Viết các hợp đồng thông minh cho công nghệ sổ cái phân tán

Pablo Lamela Seijas, Simon Thompson
{pl240, SJThompson} @ kent.ac.uk
Đại học Kent, Vương quốc Anh

Darryl McAdams
darryl.mcadams@iohk.io
IOHK

Ngày 10 tháng 2 năm 2017

Tóm lược

Chúng tôi cung cấp tổng quan về các ngôn ngữ tập lệnh được sử dụng trong các Crypto hiện nay. Đặc biệt là chúng tôi xem xét một số chi tiết về các ngôn ngữ tập lệnh của Bitcoin, Nxt và Ethereum, trong ngữ cảnh tổng quan cấp độ cao về Công nghệ sổ cái phân tán và Crypto. Chúng tôi khảo sát các cách tiếp cận khác nhau và đưa ra một cái nhìn tổng quan về những lời bình luận đối với các ngôn ngữ hiện nay. Chúng tôi cũng đề cập đến các công nghệ có thể được sử dụng để củng cố các phần mở rộng và đổi mới trong tập lệnh và hợp đồng, bao gồm các công nghệ để xác minh, chẳng hạn như bằng chứng không kiến thức (Zero Knowledge Proof), mã Code mang theo bằng chứng (Proof-Carrying Code) và phân tích tĩnh, cũng như các cách tiếp cận để làm cho hệ thống hiệu quả hơn, ví dụ: Cây cú pháp trừu tượng Merkelized (MAST - Merkelized Abstract Syntax Tree).

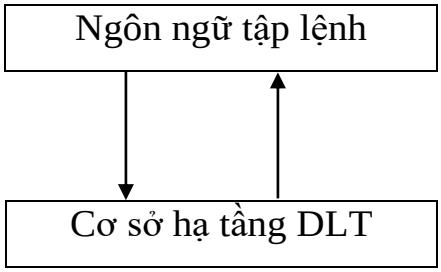
1. Giới thiệu

Công nghệ sổ cái phân tán (DLT - Distributed Ledger Technology) gần đây đã thu hút sự chú ý từ các cá nhân và công ty khởi nghiệp, cũng như các tổ chức thương mại, các tổ chức quốc gia và quốc tế [18]. DLT cung cấp khả năng ghi lại thông tin một cách an toàn theo cách phân tán và phi tập trung. Dựa trên điều này, có thể xây dựng không chỉ cơ sở dữ liệu phân tán mà còn ghi lại kết quả của các giao dịch có giá trị tài chính, đặc biệt là Crypto. Nhiều loại ngôn ngữ tập lệnh khác nhau đã được giới thiệu cho phép người dùng viết các hợp đồng thông minh mô tả các giao dịch này. Trong nghiên cứu này, chúng tôi giới thiệu tổng quan về các ngôn ngữ tập lệnh được sử dụng trong các loại Crypto hiện nay. Đặc biệt là chúng tôi xem xét các ngôn ngữ tập lệnh của Bitcoin, Nxt và Ethereum, trong ngữ cảnh tổng quan cấp độ cao

về Công nghệ sổ cái phân tán và Crypto. Chúng tôi cũng đề cập đến các công nghệ có thể được sử dụng để củng cố các phần mở rộng và đổi mới trong tập lệnh và hợp đồng, bao gồm các công nghệ để xác minh, chẳng hạn như bằng chứng không kiến thức, mã Code mang bằng chứng và phân tích tĩnh, và các phương pháp tiếp cận để làm cho hệ thống hiệu quả hơn, ví dụ như Cây cú pháp trừu tượng Merkelized. [7] Khi khảo sát một mặt cắt ngang của công việc trong lĩnh vực này, điều đáng chú ý là nhiều cách tiếp cận khác nhau để viết tập lệnh.

