\|}{32} \right\rceil \tag{216}$$

$$\mathbf{o}[0..11] = 0 \tag{217}$$

$$\mathbf{o}[12..31] = \texttt{RIPEMD160}(I_{\mathbf{d}}) \tag{218}$$

Cho mục đích ở đây, chúng ta giả định rằng chúng ta có các hàm mật mã tiêu chuẩn được định nghĩa rõ ràng cho RIPEMD-160 và SHA2-256 dưới dạng:

$$\texttt{SHA256}(\mathbf{i} \in \mathbb{B}) \equiv o \in \mathbb{B}_{32} \tag{219}$$

$$\texttt{RIPEMD160}(\mathbf{i} \in \mathbb{B}) \equiv o \in \mathbb{B}_{20} \tag{220}$$

Hợp đồng thứ tư, hàm đồng nhất $\Xi_{\texttt{ID}}$, đơn giản là định nghĩa đầu ra là đầu vào:

$$\Xi_{\texttt{ID}} \equiv \Xi_{\texttt{PRE}} \quad \text{trong đó:} \tag{221}$$

$$g_{\text{r}} = 15 + 3\left\lceil \frac{\|I_{\mathbf{d}}\|}{32} \right\rceil \tag{222}$$

$$\mathbf{o} = I_{\mathbf{d}} \tag{223}$$

Hợp đồng thứ năm thực hiện phép lũy thừa với độ chính xác tùy ý theo modulo. Ở đây, 0^0 được coi là một, và $x \bmod 0$ bằng không đối với tất cả các x. Từ (word) đầu tiên trong dữ liệu đầu vào chỉ định số byte mà số nguyên không âm đầu tiên B chiếm. Từ (word) thứ hai trong dữ liệu đầu vào chỉ định số byte mà số nguyên không âm thứ hai E chiếm. Từ thứ ba trong dữ liệu đầu vào chỉ định số byte mà số nguyên không âm thứ ba M chiếm. Ba từ (word) này được theo sau bởi B, E và M. Phần còn lại của dữ liệu đầu vào bị loại bỏ. Khi dữ liệu đầu vào quá ngắn, các byte bị thiếu được coi như là không. Đầu ra được mã hóa theo big-endian giống như định dạng của M.

$$\Xi_{\texttt{EXPMOD}} \equiv \Xi_{\texttt{PRE}} \quad \text{ngoại trừ:} \tag{224}$$

$$g_{\text{r}} = \max\left(200, \left\lfloor \frac{f\left(\max(\ell_{\text{M}}, \ell_{\text{B}})\right) \max(\ell'_{\text{E}}, 1)}{G_{\text{quaddivisor}}} \right\rfloor\right) \tag{225}$$

$$G_{\text{quaddivisor}} \equiv 3 \tag{226}$$

$$f(x) \equiv \left\lceil \frac{x}{8} \right\rceil^2 \tag{227}$$

$$\ell'_{\text{E}} = \begin{cases} 0 & \text{nếu } \ell_{\text{E}} \leq 32 \wedge E = 0 \\ \lfloor \log_2(E) \rfloor & \text{nếu } \ell_{\text{E}} \leq 32 \wedge E \neq 0 \\ 8(\ell_{\text{E}} - 32) + \lfloor \log_2(i[(96 + \ell_{\text{B}})..(127 + \ell_{\text{B}})]) \rfloor & \text{nếu } 32 < \ell_{\text{E}} \wedge i[(96 + \ell_{\text{B}})..(127 + \ell_{\text{B}})] \neq 0 \\ 8(\ell_{\text{E}} - 32) & \text{ngược lại} \end{cases} \tag{228}$$

$$\mathbf{o} = \left(B^E \bmod M\right) \in \mathbb{N}_{8\ell_{\text{M}}} \tag{229}$$

$$\ell_{\text{B}} \equiv i[0..31] \tag{230}$$

$$\ell_{\text{E}} \equiv i[32..63] \tag{231}$$

$$\ell_{\text{M}} \equiv i[64..95] \tag{232}$$

$$B \equiv i[96..(95 + \ell_{\text{B}})] \tag{233}$$

$$E \equiv i[(96 + \ell_{\text{B}})..(95 + \ell_{\text{B}} + \ell_{\text{E}})] \tag{234}$$

$$M \equiv i[(96 + \ell_{\text{B}} + \ell_{\text{E}})..(95 + \ell_{\text{B}} + \ell_{\text{E}} + \ell_{\text{M}})] \tag{235}$$

$$i[x] \equiv \begin{cases} I_{\mathbf{d}}[x] & \text{nếu } x < \|I_{\mathbf{d}}\| \\ 0 & \text{ngược lại} \end{cases} \tag{236}$$

E.1. **Các Hợp Đồng Biên Dịch Trước Liên Quan Đến zkSNARK.** Chúng ta chọn hai số, cả hai đều là số nguyên tố.

$$p \equiv 21888242871839275222246405745257275088696311157297823662689037894645226208583 \tag{237}$$

$$q \equiv 21888242871839275222246405745257275088548364400416034343698204186575808495617 \tag{238}$$

Vì p là một số nguyên tố, tập hợp $\{0, 1, \ldots, p-1\}$ tạo thành một trường với phép cộng và nhân theo modulo p. Chúng ta gọi trường này là F_{p}.

Chúng ta định nghĩa một tập hợp C_1 với

$$C_1 \equiv \{(X, Y) \in F_\mathrm{p} \times F_\mathrm{p} \mid Y^2 = X^3 + 3\} \cup \{(0, 0)\} \tag{239}$$

Chúng ta định nghĩa một phép toán nhị phân $+$ trên C_1 cho các phần tử phân biệt $(X_1, Y_1), (X_2, Y_2)$ với:

$$\begin{aligned}(X_1, Y_1) + (X_2, Y_2) &\equiv \begin{cases}(X, Y) & \text{nếu } X_1 \neq X_2\\ (0, 0) & \text{ngược lại}\end{cases}\\ \lambda &\equiv \frac{Y_2 - Y_1}{X_2 - X_1}\\ X &\equiv \lambda^2 - X_1 - X_2\\ Y &\equiv \lambda(X_1 - X) - Y_1\end{aligned} \tag{240}$$

Trong trường hợp $(X_1, Y_1) = (X_2, Y_2)$, chúng ta định nghĩa $+$ trên C_1 với:

$$\begin{aligned}(X_1, Y_1) + (X_2, Y_2) &\equiv \begin{cases}(X, Y) & \text{nếu } Y_1 \neq 0\\ (0, 0) & \text{ngược lại}\end{cases}\\ \lambda &\equiv \frac{3X_1^2}{2Y_1}\\ X &\equiv \lambda^2 - 2X_1\\ Y &\equiv \lambda(X_1 - X) - Y_1\end{aligned} \tag{241}$$

$(C_1, +)$ được biết đến là một nhóm. Chúng ta định nghĩa phép nhân vô hướng $\cdot$ với:

$$n \cdot P \equiv (0, 0) + \underbrace{P + \cdots + P}_{n} \tag{242}$$

cho một số tự nhiên n và một điểm P trong C_1.

Chúng ta định nghĩa P_1 là một điểm $(1, 2)$ trên C_1. Đặt G_1 là nhóm con của $(C_1, +)$ được tạo ra bởi P_1. G_1 được biết đến là một nhóm cyclic có bậc q. Với một điểm P trong G_1, chúng ta định nghĩa $\log_{P_1}(P)$ là số tự nhiên nhỏ nhất n sao cho $n \cdot P_1 = P$. $\log_{P_1}(P)$ có giá trị tối đa là $q - 1$.

Đặt F_{p^2} là một trường $F_p[i]/(i^2 + 1)$. Chúng ta định nghĩa một tập hợp C_2 với

$$C_2 \equiv \{(X, Y) \in F_{p^2} \times F_{p^2} \mid Y^2 = X^3 + 3(i + 9)^{-1}\} \cup \{(0, 0)\} \tag{243}$$

Chúng ta định nghĩa một phép toán nhị phân $+$ và phép nhân vô hướng $\cdot$ với các phương trình giống như (240), (241) và (242). $(C_2, +)$ cũng được biết đến là một nhóm. Chúng ta định nghĩa P_2 trong C_2 với:

$$\begin{aligned}P_2 \equiv\ & (11559732032986387107991004021392285783925812861821192530917403151452391805634 \times i\\ & +10857046999023057135944570762232829481370756359578518086990519993285655852781,\\ & 4082367875863433681332203403145435568316851327593401208105741076214120093531 \times i\\ & +84956539231234314176049732474892724384181905872636001487702806493069581019 30)\end{aligned} \tag{244}$$

Chúng ta định nghĩa G_2 là nhóm con của $(C_2, +)$ được tạo ra bởi P_2. G_2 được biết đến là duy nhất nhóm cyclic có bậc q trên C_2. Với một điểm P trong G_2, chúng ta định nghĩa $\log_{P_2}(P)$ là số tự nhiên nhỏ nhất n sao cho $n \cdot P_2 = P$. Với định nghĩa này, $\log_{P_2}(P)$ có giá trị tối đa là $q - 1$.

Đặt G_T là nhóm Abel nhân (multiplicative abelian group) dưới cơ sở của $F_{q^{12}}$. Đã biết rằng một ánh xạ tuyến tính không suy giảm $e : G_1 \times G_2 \to G_T$ tồn tại. Ánh xạ tuyến tính này là một ánh xạ đôi loại ba. Có nhiều ánh xạ đôi như vậy, không quan trọng cái nào được chọn làm e.

Đặt $P_T = e(P_1, P_2)$, a là một tập hợp gồm k điểm trong G_1, và b là một tập hợp gồm k điểm trong G_2. Theo định nghĩa của ánh xạ đôi, các điều sau đây là tương đương

$$\log_{P_1}(a_1) \times \log_{P_2}(b_1) + \cdots + \log_{P_1}(a_k) \times \log_{P_2}(b_k) \equiv 1 \mod q \tag{245}$$

$$\prod_{i=0}^{k} e\left(a_i, b_i\right) = P_T \tag{246}$$

Do đó, phép toán đôi cung cấp một phương pháp để xác minh (245).

Một số 32 bytes $\mathbf{x} \in \mathbf{P}_{256}$ có thể hoặc không đại diện cho một phần tử trong F_p.

$$\delta_\mathrm{p}(\mathbf{x}) \equiv \begin{cases}\mathbf{x} & \text{nếu } \mathbf{x} < p\\ \varnothing & \text{ngược lại}\end{cases} \tag{247}$$

Một dữ liệu 64 bytes $\mathbf{x} \in \mathbf{B}_{512}$ có thể hoặc không đại diện cho một phần tử trong G_1.

$$\delta_1(\mathbf{x}) \equiv \begin{cases} g_1 & \text{nếu } g_1 \in G_1 \\ \varnothing & \text{ngược lại} \end{cases} \tag{248}$$

$$g_1 \equiv \begin{cases} (x, y) & \text{nếu } x \neq \varnothing \wedge y \neq \varnothing \\ \varnothing & \text{ngược lại} \end{cases} \tag{249}$$

$$x \equiv \delta_{\mathrm{p}}(\mathbf{x}[0..31]) \tag{250}$$

$$y \equiv \delta_{\mathrm{p}}(\mathbf{x}[32..63]) \tag{251}$$

Một dữ liệu 128 bytes $\mathbf{x} \in \mathbf{B}_{1024}$ có thể hoặc không đại diện cho một phần tử trong G_2.

$$\delta_2(\mathbf{x}) \equiv \begin{cases} g_2 & \text{nếu } g_2 \in G_2 \\ \varnothing & \text{ngược lại} \end{cases} \tag{252}$$

$$g_2 \equiv \begin{cases} ((x_0 i + y_0), (x_1 i + y_1)) & \text{nếu } x_0 \neq \varnothing \wedge y_0 \neq \varnothing \wedge x_1 \neq \varnothing \wedge y_1 \neq \varnothing \\ \varnothing & \text{ngược lại} \end{cases} \tag{253}$$

$$x_0 \equiv \delta_{\mathrm{p}}(\mathbf{x}[0..31]) \tag{254}$$

$$y_0 \equiv \delta_{\mathrm{p}}(\mathbf{x}[32..63]) \tag{255}$$

$$x_1 \equiv \delta_{\mathrm{p}}(\mathbf{x}[64..95]) \tag{256}$$

$$y_1 \equiv \delta_{\mathrm{p}}(\mathbf{x}[96..127]) \tag{257}$$

Chúng ta định nghĩa $\Xi_{\texttt{SNARKV}}$ là một hợp đồng được biên dịch trước kiểm tra xem (245) có thỏa mãn hay không, dùng cho việc xác minh zkSNARK.