- Một số ngôn ngữ tập lệnh là Turing-Incomplete: ví dụ như tập lệnh Bitcoin không có cơ sở để lặp lại hoặc đệ quy; trong khi những thứ khác như tập lệnh Ethereum, là Turing-Complete, ít nhất về nguyên tắc, hỗ trợ các cấu trúc lặp. Mặt khác, chúng mất đi tính hoàn chỉnh trong thực tế vì giới hạn thời gian chạy được đặt trên thời gian thực hiện có thể, kích thước hệ thống, v.v.
- Bitcoin và Ethereum cung cấp mô hình tính toán cấp thấp bằng máy ảo, trong đó có thể biên dịch ngôn ngữ cấp cao hơn, chẳng hạn như Solidity của Ethereum; các hệ thống khác, bao gồm cả Nxt, cung cấp API cấp cao hơn bằng ngôn ngữ dành cho mục đích chung (ở đây là JavaScript).
- Một cách tiếp cận khác coi các giao dịch DLT phù hợp với một mô hình quy trình kinh doanh rộng hơn và sẽ đề xuất tạo ra chúng bằng cách chuyển đổi một ngôn ngữ như BPMN hoặc các máy trạng thái (State Machine) hữu hạn.
- Cuối cùng, có thể sử dụng công nghệ ảo hóa để tách biệt các tính toán, như được thực hiện trong Hyperledger với vùng chứa Docker.

Mặc dù chúng tôi đã nói về các ngôn ngữ tập lệnh như thể chúng là ngôn ngữ lập trình cho mục đích chung, nhưng chúng cũng có thể vượt xa điều đó trong việc cung cấp các tiện ích cụ thể bao gồm tính ngẫu nhiên, đăng ký tên, ẩn danh, liên kết khuyến khích, kiểm soát giao dịch, v.v. Do đó, chúng tôi thấy các tương tác giữa ngôn ngữ và cơ sở hạ tầng DLT chung mà nó được xây dựng:



Rõ ràng, cơ sở hạ tầng DLT định hình khá nhiều ngôn ngữ tập lệnh, nhưng cũng có thể có những tương tác theo hướng ngược lại. Đặc biệt, việc khám phá cách thức cung cấp tính ngẫu nhiên trong Ethereum, cho thấy rằng việc sử dụng dấu thời gian điển hình như một nguồn ngẫu nhiên có thể cung cấp một cơ chế tấn công tiềm năng. Chúng tôi đưa ra một cái nhìn tổng quan về điều này và các lỗ hổng bảo mật tiềm ẩn khác ở cuối bài nghiên cứu.

Để cung cấp một số ngữ cảnh, chúng tôi cung cấp các giới thiệu ngắn gọn và tài liệu tham khảo với thông tin về các loại Crypto khác nhau và các hệ thống phi tập trung, nhưng mục đích chính của công việc này là tổng quan về ngôn ngữ tập lệnh và các công nghệ liên quan.

Chúng tôi bắt đầu bằng cách giới thiệu về tập lệnh Bitcoin trong ngữ cảnh trong Phần 2, làm tương tự đối với Ethereum và Nxt trong Phần 3 và 4. Các hệ thống DLT khác được khảo sát trong Phần 5 và các cách tiếp cận khác đối với tập lệnh trong Phần 6. Phần 7 trình bày một số bình luận về các hệ thống hiện nay. Phần 8 trình bày một số công nghệ có thể đóng góp vào thế hệ tiếp theo của tập lệnh Blockchain. Phần 9 kết thúc bài nghiên cứu.

2. Bitcoin

Bitcoin là cách triển khai đầu tiên được sử dụng rộng rãi của một loại Crypto phi tập trung và giới thiệu cơ chế đồng thuận phi tập trung của một *Blockchain*.

Blockchain của Bitcoin là một danh sách các Block giao dịch được liên kết lại [2]. Mỗi Block trong Blockchain được xác định bằng một Hash, được tạo bằng cách sử dụng thuật toán mã hoá Hash SHA256 trên tiêu đề của Block. Mỗi Block cũng tham chiếu đến Block trước đó, được gọi là Block mẹ, thông qua trường "Hash Block trước đó" trong tiêu đề Block. Nói cách khác, mỗi Block chứa Hash của Block trước đó bên trong tiêu đề của chính nó. Trình tự Hash liên kết mỗi Block với Block trước đó tạo ra một chuỗi quay ngược trở lại toàn bộ Block đầu tiên từng được tạo, được gọi là Block Genesis.