$$\Xi_{\texttt{SNARKV}} \equiv \Xi_{\texttt{PRE}} \quad \text{ngoại trừ:} \tag{258}$$

$$\Xi_{\texttt{SNARKV}}(\boldsymbol{\sigma}, g, A, I) = (\varnothing, 0, A, ()) \quad \text{nếu } F \tag{259}$$

$$F \equiv (\|I_{\mathbf{d}}\| \bmod 192 \neq 0 \vee (\exists j.\ a_{\mathrm{j}} = \varnothing \vee b_{\mathrm{j}} = \varnothing)) \tag{260}$$

$$k = \frac{\|I_{\mathbf{d}}\|}{192} \tag{261}$$

$$g_{\mathrm{r}} = 34000k + 45000 \tag{262}$$

$$\mathbf{o}[0..31] \equiv \begin{cases} 0x0000000000000000000000000000000000000000000000000000000000000001 & \text{nếu } v \wedge \neg F \\ 0x0000000000000000000000000000000000000000000000000000000000000000 & \text{nếu } \neg v \wedge \neg F \end{cases} \tag{263}$$

$$v \equiv (\log_{P_1}(a_1) \times \log_{P_2}(b_1) + \cdots + \log_{P_1}(a_k) \times \log_{P_2}(b_k) \equiv 1 \mod q) \tag{264}$$

$$a_1 \equiv \delta_1(I_{\mathbf{d}}[0..63]) \tag{265}$$

$$b_1 \equiv \delta_2(I_{\mathbf{d}}[64..191]) \tag{266}$$

$$\vdots$$

$$a_k \equiv \delta_1(I_{\mathbf{d}}[(\|I_{\mathbf{d}}\| - 192)..(\|I_{\mathbf{d}}\| - 129)]) \tag{267}$$

$$b_k \equiv \delta_2(I_{\mathbf{d}}[(\|I_{\mathbf{d}}\| - 128)..(\|I_{\mathbf{d}}\| - 1)]) \tag{268}$$

Chúng ta định nghĩa một hợp đồng được biên dịch trước cho phép cộng trên G_1.

$$\Xi_{\texttt{BN\_ADD}} \equiv \Xi_{\texttt{BN\_PRE}} \quad \text{ngoại trừ:} \tag{269}$$

$$\Xi_{\texttt{BN\_ADD}}(\boldsymbol{\sigma}, g, A, I) = (\varnothing, 0, A, ()) \quad \text{nếu } x = \varnothing \vee y = \varnothing \tag{270}$$

$$g_{\mathrm{r}} = 150 \tag{271}$$

$$\mathbf{o} \equiv \delta_1^{-1}(x + y) \quad \text{trong đó } + \text{ là hoạt động nhóm trong } G_1 \tag{272}$$

$$x \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[0..63]\right) \tag{273}$$

$$y \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[64..127]\right) \tag{274}$$

$$\bar{I}_{\mathbf{d}}[x] \equiv \begin{cases} I_{\mathbf{d}}[x] & \text{nếu } x < \|I_{\mathbf{d}}\| \\ 0 & \text{ngược lại} \end{cases} \tag{275}$$

Chúng ta định nghĩa một hợp đồng được biên dịch trước cho phép nhân vô hướng trên G_1, trong đó $\bar{I}_{\mathbf{d}}$ được định nghĩa trong (275).

$$\Xi_{\texttt{BN\_MUL}} \equiv \Xi_{\texttt{PRE}} \quad \text{ngoại trừ:} \tag{276}$$
$$\Xi_{\texttt{BN\_MUL}}(\boldsymbol{\sigma}, g, A, I) = (\varnothing, 0, A, ()) \quad \text{nếu } x = \varnothing \tag{277}$$
$$g_{\mathrm{r}} = 6000 \tag{278}$$
$$\mathbf{o} \equiv \delta_1^{-1}(n \cdot x) \quad \text{trong đó } \cdot \text{ là phép nhân vô hướng trong } G_1 \tag{279}$$
$$x \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[0..63]\right) \tag{280}$$
$$n \equiv \bar{I}_{\mathbf{d}}[64..95] \tag{281}$$

E.2. **Hợp Đồng Được Biên Dịch Trước BLAKE2.** EIP-152 do Hess et al. [2016] định nghĩa $\Xi_{\texttt{BLAKE2\_F}}$ là một hợp đồng được biên dịch trước thực hiện hàm nén `F` được sử dụng trong thuật toán băm mật mã BLAKE2. Hàm nén `F` được chỉ định trong RFC 7693 của Saarinen and Aumasson [2015].

$$\Xi_{\texttt{BLAKE2\_F}} \equiv \Xi_{\texttt{PRE}} \quad \text{ngoại trừ:} \tag{282}$$
$$\Xi_{\texttt{BLAKE2\_F}}(\boldsymbol{\sigma}, g, A, I) = (\varnothing, 0, A, ()) \quad \text{nếu } \|I_{\mathbf{d}}\| \neq 213 \vee f \notin \{0, 1\} \tag{283}$$
$$g_{\mathrm{r}} = r \tag{284}$$
$$\mathbf{o} \equiv \texttt{LE}_8(h'_0) \cdot ... \cdot \texttt{LE}_8(h'_7) \tag{285}$$
$$(h'_0, \ldots, h'_7) \equiv \texttt{F}(h, m, t_{\text{low}}, t_{\text{high}}, f) \quad \text{with } r \text{ rounds and } w = 64 \tag{286}$$
$$\texttt{BE}_4(r) \equiv I_{\mathbf{d}}[0..4] \tag{287}$$
$$\texttt{LE}_8(h_0) \equiv I_{\mathbf{d}}[4..12] \tag{288}$$
$$\ldots \tag{289}$$
$$\texttt{LE}_8(h_7) \equiv I_{\mathbf{d}}[60..68] \tag{290}$$
$$\texttt{LE}_8(m_0) \equiv I_{\mathbf{d}}[68..76] \tag{291}$$
$$\ldots \tag{292}$$
$$\texttt{LE}_8(m_{15}) \equiv I_{\mathbf{d}}[188..196] \tag{293}$$
$$\texttt{LE}_8(t_{\text{low}}) \equiv I_{\mathbf{d}}[196..204] \tag{294}$$
$$\texttt{LE}_8(t_{\text{high}}) \equiv I_{\mathbf{d}}[204..212] \tag{295}$$
$$f \equiv I_{\mathbf{d}}[212] \tag{296}$$

trong đó $r \in \mathbb{B}_{32}$, $\forall i \in 0..7 : h_i \in \mathbb{B}_{64}$, $\forall i \in 0..15 : m_i \in \mathbb{B}_{64}$, $t_{\text{low}} \in \mathbb{B}_{64}$, $t_{\text{high}} \in \mathbb{B}_{64}$, $f \in \mathbb{B}_8$, $\texttt{BE}_k$ là biểu diễn k-byte big-endian—so sánh với (185):

$$\texttt{BE}_k(x) \equiv (b_0, b_1, ..., b_{k-1}) : x = \sum_{n=0}^{k-1} b_n \cdot 256^{k-1-n} \tag{297}$$

và $\texttt{LE}_k$ là biểu diễn k-byte little-endian:

$$\texttt{LE}_k(x) \equiv (b_0, b_1, ..., b_{k-1}) : x = \sum_{n=0}^{k-1} b_n \cdot 256^{n} \tag{298}$$

Appendix F. Ký Giao Dịch (Signing Transactions)

Các giao dịch được ký bằng chữ ký ECDSA có thể khôi phục. Phương pháp này sử dụng đường cong SECP-256k1 như mô tả bởi Courtois et al. [2014], và được thực hiện giống như mô tả bởi Gura et al. [2004] trên trang 9 của 15, đoạn 3.

Giả sử người gửi có một khóa riêng (private key) hợp lệ p_{r}, đó là một số nguyên dương được chọn ngẫu nhiên (được biểu diễn dưới dạng mảng byte có độ dài 32 theo định dạng big-endian) trong khoảng $[1, \texttt{secp256k1n} - 1]$.

Chúng ta giả định sự tồn tại của các hàm `ECDSAPUBKEY`, `ECDSASIGN` và `ECDSARECOVER`. Các hàm này được định nghĩa chính thức trong văn bản chuyên ngành, vd. bởi Johnson et al. [2001].

$$\texttt{ECDSAPUBKEY}(p_{\mathrm{r}} \in \mathbb{B}_{32}) \equiv p_{\mathrm{u}} \in \mathbb{B}_{64} \tag{299}$$
$$\texttt{ECDSASIGN}(e \in \mathbb{B}_{32}, p_{\mathrm{r}} \in \mathbb{B}_{32}) \equiv (v \in \mathbb{B}_1, r \in \mathbb{B}_{32}, s \in \mathbb{B}_{32}) \tag{300}$$
$$\texttt{ECDSARECOVER}(e \in \mathbb{B}_{32}, v \in \mathbb{B}_1, r \in \mathbb{B}_{32}, s \in \mathbb{B}_{32}) \equiv p_{\mathrm{u}} \in \mathbb{B}_{64} \tag{301}$$

Ở đây, p_{u} là khóa công khai (public key), giả định là một mảng byte có kích thước 64 (được tạo thành từ sự nối liền của hai số nguyên dương mỗi số $< 2^{256}$), p_{r} là khóa riêng, một mảng byte có kích thước 32 (hoặc một số nguyên dương duy nhất trong khoảng đã nói) và e là hash của giao dịch, $h(T)$. Giả định rằng v là 'định dạng nhận diện'. Định dạng nhận diện là giá trị 1 byte chỉ định tính chẵn hay lẻ và sự hữu hạn của các tọa độ của điểm đường cong mà r là giá trị x; giá trị này nằm trong khoảng $[0, 3]$, tuy nhiên chúng ta tuyên bố rằng hai khả năng cao nhất, đại diện cho giá trị vô hạn, là không hợp lệ. Giá trị 0 đại diện cho một giá trị y chẵn và 1 đại diện cho một giá trị y lẻ.

Chúng ta tuyên bố rằng một chữ ký ECDSA không hợp lệ trừ khi tất cả các điều kiện sau đây đều đúng:

$$0 < r < \texttt{secp256k1n} \tag{302}$$

$$0 < s < \texttt{secp256k1n} \div 2 + 1 \tag{303}$$

$$v \in \{0, 1\} \tag{304}$$

trong đó:

$$\texttt{secp256k1n} = 115792089237316195423570985008687907852837564279074904382605163141518161494337 \tag{305}$$

Lưu ý rằng ràng buộc về s này nghiêm túc hơn so với ràng buộc 202 trong Ξ_{ECREC} biên dịch trước; xem EIP-2 của Buterin [2015] để biết thêm chi tiết.

Đối với một khóa riêng (private key) p_{r} cụ thể, địa chỉ Ethereum $A(p_{\text{r}})$ (một giá trị 160-bit) tương ứng với nó được định nghĩa là 160 bit bên phải của hash Keccak-256 của khóa công khai ECDSA tương ứng:

$$A(p_{\text{r}}) = \mathcal{B}_{96..255}\big(\texttt{KEC}\big(\texttt{ECDSAPUBKEY}(p_{\text{r}})\big)\big) \tag{306}$$

Hash của thông điệp, $h(T)$, cần được ký là hash Keccak-256 của giao dịch. Có bốn phiên bản khác nhau của các kịch bản ký khác nhau:

$$L_{\text{X}}(T) \equiv \begin{cases} (T_{\text{n}}, T_{\text{p}}, T_{\text{g}}, T_{\text{t}}, T_{\text{v}}, \mathbf{p}) & \text{nếu } T_{\text{x}} = 0 \wedge T_{\text{w}} \in \{27, 28\} \\ (T_{\text{n}}, T_{\text{p}}, T_{\text{g}}, T_{\text{t}}, T_{\text{v}}, \mathbf{p}, \beta, (), ()) & \text{nếu } T_{\text{x}} = 0 \wedge T_{\text{w}} \in \{2\beta + 35, 2\beta + 36\} \\ (T_{\text{c}}, T_{\text{n}}, T_{\text{p}}, T_{\text{g}}, T_{\text{t}}, T_{\text{v}}, \mathbf{p}, T_{\mathbf{A}}) & \text{nếu } T_{\text{x}} = 1 \\ (T_{\text{c}}, T_{\text{n}}, T_{\text{f}}, T_{\text{m}}, T_{\text{g}}, T_{\text{t}}, T_{\text{v}}, \mathbf{p}, T_{\mathbf{A}}) & \text{nếu } T_{\text{x}} = 2 \end{cases} \tag{307}$$

trong đó

$$\mathbf{p} \equiv \begin{cases} T_{\mathbf{i}} & \text{nếu } T_{\text{t}} = \varnothing \\ T_{\mathbf{d}} & \text{ngược lại} \end{cases}$$

$$h(T) \equiv \begin{cases} \texttt{KEC}(\texttt{RLP}(L_{\text{X}}(T))) & \text{nếu } T_{\text{x}} = 0 \\ \texttt{KEC}(T_{\text{x}} \cdot \texttt{RLP}(L_{\text{X}}(T))) & \text{ngược lại} \end{cases} \tag{308}$$

Giao dịch đã ký $G(T, p_{\text{r}})$ được định nghĩa là:

$$G(T, p_{\text{r}}) \equiv T \quad \text{ngoại trừ:} \tag{309}$$

$$(T_{\text{y}}, T_{\text{r}}, T_{\text{s}}) = \texttt{ECDSASIGN}(h(T), p_{\text{r}}) \tag{310}$$

Nhắc lại từ phần trước:

$$T_{\text{r}} = r \tag{311}$$

$$T_{\text{s}} = s \tag{312}$$

và T_{w} của các giao dịch kế thừa là $27 + T_{\text{y}}$ hoặc $2\beta + 35 + T_{\text{y}}$.

Sau đó chúng ta có thể xác định hàm người gửi S của giao dịch là:

$$S(T) \equiv \mathcal{B}_{96..255}\big(\texttt{KEC}\big(\texttt{ECDSARECOVER}(h(T), v, T_{\text{r}}, T_{\text{s}})\big)\big) \tag{313}$$

$$v \equiv \begin{cases} T_{\text{w}} - 27 & \text{nếu } T_{\text{x}} = 0 \wedge T_{\text{w}} \in \{27, 28\} \\ (T_{\text{w}} - 35) \bmod 2 & \text{nếu } T_{\text{x}} = 0 \wedge T_{\text{w}} \in \{2\beta + 35, 2\beta + 36\} \\ T_{\text{y}} & \text{nếu } T_{\text{x}} = 1 \vee T_{\text{x}} = 2 \end{cases} \tag{314}$$

Việc khẳng định rằng người gửi giao dịch đã ký bằng địa chỉ của người ký phải là hiển nhiên:

$$\forall T : \forall p_{\text{r}} : S(G(T, p_{\text{r}})) \equiv A(p_{\text{r}}) \tag{315}$$

Appendix G. Biểu Phí (Fee Schedule)

Biểu phí G là một bộ (tuple) giá trị vô hướng tương ứng với chi phí tương đối, tính bằng gas, của một số hoạt động trừu tượng mà một giao dịch có thể thực hiện.

Tên	Giá trị	Mô tả
G_{zero}	0	Không có chi phí cho các hoạt động của tập hợp W_{zero}.
G_{jumpdest}	1	Số gas cần thanh toán cho một hoạt động JUMPDEST.
G_{base}	2	Số gas cần thanh toán cho các hoạt động của tập hợp W_{base}.
G_{verylow}	3	Số gas cần thanh toán cho các hoạt động của tập hợp W_{verylow}.
G_{low}	5	Số gas cần thanh toán cho các hoạt động của tập hợp W_{low}.
G_{mid}	8	Số gas cần thanh toán cho các hoạt động của tập hợp W_{mid}.
G_{high}	10	Số gas cần thanh toán cho các hoạt động của tập hợp W_{high}.
$G_{\text{warmaccess}}$	100	Chi phí truy cập tài khoản hoặc bộ nhớ "nóng".
$G_{\text{accesslistaddress}}$	2400	Chi phí làm "nóng" một tài khoản với danh sách truy cập.
$G_{\text{accessliststorage}}$	1900	Chi phí làm "nóng" một bộ nhớ với danh sách truy cập.
$G_{\text{coldaccountaccess}}$	2600	Chi phí truy cập tài khoản "lạnh".
$G_{\text{coldsload}}$	2100	Chi phí truy cập bộ nhớ "lạnh".
G_{sset}	20000	Thanh toán cho hoạt động SSTORE khi giá trị lưu trữ được thiết lập khác không từ không.
G_{sreset}	2900	Thanh toán cho hoạt động SSTORE khi giá trị lưu trữ giữ nguyên giá trị không hoặc được thiết lập thành không.
R_{sclear}	4800	Hoàn trả (được thêm vào bộ đếm hoàn trả) khi giá trị lưu trữ được thiết lập thành không từ khác không. Số lượng hoàn trả được định nghĩa là $G_{\text{sreset}} + G_{\text{accessliststorage}}$.
$G_{\text{selfdestruct}}$	5000	Số gas cần thanh toán cho một hoạt động SELFDESTRUCT.
G_{create}	32000	Thanh toán cho một hoạt động CREATE.
$G_{\text{codedeposit}}$	200	Thanh toán cho mỗi byte cho một hoạt động CREATE để đặt mã vào trạng thái.
$G_{\text{callvalue}}$	9000	Thanh toán cho việc chuyển khoản (transfer) giá trị khác không trong hoạt động CALL.
$G_{\text{callstipend}}$	2300	Một khoản trợ cấp cho hợp đồng được gọi được trừ khỏi $G_{\text{callvalue}}$ để chuyển (transfer) giá trị khác không.
$G_{\text{newaccount}}$	25000	Thanh toán cho hoạt động CALL hoặc SELFDESTRUCT tạo một tài khoản mới.
G_{exp}	10	Thanh toán một phần cho hoạt động EXP.
G_{expbyte}	50	Thanh toán một phần khi nhân với số byte trong số mũ cho hoạt động EXP.
G_{memory}	3	Thanh toán cho mỗi từ bổ sung khi mở rộng bộ nhớ.
G_{txcreate}	32000	Thanh toán bởi tất cả các giao dịch tạo hợp đồng sau chuyển đổi *Homestead*.
$G_{\text{txdatazero}}$	4	Thanh toán cho mỗi byte zero của dữ liệu hoặc mã trong một giao dịch.
$G_{\text{txdatanonzero}}$	16	Thanh toán cho mỗi byte khác không của dữ liệu hoặc mã trong một giao dịch.
$G_{\text{transaction}}$	21000	Thanh toán cho mỗi giao dịch.
G_{log}	375	Thanh toán một phần cho hoạt động LOG.
G_{logdata}	8	Thanh toán cho mỗi byte trong dữ liệu hoạt động LOG.
G_{logtopic}	375	Thanh toán cho mỗi chủ đề (topic) của hoạt động LOG.
$G_{\text{keccak256}}$	30	Thanh toán cho mỗi hoạt động KECCAK256.
$G_{\text{keccak256word}}$	6	Thanh toán cho mỗi word (làm tròn lên) cho dữ liệu đầu vào của hoạt động KECCAK256.
G_{copy}	3	Thanh toán một phần cho hoạt động *COPY, nhân với số words được sao chép, làm tròn lên.
$G_{\text{blockhash}}$	20	Thanh toán cho mỗi hoạt động BLOCKHASH.

Appendix H. Đặc Tả Máy Ảo (Virtual Machine Specification)

Khi diễn giải các giá trị nhị phân 256 bit như số nguyên, biểu diễn là big-endian.

Khi một dữ liệu máy ảo 256 bit được chuyển đổi sang và từ một địa chỉ hoặc hash 160 bit, 20 byte bên phải (thứ tự thấp đầu tiên đối với big-endian) được sử dụng và 12 byte bên trái được loại bỏ hoặc điền vào bằng số 0, do đó các giá trị số nguyên (khi byte được diễn giải theo big-endian) là tương đương.

H.1. **Chi Phí Gas (Gas Cost).** Hàm chi phí gas chung, C, được định nghĩa như sau:

(316)
$$C(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \equiv C_{\text{mem}}(\boldsymbol{\mu}'_{\text{i}}) - C_{\text{mem}}(\boldsymbol{\mu}_{\text{i}}) + \begin{cases} C_{\text{SSTORE}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) & \text{nếu} \quad w = \text{SSTORE} \\ G_{\text{exp}} & \text{nếu} \quad w = \text{EXP} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ G_{\text{exp}} + G_{\text{expbyte}} \times (1 + \lfloor \log_{256}(\boldsymbol{\mu}_{\mathbf{s}}[1]) \rfloor) & \text{nếu} \quad w = \text{EXP} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] > 0 \\ G_{\text{verylow}} + G_{\text{copy}} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[2] \div 32 \rceil & \text{nếu} \quad w \in W_{\text{copy}} \\ C_{\text{aaccess}}(\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}, A) + G_{\text{copy}} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[3] \div 32 \rceil & \text{nếu} \quad w = \text{EXTCODECOPY} \\ C_{\text{aaccess}}(\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}, A) & \text{nếu} \quad w \in W_{\text{extaccount}} \\ G_{\text{log}} + G_{\text{logdata}} \times \boldsymbol{\mu}_{\mathbf{s}}[1] & \text{nếu} \quad w = \text{LOG0} \\ G_{\text{log}} + G_{\text{logdata}} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + G_{\text{logtopic}} & \text{nếu} \quad w = \text{LOG1} \\ G_{\text{log}} + G_{\text{logdata}} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 2G_{\text{logtopic}} & \text{nếu} \quad w = \text{LOG2} \\ G_{\text{log}} + G_{\text{logdata}} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 3G_{\text{logtopic}} & \text{nếu} \quad w = \text{LOG3} \\ G_{\text{log}} + G_{\text{logdata}} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 4G_{\text{logtopic}} & \text{nếu} \quad w = \text{LOG4} \\ C_{\text{CALL}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) & \text{nếu} \quad w \in W_{\text{call}} \\ C_{\text{SELFDESTRUCT}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) & \text{nếu} \quad w = \text{SELFDESTRUCT} \\ G_{\text{create}} & \text{nếu} \quad w = \text{CREATE} \\ G_{\text{create}} + G_{\text{keccak256word}} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[2] \div 32 \rceil & \text{nếu} \quad w = \text{CREATE2} \\ G_{\text{keccak256}} + G_{\text{keccak256word}} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[1] \div 32 \rceil & \text{nếu} \quad w = \text{KECCAK256} \\ G_{\text{jumpdest}} & \text{nếu} \quad w = \text{JUMPDEST} \\ C_{\text{SLOAD}}(\boldsymbol{\mu}, A, I) & \text{nếu} \quad w = \text{SLOAD} \\ G_{\text{zero}} & \text{nếu} \quad w \in W_{\text{zero}} \\ G_{\text{base}} & \text{nếu} \quad w \in W_{\text{base}} \\ G_{\text{verylow}} & \text{nếu} \quad w \in W_{\text{verylow}} \\ G_{\text{low}} & \text{nếu} \quad w \in W_{\text{low}} \\ G_{\text{mid}} & \text{nếu} \quad w \in W_{\text{mid}} \\ G_{\text{high}} & \text{nếu} \quad w \in W_{\text{high}} \\ G_{\text{blockhash}} & \text{nếu} \quad w = \text{BLOCKHASH} \end{cases}$$

(317)
$$w \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\text{pc}}] & \text{nếu} \quad \boldsymbol{\mu}_{\text{pc}} < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{ngược lại} \end{cases}$$

trong đó:

(318)
$$C_{\text{mem}}(a) \equiv G_{\text{memory}} \cdot a + \left\lfloor \frac{a^2}{512} \right\rfloor$$

(319)
$$C_{\text{aaccess}}(x, A) \equiv \begin{cases} G_{\text{warmaccess}} & \text{nếu} \quad x \in A_{\mathbf{a}} \\ G_{\text{coldaccountaccess}} & \text{ngược lại} \end{cases}$$

với C_{CALL}, $C_{\text{SELFDESTRUCT}}$, C_{SLOAD} và C_{SSTORE} như được chỉ định trong phần thích hợp dưới đây. Chúng ta định nghĩa các tập hợp con sau của các chỉ thị (instructions):

W_{zero} = {STOP, RETURN, REVERT}

W_{base} = {ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, PREVRANDAO, GASLIMIT, CHAINID, RETURNDATASIZE, POP, PC, MSIZE, GAS, BASEFEE}

W_{verylow} = {ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, SHL, SHR, SAR, CALLDATALOAD, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}

W_{low} = {MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND, SELFBALANCE}

W_{mid} = {ADDMOD, MULMOD, JUMP}

W_{high} = {JUMPI}

W_{copy} = {CALLDATACOPY, CODECOPY, RETURNDATACOPY}

W_{call} = {CALL, CALLCODE, DELEGATECALL, STATICCALL}

$W_{\text{extaccount}}$ = {BALANCE, EXTCODESIZE, EXTCODEHASH}

Lưu ý rằng thành phần chi phí bộ nhớ, được đưa ra dưới dạng tích của G_{memory} và tối đa giữa 0 & trần số của từ (words) để bộ nhớ trở nên lớn hơn số lượng từ (words) hiện tại, $\boldsymbol{\mu}_{\text{i}}$, để tất cả các truy cập đều tham chiếu đến bộ nhớ hợp lệ, cho cả đọc và ghi. Những truy cập này phải là cho một số lượng byte khác không.

Việc tham chiếu đến một phạm vi có độ dài bằng không (ví dụ: cố gắng truyền nó như là phạm vi đầu vào của một CALL) không đòi hỏi mở rộng bộ nhớ về phía đầu phạm vi. $\boldsymbol{\mu}'_{\mathrm{i}}$ được định nghĩa là số từ (words) tối đa mới của bộ nhớ hoạt động (active); có các trường hợp đặc biệt là khi chúng không bằng nhau.

Lưu ý rằng C_{mem} là hàm chi phí bộ nhớ (hàm mở rộng là sự chênh lệch giữa chi phí trước và sau). Nó là một đa thức, với hệ số bậc cao chia và làm tròn xuống, và do đó tuyến tính cho đến khi sử dụng 704B bộ nhớ, sau đó chi phí nó cao hơn đáng kể.

Trong quá trình định nghĩa bộ chỉ thị (instruction set), chúng ta đã định nghĩa hàm mở rộng bộ nhớ cho phạm vi, M, như sau:

$$M(s, f, l) \equiv \begin{cases} s & \text{nếu} \quad l = 0 \\ \max(s, \lceil (f + l) \div 32 \rceil) & \text{ngược lại} \end{cases} \tag{320}$$

Một hàm hữu ích khác là hàm "tất cả trừ 1/64" L được định nghĩa như sau:

$$L(n) \equiv n - \lfloor n/64 \rfloor \tag{321}$$

H.2. **Bộ chỉ thị (Instruction Set).** Như đã được xác định trước đó trong phần 9, những định nghĩa này diễn ra trong ngữ cảnh cuối cùng đó. Cụ thể, chúng ta giả định O là hàm tiến triển trạng thái của EVM và định nghĩa các thuật ngữ liên quan đến trạng thái của chu kỳ tiếp theo $(\boldsymbol{\sigma}', \boldsymbol{\mu}')$ như sau:

$$O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}', A', I) \quad \text{với các ngoại lệ, như đã ghi chú} \tag{322}$$

Dưới đây là các ngoại lệ khác nhau cho các quy tắc chuyển trạng thái được chỉ định trong phần 9 được xác định cho từng chỉ thị, cùng với các định nghĩa cụ thể cho J và C dựa trên từng chỉ thị. Đối với mỗi chỉ thị, cũng được xác định α, các phần tử được thêm vào ngăn xếp, và δ, các phần tử bị loại khỏi ngăn xếp, như đã được định nghĩa trong phần 9.

0s: Dừng và Các Phép Toán Số Học (Stop and Arithmetic Operations)

Tất cả các phép toán số học đều là modulo 2^{256} trừ khi có ghi chú khác. Lũy thừa bậc 0 của 0, 0^0, được định nghĩa là một.

Giá trị	Mnemonic	δ	α	Mô tả
0x00	STOP	0	0	Dừng thực thi.
0x01	ADD	2	1	Phép cộng. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x02	MUL	2	1	Phép nhân. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] \times \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x03	SUB	2	1	Phép trừ. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] - \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x04	DIV	2	1	Phép chia lấy phần nguyên. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \lfloor \boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1] \rfloor & \text{ngược lại} \end{cases}$
0x05	SDIV	2	1	Phép chia lấy phần nguyên có dấu (cắt bớt (truncated)). $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ -2^{255} & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] = -2^{255} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] = -1 \\ \mathbf{sgn}(\boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1]) \lfloor \lvert \boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1] \rvert \rfloor & \text{ngược lại} \end{cases}$ Trong đó, tất cả các giá trị được xử lý như số nguyên có dấu 256-bit theo phương pháp bù hai Lưu ý về quy tắc tràn (semantic overflow) khi -2^{255} bị phủ định (negated).
0x06	MOD	2	1	Phép còn dư modulo. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \boldsymbol{\mu}_{\mathbf{s}}[0] \bmod \boldsymbol{\mu}_{\mathbf{s}}[1] & \text{ngược lại} \end{cases}$
0x07	SMOD	2	1	Phép còn dư có dấu modulo. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \mathbf{sgn}(\boldsymbol{\mu}_{\mathbf{s}}[0])(\lvert \boldsymbol{\mu}_{\mathbf{s}}[0] \rvert \bmod \lvert \boldsymbol{\mu}_{\mathbf{s}}[1] \rvert) & \text{ngược lại} \end{cases}$ Trong đó, tất cả các giá trị được xử lý như số nguyên có dấu 256-bit theo phương pháp bù hai
0x08	ADDMOD	3	1	Phép cộng modulo. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] = 0 \\ (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1]) \bmod \boldsymbol{\mu}_{\mathbf{s}}[2] & \text{ngược lại} \end{cases}$ Tất cả các tính toán trung gian của phép toán này không phải là modulo 2^{256}.
0x09	MULMOD	3	1	Phép nhân modulo. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] = 0 \\ (\boldsymbol{\mu}_{\mathbf{s}}[0] \times \boldsymbol{\mu}_{\mathbf{s}}[1]) \bmod \boldsymbol{\mu}_{\mathbf{s}}[2] & \text{ngược lại} \end{cases}$ Tất cả các tính toán trung gian của phép toán này không phải là modulo 2^{256}.
0x0a	EXP	2	1	Phép mũ. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]^{\boldsymbol{\mu}_{\mathbf{s}}[1]}$
0x0b	SIGNEXTEND	2	1	Mở rộng chiều dài của số nguyên có dấu theo phương pháp bù hai. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_{\mathrm{i}} \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[1]_{\mathrm{t}} & \text{nếu} \quad i \leqslant t \quad \text{trong đó } t = 256 - 8(\boldsymbol{\mu}_{\mathbf{s}}[0] + 1) \\ \boldsymbol{\mu}_{\mathbf{s}}[1]_{\mathrm{i}} & \text{ngược lại} \end{cases}$

$\boldsymbol{\mu}_{\mathbf{s}}[x]_{\mathrm{i}}$ cho biết bit thứ i (đếm từ 0) của $\boldsymbol{\mu}_{\mathbf{s}}[x]$

10s: So sánh & Các Phép Toán Logic Bitwise

Giá trị	Mnemonic	δ	α	Mô tả
0x10	LT	2	1	Phép so sánh nhỏ hơn. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] < \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{ngược lại} \end{cases}$
0x11	GT	2	1	Phép so sánh lớn hơn. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] > \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{ngược lại} \end{cases}$
0x12	SLT	2	1	Phép so sánh nhỏ hơn có dấu. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] < \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{ngược lại} \end{cases}$ Trong đó, tất cả các giá trị được xử lý như số nguyên có dấu 256-bit theo phương pháp bù hai.
0x13	SGT	2	1	Phép so sánh lớn hơn có dấu. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] > \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{ngược lại} \end{cases}$ Trong đó, tất cả các giá trị được xử lý như số nguyên có dấu 256-bit theo phương pháp bù hai.
0x14	EQ	2	1	Phép so sánh bằng. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] = \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{ngược lại} \end{cases}$
0x15	ISZERO	1	1	Toán tử NOT đơn giản. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] = 0 \\ 0 & \text{ngược lại} \end{cases}$
0x16	AND	2	1	Phép AND bitwise. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \wedge \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x17	OR	2	1	Phép OR bitwise. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \vee \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x18	XOR	2	1	Phép XOR bitwise. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \oplus \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x19	NOT	1	1	Phép NOT bitwise. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \begin{cases} 1 & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0]_i = 0 \\ 0 & \text{ngược lại} \end{cases}$
0x1a	BYTE	2	1	Lấy một byte từ một từ. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[1]_{(i-248+8\boldsymbol{\mu}_{\mathbf{s}}[0])} & \text{nếu} \quad i \geq 248 \wedge \boldsymbol{\mu}_{\mathbf{s}}[0] < 32 \\ 0 & \text{ngược lại} \end{cases}$ Đối với byte thứ N, chúng ta đếm từ trái sang (tức là N=0 sẽ là byte quan trọng nhất theo thứ tự big endian).
0x1b	SHL	2	1	Phép dịch trái. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv (\boldsymbol{\mu}_{\mathbf{s}}[1] \times 2^{\boldsymbol{\mu}_{\mathbf{s}}[0]}) \bmod 2^{256}$
0x1c	SHR	2	1	Phép dịch phải logic. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \lfloor \boldsymbol{\mu}_{\mathbf{s}}[1] \div 2^{\boldsymbol{\mu}_{\mathbf{s}}[0]} \rfloor$
0x1d	SAR	2	1	Phép dịch phải toán tử (có dấu). $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \lfloor \boldsymbol{\mu}_{\mathbf{s}}[1] \div 2^{\boldsymbol{\mu}_{\mathbf{s}}[0]} \rfloor$ Trong đó, $\boldsymbol{\mu}'_{\mathbf{s}}[0]$ và $\boldsymbol{\mu}_{\mathbf{s}}[1]$ được xử lý như số nguyên có dấu 256-bit theo phương pháp bù hai, trong khi $\boldsymbol{\mu}_{\mathbf{s}}[0]$ được xử lý như số nguyên không dấu.

20s: KECCAK256

Giá trị	Mnemonic	δ	α	Description
0x20	KECCAK256	2	1	Tính toán KECCAK-256 hash. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \texttt{KEC}(\boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \ldots (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1] - 1)])$ $\boldsymbol{\mu}'_i \equiv M(\boldsymbol{\mu}_i, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[1])$

30s: Thông Tin Môi Trường

Giá trị	Mnemonic	δ	α	Mô tả
0x30	ADDRESS	0	1	Lấy địa chỉ của tài khoản đang thực thi. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{a}}$
0x31	BALANCE	1	1	Lấy số dư của tài khoản được chỉ định. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}]_{\mathrm{b}} & \text{nếu} \quad \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}] \neq \varnothing \\ 0 & \text{ngược lại} \end{cases}$ $A'_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}\}$
0x32	ORIGIN	0	1	Lấy địa chỉ gốc của quá trình thực thi. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{o}}$ Đây là người gửi giao dịch ban đầu; không bao giờ là một tài khoản có mã nguồn liên quan
0x33	CALLER	0	1	Lấy địa chỉ của tài khoản gọi hàm. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{s}}$ Đây là địa chỉ của tài khoản trực tiếp chịu trách nhiệm cho quá trình thực thi này.
0x34	CALLVALUE	0	1	Lấy giá trị được gửi kèm bởi chỉ thị/giao dịch gây ra quá trình thực thi này. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{v}}$
0x35	CALLDATALOAD	1	1	Lấy dữ liệu đầu vào của môi trường hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathbf{d}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)] \quad \text{với} \quad I_{\mathbf{d}}[x] = 0 \quad \text{nếu} \quad x \geqslant \|I_{\mathbf{d}}\|$ Điều này liên quan đến dữ liệu đầu vào được truyền với chỉ thị gọi tin nhắn hoặc giao dịch.
0x36	CALLDATASIZE	0	1	Lấy kích thước dữ liệu đầu vào trong môi trường hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|I_{\mathbf{d}}\|$ Điều này liên quan đến dữ liệu đầu vào được truyền với chỉ thị gọi tin nhắn hoặc giao dịch.
0x37	CALLDATACOPY	3	0	Sao chép dữ liệu đầu vào trong môi trường hiện tại vào bộ nhớ. $\forall i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2] - 1\} : \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] + i] \equiv \begin{cases} I_{\mathbf{d}}[\boldsymbol{\mu}_{\mathbf{s}}[1] + i] & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] + i < \|I_{\mathbf{d}}\| \\ 0 & \text{ngược lại} \end{cases}$ Cộng dồn trong $\boldsymbol{\mu}_{\mathbf{s}}[1] + i$ không phải theo modulo 2^{256}. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$ Điều này liên quan đến dữ liệu đầu vào được truyền với chỉ thị gọi tin nhắn hoặc giao dịch.
0x38	CODESIZE	0	1	Lấy kích thước mã chạy trong môi trường hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|I_{\mathbf{b}}\|$
0x39	CODECOPY	3	0	Sao chép mã chạy trong môi trường hiện tại vào bộ nhớ. $\forall i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2] - 1\} : \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] + i] \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\mathbf{s}}[1] + i] & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] + i < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{ngược lại} \end{cases}$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$ Cộng dồn trong $\boldsymbol{\mu}_{\mathbf{s}}[1] + i$ không phải theo modulo 2^{256}.
0x3a	GASPRICE	0	1	Lấy giá trị của gas trong môi trường hiện tại. Đây là *giá trị hiệu quả của gas* được xác định trong phần 6. Lưu ý rằng từ *London* hard fork trở đi, giá trị này không còn đại diện cho giá trị nhận được bởi người xác nhận, mà chỉ là giá trị được người gửi trả. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{p}}$
0x3b	EXTCODESIZE	1	1	Lấy kích thước mã của một tài khoản. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} \|\mathbf{b}\| & \text{nếu} \quad \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}] \neq \varnothing \\ 0 & \text{ngược lại} \end{cases}$ với $\texttt{KEC}(\mathbf{b}) \equiv \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}]_{\mathrm{c}}$ $A'_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}\}$
0x3c	EXTCODECOPY	4	0	Sao chép mã của một tài khoản vào bộ nhớ. $\forall i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[3] - 1\} : \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[1] + i] \equiv \begin{cases} \mathbf{b}[\boldsymbol{\mu}_{\mathbf{s}}[2] + i] & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] + i < \|\mathbf{b}\| \\ \text{STOP} & \text{ngược lại} \end{cases}$ với $\texttt{KEC}(\mathbf{b}) \equiv \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}]_{\mathrm{c}}$ Chúng ta giả sử $\mathbf{b} \equiv ()$ nếu $\boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}] = \varnothing$. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[1], \boldsymbol{\mu}_{\mathbf{s}}[3])$ Cộng dồn trong $\boldsymbol{\mu}_{\mathbf{s}}[2] + i$ không phải theo modulo 2^{256}. $A'_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}\}$

0x3d	RETURNDATASIZE	0	1	Lấy kích thước dữ liệu đầu ra từ lời gọi trước đó từ môi trường hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|\boldsymbol{\mu}_{\mathbf{o}}\|$
0x3e	RETURNDATACOPY	3	0	Sao chép dữ liệu đầu ra từ lời gọi trước đó vào bộ nhớ. $\forall i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2] - 1\} : \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] + i] \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{o}}[\boldsymbol{\mu}_{\mathbf{s}}[1] + i] & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] + i < \|\boldsymbol{\mu}_{\mathbf{o}}\| \\ 0 & \text{ngược lại} \end{cases}$ Cộng trong $\boldsymbol{\mu}_{\mathbf{s}}[1] + i$ không phải là phép toán modulo 2^{256}. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$
0x3f	EXTCODEHASH	1	1	Lấy mã hash của tài khoản. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{nếu} \quad \texttt{DEAD}(\boldsymbol{\sigma}, \boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}) \\ \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}]_{\mathrm{c}} & \text{ngược lại} \end{cases}$ $A'_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{\boldsymbol{\mu}_{\mathbf{s}}[0] \bmod 2^{160}\}$

40s: Thông tin Khối

Giá trị	Mnemonic	δ	α	Mô tả
0x40	BLOCKHASH	1	1	Lấy giá trị hash của một trong 256 khối gần đây nhất đã hoàn thành. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv P(I_{\mathrm{H_p}}, \boldsymbol{\mu}_{\mathbf{s}}[0], 0)$ với P là hash của một khối cụ thể, tới một tuổi tối đa . 0 được để lại trên ngăn xếp nếu số khối cần tìm lớn hơn hoặc bằng số khối hiện tại hoặc nhiều hơn 256 khối so với khối hiện tại. $P(h, n, a) \equiv \begin{cases} 0 & \text{nếu} \quad n > H_{\mathrm{i}} \vee a = 256 \vee h = 0 \\ h & \text{nếu} \quad n = H_{\mathrm{i}} \\ P(H_{\mathrm{p}}, n, a + 1) & \text{ngược lại} \end{cases}$ và chúng ta khẳng định rằng tiêu đề H có thể được xác định từ hash của nó h trừ khi h bằng (như là trường hợp của hash cha của khối khởi tạo).
0x41	COINBASE	0	1	Lấy địa chỉ hưởng của khối hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_c}}$
0x42	TIMESTAMP	0	1	Lấy thời điểm của khối hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_s}}$
0x43	NUMBER	0	1	Lấy số của khối hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_i}}$
0x44	PREVRANDAO	0	1	Lấy trộn RANDAO mới nhất của trạng thái post beacon của khối trước. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_a}}$
0x45	GASLIMIT	0	1	Lấy giới hạn gas của khối hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_l}}$
0x46	CHAINID	0	1	Lấy ID chuỗi. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \beta$
0x47	SELFBALANCE	0	1	Lấy số dư của tài khoản đang thực thi hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}}$
0x48	BASEFEE	0	1	Lấy phí cố định của khối hiện tại. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{H_f}}$

50s: Thao tác Stack, Memory, Storage và Flow

Giá trị	Mnemonic	δ	α	Mô tả
0x50	POP	1	0	Loại bỏ mục từ ngăn xếp.
0x51	MLOAD	1	1	Đọc từ bộ nhớ một từ. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)]$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 32) \div 32 \rceil)$ Phép cộng trong tính toán của $\boldsymbol{\mu}'_{\mathrm{i}}$ không chịu phép toán modulo 2^{256}.
0x52	MSTORE	2	0	Lưu từ vào bộ nhớ. $\boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 32) \div 32 \rceil)$ Phép cộng trong tính toán của $\boldsymbol{\mu}'_{\mathrm{i}}$ không chịu phép toán modulo 2^{256}.
0x53	MSTORE8	2	0	Lưu byte vào bộ nhớ. $\boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] \equiv (\boldsymbol{\mu}_{\mathbf{s}}[1] \bmod 256)$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 1) \div 32 \rceil)$ Phép cộng trong tính toán của $\boldsymbol{\mu}'_{\mathrm{i}}$ không chịu phép toán modulo 2^{256}.
0x54	SLOAD	1	1	Đọc từ kho từ bộ nhớ. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]]$ $A'_{\mathbf{K}} \equiv A_{\mathbf{K}} \cup \{(I_{\mathrm{a}}, \boldsymbol{\mu}_{\mathbf{s}}[0])\}$ $C_{\mathrm{SLOAD}}(\boldsymbol{\mu}, A, I) \equiv \begin{cases} G_{\mathrm{warmaccess}} & \text{nếu } (I_{\mathrm{a}}, \boldsymbol{\mu}_{\mathbf{s}}[0]) \in A_{\mathbf{K}} \\ G_{\mathrm{coldsload}} & \text{ngược lại} \end{cases}$
0x55	SSTORE	2	0	Lưu từ vào kho bộ nhớ. $\boldsymbol{\sigma}'[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $A'_{\mathbf{K}} \equiv A_{\mathbf{K}} \cup \{(I_{\mathrm{a}}, \boldsymbol{\mu}_{\mathbf{s}}[0])\}$ $C_{\mathrm{SSTORE}}(\boldsymbol{\sigma}, \boldsymbol{\mu})$ và A'_{r} được chỉ định bởi EIP-2200 như sau. Chúng ta nhắc nhở độc giả rằng trạng thái kiểm tra ("gốc") $\boldsymbol{\sigma}_0$ là trạng thái nếu giao dịch hiện tại bị quay lại. Hãy để $v_0 = \boldsymbol{\sigma}_0[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]]$ là giá trị ban đầu của kho bộ nhớ. Hãy để $v = \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]]$ là giá trị hiện tại. Hãy để $v' = \boldsymbol{\mu}_{\mathbf{s}}[1]$ là giá trị mới. Khi đó: $C_{\mathrm{SSTORE}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \equiv \begin{cases} 0 & \text{nếu } (I_{\mathrm{a}}, \boldsymbol{\mu}_{\mathbf{s}}[0]) \in A_{\mathbf{K}} \\ G_{\mathrm{coldsload}} & \text{ngược lại} \end{cases}$ $+ \begin{cases} G_{\mathrm{warmaccess}} & \text{nếu } v = v' \vee v_0 \neq v \\ G_{\mathrm{sset}} & \text{nếu } v \neq v' \wedge v_0 = v \wedge v_0 = 0 \\ G_{\mathrm{sreset}} & \text{nếu } v \neq v' \wedge v_0 = v \wedge v_0 \neq 0 \end{cases}$ $A'_{\mathrm{r}} \equiv A_{\mathrm{r}} + \begin{cases} R_{\mathrm{sclear}} & \text{nếu } v \neq v' \wedge v_0 = v \wedge v' = 0 \\ r_{\mathrm{dirtyclear}} + r_{\mathrm{dirtyreset}} & \text{nếu } v \neq v' \wedge v_0 \neq v \\ 0 & \text{ngược lại} \end{cases}$ trong đó $r_{\mathrm{dirtyclear}} \equiv \begin{cases} -R_{\mathrm{sclear}} & \text{nếu } v_0 \neq 0 \wedge v = 0 \\ R_{\mathrm{sclear}} & \text{nếu } v_0 \neq 0 \wedge v' = 0 \\ 0 & \text{ngược lại} \end{cases}$ $r_{\mathrm{dirtyreset}} \equiv \begin{cases} G_{\mathrm{sset}} - G_{\mathrm{warmaccess}} & \text{nếu } v_0 = v' \wedge v_0 = 0 \\ G_{\mathrm{sreset}} - G_{\mathrm{warmaccess}} & \text{nếu } v_0 = v' \wedge v_0 \neq 0 \\ 0 & \text{ngược lại} \end{cases}$
0x56	JUMP	1	0	Thay đổi bộ đếm chương trình. $J_{\mathrm{JUMP}}(\boldsymbol{\mu}) \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$ Điều này có tác dụng viết giá trị đó vào $\boldsymbol{\mu}_{\mathrm{pc}}$. Xem phần 9.
0x57	JUMPI	2	0	Thay đổi điều kiện bộ đếm chương trình. $J_{\mathrm{JUMPI}}(\boldsymbol{\mu}) \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[0] & \text{nếu } \boldsymbol{\mu}_{\mathbf{s}}[1] \neq 0 \\ \boldsymbol{\mu}_{\mathrm{pc}} + 1 & \text{ngược lại} \end{cases}$ Điều này có tác dụng viết giá trị đó vào $\boldsymbol{\mu}_{\mathrm{pc}}$. Xem phần 9.
0x58	PC	0	1	Lấy giá trị bộ đếm chương trình *trước* khi tăng tương ứng với lệnh này. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathrm{pc}}$

0x59	MSIZE	0	1	Lấy kích thước bộ nhớ hoạt động hiện tại trong byte. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv 32\boldsymbol{\mu}_{\mathrm{i}}$
0x5a	GAS	0	1	Lấy lượng gas khả dụng, bao gồm giảm tương ứng cho chi phí của lệnh này. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathrm{g}}$
0x5b	JUMPDEST	0	0	Đánh dấu một đích hợp lệ cho các lệnh nhảy. Thao tác này không ảnh hưởng đến trạng thái máy trong quá trình thực thi.

60s & 70s: Các Phép Toán Đẩy (Push)

Giá trị	Mnemonic	δ	α	Mô Tả
0x60	PUSH1	0	1	Đặt một byte lên ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv c(\boldsymbol{\mu}_{\mathrm{pc}} + 1)$ trong đó $c(x) \equiv \begin{cases} I_{\mathbf{b}}[x] & \text{nếu } x < \lVert I_{\mathbf{b}} \rVert \\ 0 & \text{ngược lại} \end{cases}$ Các byte được đọc theo dòng từ mảng byte của mã chương trình. Hàm c đảm bảo các byte mặc định là không nếu chúng vượt quá giới hạn. Byte được căn chỉnh về bên phải (lấy vị trí quan trọng nhất trong big endian).
0x61	PUSH2	0	1	Đặt một mục 2 byte lên ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{c}\big((\boldsymbol{\mu}_{\mathrm{pc}} + 1) \dots (\boldsymbol{\mu}_{\mathrm{pc}} + 2)\big)$ với $\boldsymbol{c}(\boldsymbol{x}) \equiv (c(\boldsymbol{x}_0), ..., c(\boldsymbol{x}_{\lVert x \rVert - 1}))$ với c được định nghĩa như trên. Các byte được căn chỉnh về bên phải (lấy vị trí quan trọng nhất trong big endian).
⋮	⋮	⋮	⋮	⋮
0x7f	PUSH32	0	1	Đặt một mục 32 byte (tức là từ đầy đủ) lên ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{c}\big((\boldsymbol{\mu}_{\mathrm{pc}} + 1) \dots (\boldsymbol{\mu}_{\mathrm{pc}} + 32)\big)$ trong đó $\boldsymbol{c}$ được định nghĩa như trên. Các byte được căn chỉnh về bên phải (lấy vị trí quan trọng nhất trong big endian).

80s: Các Phép Toán Nhân Bản (Duplication)

Giá trị	Mnemonic	δ	α	Mô Tả
0x80	DUP1	1	2	Nhân bản mục 1 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
0x81	DUP2	2	3	Nhân bản mục 2 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$
⋮	⋮	⋮	⋮	⋮
0x8f	DUP16	16	17	Nhân bản mục thứ 16 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[15]$

90s: Các Phép Toán Trao Đổi (Exchange)

Giá trị	Mnemonic	δ	α	Mô Tả
0x90	SWAP1	2	2	Trao đổi mục 1 và mục 2 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $\boldsymbol{\mu}'_{\mathbf{s}}[1] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
0x91	SWAP2	3	3	Trao đổi mục 1 và mục 3 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[2]$ $\boldsymbol{\mu}'_{\mathbf{s}}[2] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
⋮	⋮	⋮	⋮	⋮
0x9f	SWAP16	17	17	Trao đổi mục 1 và mục 17 của ngăn xếp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[16]$ $\boldsymbol{\mu}'_{\mathbf{s}}[16] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$

a0s: Các Phép Toán Ghi Log

Đối với tất cả các phép toán ghi log, sự thay đổi trạng thái là để thêm một bản ghi log bổ sung vào chuỗi log của trạng thái con:
$A'_{\mathbf{l}} \equiv A_{\mathbf{l}} \cdot (I_{\mathrm{a}}, \mathbf{t}, \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \ldots (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1] - 1)])$
và để cập nhật bộ đếm tiêu thụ bộ nhớ:
$\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[1])$
Dãy chủ đề của bản ghi, $\mathbf{t}$, thay đổi tùy thuộc vào số lượng chủ đề:

Giá trị	**Mnemonic**	δ	α	**Mô Tả**
0xa0	LOG0	2	0	Thêm bản ghi log không có chủ đề nào. $\mathbf{t} \equiv ()$
0xa1	LOG1	3	0	Thêm bản ghi log có một chủ đề. $\mathbf{t} \equiv (\boldsymbol{\mu}_{\mathbf{s}}[2])$
⋮	⋮	⋮	⋮	⋮
0xa4	LOG4	6	0	Thêm bản ghi log có bốn chủ đề. $\mathbf{t} \equiv (\boldsymbol{\mu}_{\mathbf{s}}[2], \boldsymbol{\mu}_{\mathbf{s}}[3], \boldsymbol{\mu}_{\mathbf{s}}[4], \boldsymbol{\mu}_{\mathbf{s}}[5])$

f0s: Hoạt động hệ thống (System operations)

Giá trị	Mnemonic	δ	α	Mô tả
0xf0	CREATE	3	1	Tạo một tài khoản mới với mã (code) được liên kết. $\mathbf{i} \equiv \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[1] \dots (\boldsymbol{\mu}_{\mathbf{s}}[1] + \boldsymbol{\mu}_{\mathbf{s}}[2] - 1)]$ $\zeta \equiv \varnothing$ $(\boldsymbol{\sigma}', g', A', z, \mathbf{o}) \equiv \begin{cases} \Lambda(\boldsymbol{\sigma}^*, A, I_{\mathrm{a}}, I_{\mathrm{o}}, L(\boldsymbol{\mu}_{\mathrm{g}}), I_{\mathrm{p}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \mathbf{i}, I_{\mathrm{e}}+1, \zeta, I_{\mathrm{w}}) & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] \leqslant \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}} \\ & \quad \wedge\, I_{\mathrm{e}} < 1024 \\ (\boldsymbol{\sigma}, L(\boldsymbol{\mu}_{\mathrm{g}}), A, 0, ()) & \text{ngược lại} \end{cases}$ $\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma} \quad$ ngoại trừ $\quad \boldsymbol{\sigma}^*[I_{\mathrm{a}}]_{\mathrm{n}} = \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{n}} + 1$ $\boldsymbol{\mu}'_{\mathrm{g}} \equiv \boldsymbol{\mu}_{\mathrm{g}} - L(\boldsymbol{\mu}_{\mathrm{g}}) + g'$ $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv x$ trong đó $x = 0$ if $z = 0$, i.e., quá trình tạo hợp đồng không thành công, hoặc $I_{\mathrm{e}} = 1024$ (đạt đến giới hạn độ sâu cuộc gọi tối đa) hoặc $\boldsymbol{\mu}_{\mathbf{s}}[0] > \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}}$ (balance của người gọi quá thấp để hoàn thành việc chuyển (transfer) giá trị); và nếu không $x = \texttt{ADDR}(I_{\mathrm{a}}, \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{n}}, \zeta, \mathbf{i})$, th địa chỉ của tài khoản mới được tạo (89). $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[1], \boldsymbol{\mu}_{\mathbf{s}}[2])$ $\boldsymbol{\mu}'_{\mathbf{o}} \equiv \begin{cases} () & \text{nếu} \quad z = 1 \\ \mathbf{o} & \text{ngược lại} \end{cases}$ Do đó, thứ tự toán hạng là: giá trị, offset đầu vào, kích thước đầu vào.
0xf1	CALL	7	1	Gọi tin nhắn vào một tài khoản. $\mathbf{i} \equiv \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[3] \dots (\boldsymbol{\mu}_{\mathbf{s}}[3] + \boldsymbol{\mu}_{\mathbf{s}}[4] - 1)]$ $(\boldsymbol{\sigma}', g', A', x, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}, A^*, I_{\mathrm{a}}, I_{\mathrm{o}}, t, t, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] \leqslant \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}} \wedge \\ \quad I_{\mathrm{p}}, \boldsymbol{\mu}_{\mathbf{s}}[2], \boldsymbol{\mu}_{\mathbf{s}}[2], \mathbf{i}, I_{\mathrm{e}}+1, I_{\mathrm{w}}) & \quad I_{\mathrm{e}} < 1024 \\ (\boldsymbol{\sigma}, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), A, 0, ()) & \text{ngược lại} \end{cases}$ $n \equiv \min(\{\boldsymbol{\mu}_{\mathbf{s}}[6], \|\mathbf{o}\|\})$ $\boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[5] \dots (\boldsymbol{\mu}_{\mathbf{s}}[5] + n - 1)] = \mathbf{o}[0 \dots (n-1)]$ $\boldsymbol{\mu}'_{\mathbf{o}} = \mathbf{o}$ $\boldsymbol{\mu}'_{\mathrm{g}} \equiv \boldsymbol{\mu}_{\mathrm{g}} - C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) + g'$ $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv x$ $A^* \equiv A \quad$ ngoại trừ $\quad A^*_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{t\}$ $t \equiv \boldsymbol{\mu}_{\mathbf{s}}[1] \bmod 2^{160}$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[3], \boldsymbol{\mu}_{\mathbf{s}}[4]), \boldsymbol{\mu}_{\mathbf{s}}[5], \boldsymbol{\mu}_{\mathbf{s}}[6])$ trong đó $x = 0$ nếu thực thi mã cho thao tác này không thành công, hoặc nếu $\boldsymbol{\mu}_{\mathbf{s}}[2] > \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}}$ (không đủ tiền) hoặc $I_{\mathrm{e}} = 1024$ (giới hạn độ sâu cuộc gọi đã đạt đến); $x = 1$ nếu không thì. Do đó, thứ tự toán hạng là: gas, to, value, in offset, in size, out offset, out size. $C_{\mathrm{CALL}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) \equiv C_{\mathrm{GASCAP}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) + C_{\mathrm{EXTRA}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A)$ $C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) \equiv \begin{cases} C_{\mathrm{GASCAP}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) + G_{\mathrm{callstipend}} & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] \neq 0 \\ C_{\mathrm{GASCAP}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) & \text{ngược lại} \end{cases}$ $C_{\mathrm{GASCAP}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) \equiv \begin{cases} \min\{L(\boldsymbol{\mu}_{\mathrm{g}} - C_{\mathrm{EXTRA}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A)), \boldsymbol{\mu}_{\mathbf{s}}[0]\} & \text{nếu} \quad \boldsymbol{\mu}_{\mathrm{g}} \geq C_{\mathrm{EXTRA}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) \\ \boldsymbol{\mu}_{\mathbf{s}}[0] & \text{ngược lại} \end{cases}$ $C_{\mathrm{EXTRA}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A) \equiv C_{\mathrm{aaccess}}(t, A) + C_{\mathrm{XFER}}(\boldsymbol{\mu}) + C_{\mathrm{NEW}}(\boldsymbol{\sigma}, \boldsymbol{\mu})$ $C_{\mathrm{XFER}}(\boldsymbol{\mu}) \equiv \begin{cases} G_{\mathrm{callvalue}} & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] \neq 0 \\ 0 & \text{ngược lại} \end{cases}$ $C_{\mathrm{NEW}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv \begin{cases} G_{\mathrm{newaccount}} & \text{nếu} \quad \texttt{DEAD}(\boldsymbol{\sigma}, t) \wedge \boldsymbol{\mu}_{\mathbf{s}}[2] \neq 0 \\ 0 & \text{ngược lại} \end{cases}$
0xf2	CALLCODE	7	1	Gọi tin nhắn vào tài khoản này với mã nguồn của một tài khoản thay thế. Hoàn toàn tương đương với CALL ngoại trừ: $(\boldsymbol{\sigma}', g', A', x, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}, A^*, I_{\mathrm{a}}, I_{\mathrm{o}}, I_{\mathrm{a}}, t, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), & \text{nếu} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] \leqslant \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathrm{b}} \wedge \\ \quad I_{\mathrm{p}}, \boldsymbol{\mu}_{\mathbf{s}}[2], \boldsymbol{\mu}_{\mathbf{s}}[2], \mathbf{i}, I_{\mathrm{e}}+1, I_{\mathrm{w}}) & \quad I_{\mathrm{e}} < 1024 \\ (\boldsymbol{\sigma}, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), A, 0, ()) & \text{ngược lại} \end{cases}$ Lưu ý sự thay đổi trong tham số thứ tư của cuộc gọi Θ từ giá trị stack thứ 2 $\boldsymbol{\mu}_{\mathbf{s}}[1]$ (như trong CALL) đến địa chỉ hiện tại I_{a}. Điều này có nghĩa là người nhận là thực sự cùng một tài khoản như hiện tại, chỉ là mã nguồn bị ghi đè.
0xf3	RETURN	2	0	Dừng thực thi và trả kết quả dữ liệu đầu ra. $H_{\mathrm{RETURN}}(\boldsymbol{\mu}) \equiv \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1] - 1)]$ Điều này có tác dụng dừng thực thi tại điểm này với dữ liệu đầu ra được xác định. Xem phần 9. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[1])$

0xf4	DELEGATECALL	6	1	Gọi tin nhắn vào tài khoản này với mã nguồn của một tài khoản thay thế, nhưng giữ các giá trị hiện tại của *người gửi* và *giá trị*. So với CALL, DELEGATECALL ít một tham số. Tham số bị bỏ qua là $\boldsymbol{\mu}_\mathbf{s}[2]$. Do đó, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$, $\boldsymbol{\mu}_\mathbf{s}[5]$ và $\boldsymbol{\mu}_\mathbf{s}[6]$ trong định nghĩa của CALL lần lượt được thay thế bằng $\boldsymbol{\mu}_\mathbf{s}[2]$, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$ và $\boldsymbol{\mu}_\mathbf{s}[5]$. Nếu không, nó tương đương với CALL ngoại trừ: $(\boldsymbol{\sigma}', g', A', x, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}, A^*, I_\mathrm{s}, I_\mathrm{o}, I_\mathrm{a}, t, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), & \text{nếu} \quad I_\mathrm{e} < 1024 \\ \quad I_\mathrm{p}, 0, I_\mathrm{v}, \mathbf{i}, I_\mathrm{e}+1, I_\mathrm{w}) & \\ (\boldsymbol{\sigma}, C_{\mathrm{CALLGAS}}(\boldsymbol{\sigma}, \boldsymbol{\mu}, A), A, 0, ()) & \text{ngược lại} \end{cases}$ Lưu ý các thay đổi (ngoài thay đổi của tham số thứ tư) đối với tham số thứ hai và thứ chín trong cuộc gọi Θ. Điều này có nghĩa là người nhận thực sự là cùng một tài khoản như hiện tại, đơn giản là mã nguồn bị ghi đè *và* ngữ cảnh gần như hoàn toàn giống nhau.
0xf5	CREATE2	4	1	Tạo một tài khoản mới với mã nguồn tương ứng. Hoàn toàn tương đương với CREATE ngoại trừ: Giá trị của salt $\zeta \equiv \boldsymbol{\mu}_\mathbf{s}[3]$.
0xfa	STATICCALL	6	1	Gọi tin nhắn tĩnh vào một tài khoản. Hoàn toàn tương đương với CALL ngoại trừ: Tham số $\boldsymbol{\mu}_\mathbf{s}[2]$ được thay thế bằng 0. Các tham số sâu hơn $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$, $\boldsymbol{\mu}_\mathbf{s}[5]$ và $\boldsymbol{\mu}_\mathbf{s}[6]$ được thay thế lần lượt bằng $\boldsymbol{\mu}_\mathbf{s}[2]$, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$ và $\boldsymbol{\mu}_\mathbf{s}[5]$. Tham số cuối cùng của Θ là $\bot$.
0xfd	REVERT	2	0	Dừng thực thi và hoàn ngược lại các thay đổi trạng thái, nhưng trả dữ liệu và gas còn lại. $H_{\mathrm{RETURN}}(\boldsymbol{\mu}) \equiv \boldsymbol{\mu}_\mathbf{m}[\boldsymbol{\mu}_\mathbf{s}[0] \dots (\boldsymbol{\mu}_\mathbf{s}[0] + \boldsymbol{\mu}_\mathbf{s}[1] - 1)]$ Hiệu quả của thao tác này được mô tả trong (146). Đối với tính toán gas, chúng ta sử dụng hàm mở rộng bộ nhớ, $\boldsymbol{\mu}_\mathrm{i}' \equiv M(\boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\mu}_\mathbf{s}[0], \boldsymbol{\mu}_\mathbf{s}[1])$
0xfe	INVALID	∅	∅	Chỉ thị không hợp lệ được chỉ định.
0xff	SELFDESTRUCT	1	0	Dừng thực thi và đăng ký tài khoản để xóa sau này. $A'_\mathbf{s} \equiv A_\mathbf{s} \cup \{I_\mathrm{a}\}$ $A'_\mathbf{a} \equiv A_\mathbf{a} \cup \{r\}$ $\boldsymbol{\sigma}'[r] \equiv \begin{cases} \varnothing & \text{nếu} \quad \boldsymbol{\sigma}[r] = \varnothing \ \wedge \ \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} = 0 \\ (\boldsymbol{\sigma}[r]_\mathrm{n}, \boldsymbol{\sigma}[r]_\mathrm{b} + \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b}, \boldsymbol{\sigma}[r]_\mathbf{s}, \boldsymbol{\sigma}[r]_\mathrm{c}) & \text{nếu} \quad r \neq I_\mathrm{a} \\ (\boldsymbol{\sigma}[r]_\mathrm{n}, 0, \boldsymbol{\sigma}[r]_\mathbf{s}, \boldsymbol{\sigma}[r]_\mathrm{c}) & \text{ngược lại} \end{cases}$ trong đó $r = \boldsymbol{\mu}_\mathbf{s}[0] \bmod 2^{160}$ $\boldsymbol{\sigma}'[I_\mathrm{a}]_\mathrm{b} = 0$ $C_{\mathrm{SELFDESTRUCT}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv G_{\mathrm{selfdestruct}} + \begin{cases} 0 & \text{nếu} \quad r \in A_\mathbf{a} \\ G_{\mathrm{coldaccountaccess}} & \text{ngược lại} \end{cases}$ $+ \begin{cases} G_{\mathrm{newaccount}} & \text{nếu} \quad \mathtt{DEAD}(\boldsymbol{\sigma}, r) \wedge \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \neq 0 \\ 0 & \text{ngược lại} \end{cases}$

Appendix I. Genesis Block

Khối khởi tạo gồm 15 mục, và được chỉ định như sau:

$$\big(\big(0_{256}, \mathtt{KEC}\big(\mathtt{RLP}\big(()\big)\big)\big), 0_{160}, \mathrm{stateRoot}, 0, 0, 0_{2048}, 2^{34}, 0, 0, 3141592, \mathrm{time}, 0, 0_{256}, \mathtt{KEC}\big((42)\big)\big), (), ()\big) \tag{323}$$

Trong đó, 0_{256} đề cập đến hash cha, một hash 256-bit toàn bộ là số không; 0_{160} đề cập đến địa chỉ người hưởng, một hash 160-bit toàn bộ là số không; 0_{2048} đề cập đến log bloom, 2048-bit toàn bộ là số không; 2^{34} đề cập đến độ khó; gốc cây giao dịch, gốc cây biên nhận, gas sử dụng, số khối và extradata đều là 0, tương đương với mảng byte rỗng. Cả hai chuỗi của cả ommers và giao dịch đều trống và được biểu diễn bằng $()$. $\mathtt{KEC}\big((42)\big)$ đề cập đến hash Keccak-256 của một mảng byte có độ dài một, với byte đầu tiên và duy nhất có giá trị là 42, được sử dụng cho nonce. Giá trị $\mathtt{KEC}\big(\mathtt{RLP}\big(()\big)\big)$ đề cập đến hash của danh sách ommers trong RLP, cả hai đều là danh sách rỗng.

Các chuỗi thử nghiệm bao gồm một sự phát triển premine, làm cho giá trị gốc của trạng thái là *stateRoot*. Ngoài ra, *time* sẽ được đặt thành timestamp ban đầu của khối khởi tạo. Nên kiểm tra tài liệu mới nhất để có giá trị chính xác.

Appendix J. Ethash

J.1. **Deprecation.** Phần này được giữ lại vì mục đích lịch sử, nhưng thuật toán Ethash không còn được sử dụng cho sự đồng thuận kể từ *Paris* hard fork.

J.2. **Định nghĩa.** Chúng ta sử dụng các định nghĩa sau:

Tên	Giá trị	Mô tả
$J_{\text{wordbytes}}$	4	Số byte trong một từ.
$J_{\text{datasetinit}}$	2^{30}	Số byte trong bộ dữ liệu tại khởi tạo.
$J_{\text{datasetgrowth}}$	2^{23}	Sự tăng của bộ dữ liệu mỗi kỷ nguyên.
$J_{\text{cacheinit}}$	2^{24}	Số byte trong bộ nhớ cache tại khởi tạo.
$J_{\text{cachegrowth}}$	2^{17}	Sự tăng của cache mỗi kỷ nguyên.
J_{epoch}	30000	Số khối mỗi kỷ nguyên.
J_{mixbytes}	128	Độ dài mix trong byte.
$J_{\text{hashbytes}}$	64	Độ dài hash trong byte.
J_{parents}	256	Số lượng cha của mỗi phần tử trong bộ dữ liệu.
$J_{\text{cacherounds}}$	3	Số vòng trong quá trình tạo cache.
J_{accesses}	64	Số lượt truy cập trong vòng lặp hashimoto.

J.3. **Kích thước của bộ dữ liệu và bộ nhớ cache.** Kích thước của bộ nhớ cache $\mathbf{c} \in \mathbb{B}$ và bộ dữ liệu $\mathbf{d} \in \mathbb{B}$ phụ thuộc vào kỷ nguyên, mà lược đồ cũng phụ thuộc vào số khối.

$$E_{\text{epoch}}(H_{\text{i}}) = \left\lfloor \frac{H_{\text{i}}}{J_{\text{epoch}}} \right\rfloor \tag{324}$$

Kích thước của bộ dữ liệu tăng lên $J_{\text{datasetgrowth}}$ byte, và kích thước của cache tăng lên $J_{\text{cachegrowth}}$ byte, mỗi kỷ nguyên. Để tránh sự đều đặn dẫn đến hành vi tuần hoàn, kích thước phải là một số nguyên tố. Do đó, kích thước được giảm bớt một bội số của J_{mixbytes}, đối với bộ dữ liệu, và $J_{\text{hashbytes}}$ đối với cache. Đặt $d_{\text{size}} = \|\mathbf{d}\|$ là kích thước của bộ dữ liệu. Được tính bằng công thức

$$d_{\text{size}} = E_{\text{prime}}(J_{\text{datasetinit}} + J_{\text{datasetgrowth}} \cdot E_{\text{epoch}} - J_{\text{mixbytes}}, J_{\text{mixbytes}}) \tag{325}$$

Kích thước của cache, c_{size}, được tính bằng công thức

$$c_{\text{size}} = E_{\text{prime}}(J_{\text{cacheinit}} + J_{\text{cachegrowth}} \cdot E_{\text{epoch}} - J_{\text{hashbytes}}, J_{\text{hashbytes}}) \tag{326}$$

$$E_{\text{prime}}(x, y) = \begin{cases} x & \text{nếu} \quad x/y \in \mathbb{N} \\ E_{\text{prime}}(x - 2 \cdot y, y) & \text{ngược lại} \end{cases} \tag{327}$$

J.4. **Tạo bộ dữ liệu.** Để tạo bộ dữ liệu, chúng ta cần bộ nhớ cache $\mathbf{c}$, là một mảng byte. Nó phụ thuộc vào kích thước cache c_{size} và hash gốc $\mathbf{s} \in \mathbb{B}_{32}$.

J.4.1. *Hash gốc.* Hash gốc khác nhau cho mỗi kỷ nguyên. Đối với kỷ nguyên đầu tiên, nó là hash Keccak-256 của một chuỗi 32 byte chứa toàn số 0. Đối với mọi kỷ nguyên khác, nó luôn là hash Keccak-256 của hash gốc trước đó:

$$\mathbf{s} = C_{\text{seedhash}}(H_{\text{i}}) \tag{328}$$

$$C_{\text{seedhash}}(H_{\text{i}}) = \begin{cases} \mathbf{0}_{32} & \text{nếu} \quad E_{\text{epoch}}(H_{\text{i}}) = 0 \\ \texttt{KEC}(C_{\text{seedhash}}(H_{\text{i}} - J_{\text{epoch}})) & \text{ngược lại} \end{cases} \tag{329}$$

Trong đó, $\mathbf{0}_{32}$ là 32 byte chứa toàn số 0.

J.4.2. *Bộ nhớ cache.* Quá trình sản xuất cache liên quan đến việc sử dụng hash gốc để đầu tiên lấp đầy tuần tự c_{size} byte bộ nhớ, sau đó thực hiện $J_{\text{cacherounds}}$ vòng lặp của thuật toán RandMemoHash được tạo ra bởi Lerner [2014]. Bộ nhớ cache ban đầu $\mathbf{c}'$, là một mảng của mảng các byte đơn, sẽ được xây dựng như sau.

Chúng ta định nghĩa mảng $\mathbf{c}_i$, bao gồm 64 byte đơn, là phần tử thứ i của bộ nhớ cache ban đầu:

$$\mathbf{c}_i = \begin{cases} \texttt{KEC512}(\mathbf{s}) & \text{nếu} \quad i = 0 \\ \texttt{KEC512}(\mathbf{c}_{i-1}) & \text{ngược lại} \end{cases} \tag{330}$$

Do đó, $\mathbf{c}'$ có thể được định nghĩa như sau

$$\mathbf{c}'[i] = \mathbf{c}_i \quad \forall \quad i < n \tag{331}$$

$$n = \left\lfloor \frac{c_{\text{size}}}{J_{\text{hashbytes}}} \right\rfloor \tag{332}$$

Bộ nhớ cache được tính bằng cách thực hiện $J_{\text{cacherounds}}$ vòng lặp của thuật toán RandMemoHash đối với bộ nhớ cache ban đầu $\mathbf{c}'$:

$$\mathbf{c} = E_{\text{cacherounds}}(\mathbf{c}', J_{\text{cacherounds}}) \tag{333}$$

$$E_{\text{cacherounds}}(\mathbf{x}, y) = \begin{cases} \mathbf{x} & \text{nếu} \quad y = 0 \\ E_{\text{RMH}}(\mathbf{x}) & \text{nếu} \quad y = 1 \\ E_{\text{cacherounds}}(E_{\text{RMH}}(\mathbf{x}), y-1) & \text{ngược lại} \end{cases} \tag{334}$$

Trong đó, một vòng lặp đơn sửa đổi mỗi tập hợp của cache như sau:

$$E_{\text{RMH}}(\mathbf{x}) = \big(E_{\text{rmh}}(\mathbf{x}, 0), E_{\text{rmh}}(\mathbf{x}, 1), ..., E_{\text{rmh}}(\mathbf{x}, n-1)\big) \tag{335}$$

$$E_{\text{rmh}}(\mathbf{x}, i) = \texttt{KEC512}(\mathbf{x}'[(i-1+n) \bmod n] \oplus \mathbf{x}'[\mathbf{x}'[i][0] \bmod n]) \quad \text{với} \quad \mathbf{x}' = \mathbf{x} \quad \text{ngoại trừ} \quad \mathbf{x}'[j] = E_{\text{rmh}}(\mathbf{x}, j) \quad \forall \quad j < i \tag{336}$$

J.4.3. *Tính toán toàn bộ dataset.* Đầu tiên, chúng ta kết hợp dữ liệu từ J_{parents} nút cache được chọn một cách giả tưởng, và sau đó thực hiện hash để tính toán dataset. Toàn bộ dataset sau đó được tạo ra bởi một số phần tử, mỗi phần tử có kích thước $J_{\text{hashbytes}}$ byte:

$$\mathbf{d}[i] = E_{\text{datasetitem}}(\mathbf{c}, i) \quad \forall \quad i < \left\lfloor \frac{d_{\text{size}}}{J_{\text{hashbytes}}} \right\rfloor \tag{337}$$

Để tính toán một phần tử đơn, chúng ta sử dụng một thuật toán được lấy cảm hứng từ FNV hash (Glenn Fowler [1991]) trong một số trường hợp như một phương thức thay thế không kết hợp cho XOR.

$$E_{\text{FNV}}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot (0\text{x}01000193 \oplus \mathbf{y})) \bmod 2^{32} \tag{338}$$

Bây giờ có thể tính toán phần tử đơn của dataset như sau:

$$E_{\text{datasetitem}}(\mathbf{c}, i) = E_{\text{parents}}(\mathbf{c}, i, -1, \varnothing) \tag{339}$$

$$E_{\text{parents}}(\mathbf{c}, i, p, \mathbf{m}) = \begin{cases} E_{\text{parents}}(\mathbf{c}, i, p+1, E_{\text{mix}}(\mathbf{m}, \mathbf{c}, i, p+1)) & \text{nếu} \quad p < J_{\text{parents}} - 2 \\ E_{\text{mix}}(\mathbf{m}, \mathbf{c}, i, p+1) & \text{ngược lại} \end{cases} \tag{340}$$

$$E_{\text{mix}}(\mathbf{m}, \mathbf{c}, i, p) = \begin{cases} \texttt{KEC512}(\mathbf{c}[i \bmod c_{\text{size}}] \oplus i) & \text{nếu} \quad p = 0 \\ E_{\text{FNV}}\big(\mathbf{m}, \mathbf{c}[E_{\text{FNV}}(i \oplus p, \mathbf{m}[p \bmod \lfloor J_{\text{hashbytes}}/J_{\text{wordbytes}} \rfloor]) \bmod c_{\text{size}}]\big) & \text{ngược lại} \end{cases} \tag{341}$$

J.5. **Hàm bằng chứng công việc (Proof-of-work).** Cơ bản, chúng ta duy trì một "mix" có độ rộng là J_{mixbytes} byte và lặp đi lặp lại việc lấy tuần tự J_{mixbytes} byte từ toàn bộ dataset và sử dụng hàm E_{FNV} để kết hợp nó với mix. Sử dụng J_{mixbytes} byte của quy trình truy cập tuần tự để mỗi vòng của thuật toán luôn luôn lấy một trang đầy đủ từ RAM, giảm thiểu việc thiếu những thông tin cần thiết trong bộ đệm tra cứu dịch, mà lý thuyết có thể bị ASIC tránh được.

Nếu kết quả của thuật toán này nằm dưới mục tiêu mong muốn, thì nonce là hợp lệ. Lưu ý rằng việc áp dụng thêm KEC ở cuối đảm bảo rằng tồn tại một nonce trung gian có thể được cung cấp để chứng minh rằng ít nhất một lượng công việc nhỏ đã được thực hiện; xác minh PoW nhanh chóng này có thể được sử dụng cho mục đích chống lại DDoS. Nó cũng phục vụ để cung cấp bảo đảm thống kê rằng kết quả là một số 256 bit không chệch lệ.

Hàm PoW trả về một mảng với mix được nén làm mục đầu tiên và băm Keccak-256 của việc nối mix được nén với seed hash làm mục thứ hai:

$$\texttt{PoW}(H_{\not H}, H_{\text{n}}, \mathbf{d}) = \{\mathbf{m}_{\text{c}}(\texttt{KEC}(\texttt{RLP}(L_{\text{H}}(H_{\not H}))), H_{\text{n}}, \mathbf{d}), \texttt{KEC}(\mathbf{s}_{\text{h}}(\texttt{KEC}(\texttt{RLP}(L_{\text{H}}(H_{\not H}))), H_{\text{n}}) + \mathbf{m}_{\text{c}}(\texttt{KEC}(\texttt{RLP}(L_{\text{H}}(H_{\not H}))), H_{\text{n}}, \mathbf{d}))\} \tag{342}$$

Với $H_{\not H}$ là hash của header mà không có nonce. Mix được nén $\mathbf{m}_{\text{c}}$ được lấy như sau:

$$\mathbf{m}_{\text{c}}(\mathbf{h}, \mathbf{n}, \mathbf{d}) = E_{\text{compress}}(E_{\text{accesses}}(\mathbf{d}, \sum_{i=0}^{n_{\text{mix}}} \mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}), \mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}), -1), -4) \tag{343}$$

The seed hash being:

$$\mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}) = \texttt{KEC512}(\mathbf{h} + E_{\text{revert}}(\mathbf{n})) \tag{344}$$

$E_{\text{revert}}(\mathbf{n})$ Trả về chuỗi byte hoàn nguyên của nonce $\mathbf{n}$:

$$E_{\text{revert}}(\mathbf{n})[i] = \mathbf{n}[\|\mathbf{n}\| - i] \tag{345}$$

Chúng ta lưu ý rằng toán tử "+" giữa hai chuỗi byte dẫn đến sự nối của cả hai chuỗi.

Bộ dữ liệu $\mathbf{d}$ thu được như được mô tả trong phần J.4.3.

Số lượng các chuỗi được sao chép trong hỗn hợp là:

$$n_{\text{mix}} = \left\lfloor \frac{J_{\text{mixbytes}}}{J_{\text{hashbytes}}} \right\rfloor \tag{346}$$

Để thêm các nút dữ liệu ngẫu nhiên vào hỗn hợp, hàm E_{accesses} được sử dụng:

$$E_{\text{accesses}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) = \begin{cases} E_{\text{mixdataset}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) & \text{nếu} \quad i = J_{\text{accesses}} - 2 \\ E_{\text{accesses}}(E_{\text{mixdataset}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i), \mathbf{s}, i+1) & \text{ngược lại} \end{cases} \tag{347}$$

$$E_{\text{mixdataset}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) = E_{\text{FNV}}(\mathbf{m}, E_{\text{newdata}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i)) \tag{348}$$

E_{newdata} trả về một mảng với n_{mix} phần tử:

$$E_{\text{newdata}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i)[j] = \mathbf{d}[E_{\text{FNV}}(i \oplus \mathbf{s}[0], \mathbf{m}[i \bmod \left\lfloor \frac{J_{\text{mixbytes}}}{J_{\text{wordbytes}}} \right\rfloor]) \bmod \left\lfloor \frac{d_{\text{size}}/J_{\text{hashbytes}}}{n_{\text{mix}}} \right\rfloor \cdot n_{\text{mix}} + j] \quad \forall \quad j < n_{\text{mix}} \tag{349}$$

Hỗn hợp được nén như sau:

$$E_{\text{compress}}(\mathbf{m}, i) = \begin{cases} \mathbf{m} & \text{nếu} \quad i \geqslant \|\mathbf{m}\| - 8 \\ E_{\text{compress}}(E_{\text{FNV}}(E_{\text{FNV}}(E_{\text{FNV}}(\mathbf{m}[i+4], \mathbf{m}[i+5]), \mathbf{m}[i+6]), \mathbf{m}[i+7]), i+8) & \text{ngược lại} \end{cases} \tag{350}$$

Appendix K. Sự bất thường trên mạng chính

K.1. **Xóa tài khoản mặc dù Out-of-gas.** Tại khối 2675119, trong giao dịch 0xcf416c536ec1a19ed1fb89e4ec7ffb3cf73aa413b3aa9b77d60e4fd81a4296ba, một tài khoản tại địa chỉ 0x03 đã được gọi và một ngoại lệ hết gas (out-of-gas) đã xảy ra trong cuộc gọi. Chống lại phương trình (199), điều này đã thêm 0x03 vào tập hợp các địa chỉ được chạm và giao dịch này đã biến $\boldsymbol{\sigma}[0x03]$ thành $\varnothing$.

Appendix L. Danh sách các biểu tượng toán học

Biểu tượng	Lệnh latex	Sự miêu tả
$\bigvee$	`\bigvee`	Đây là giới hạn trên, tối cao hoặc tham gia của tất cả các yếu tố được vận hành.Do đó, nó là yếu tố lớn nhất của các yếu tố như vậy (Davey and Priestley [2002]).