Về bản chất, nó có thể được coi như một nhật ký công khai. Trong trường hợp của Bitcoin, nó được sử dụng để theo dõi các giao dịch. Giả sử các công cụ khai thác đại diện cho hơn 50% sức mạnh tính toán góp phần tạo ra các Block mới cho

Blockchain là "trung thực" và các Hash được sử dụng là không thể thay đổi và không thể đoán trước, thông tin trong Blockchain trở nên khó thu hồi về mặt thống kê hơn khi các Block mới được thêm vào.

Thay đổi một Block có nghĩa là tạo lại tất cả các Block trên nó, vì mọi Block (ngoại trừ Block Genesis) đều chứa Hash của Block trước đó. Việc tạo một Block rất tốn kém vì nó yêu cầu bằng chứng công việc, điều này chỉ cho phép các thợ đào tạo ra các Block tương ứng với sức mạnh tính toán của họ (so với các công cụ khai thác còn lại); và chỉ chuỗi dài nhất được coi là hợp lệ, vì vậy việc một tác nhân duy nhất muốn thay thế chuỗi chính là điều không thể, miễn là không có tác nhân duy nhất nào kiểm soát hơn 50% tổng sức mạnh tính toán.

2.1. Bằng chứng công việc (PoW)

PoW là một cơ chế được sử dụng để đảm bảo việc đạt được sự đồng thuận phân tán mà không cần sự hiện diện của cơ quan trung ương hoặc yêu cầu một nhóm người dùng tham gia được xác định theo một cách nào đó.

PoW được thực hiện như một yêu cầu đối với mỗi Block trong Blockchain phải có một Hash nhỏ hơn một số mục tiêu nhất định. Con số mục tiêu này được xác định thông qua một phép tính được thực hiện sau mỗi 2016 Block (khoảng 2 tuần) và nhằm mục đích đảm bảo rằng, không phụ thuộc vào lượng sức mạnh tính toán toàn cầu, trung bình một Block mới sẽ được tạo ra khoảng 10 phút một lần.

2.2. Sự đảm bảo

Thiết kế của hệ thống Bitcoin có nghĩa là nó có thể cung cấp một loạt các đảm bảo, theo một tập hợp các giả định được trình bày rõ ràng. Trong phần này, chúng tôi giải thích các đảm bảo, cùng với các cơ chế mã hoá được sử dụng để hỗ trợ chúng.

Thuật toán được sử dụng để Hash các Block trong Blockchain là SHA256 kép, bao gồm việc áp dụng SHA256 hai lần vào tiêu đề của Block. Vì hàm Hash được coi là không thể thay đổi theo các giả định hợp lý, nên cách hiệu quả nhất được biết đến để có được một Hash nhỏ hơn mục tiêu là Brute Force, tức là bằng cách thử liên tục các đầu vào khác nhau cho đến khi tìm thấy một đầu vào phù hợp. Để thay đổi Hash kết quả, đầu vào của hàm Hash sẽ được sửa đổi bằng cách thay đổi trường dành riêng được gọi là *Nonce* (hoặc các phần khác của tiêu đề). Người dùng thực

hiện công việc tạo Block mới được gọi là *thợ đào (Miner)*.

2.2.1. Thỏa thuận phi tập trung

Trong trường hợp của Bitcoin, thỏa thuận về việc tạo ra các Block đạt được nhờ vào độ khó của việc tạo Block: sức mạnh tính toán cần thiết để "chiếm quyền điều khiển" việc tạo ra một sự đồng thuận mới thay thế, rất khó đạt được bởi một thợ đào đơn lẻ hoặc Các-ten của thợ đào.

2.2.2. Khả năng xác minh

Một đảm bảo khác được cung cấp bởi Bitcoin là khả năng người dùng đơn phương xác minh, mà không tin tưởng bất kỳ ai khác về một Block, và do đó toàn bộ Blockchain là hợp lệ (nó tuân thủ các quy tắc). Thuộc tính thứ hai này có thể thực hiện được vì mọi Block đều bao gồm một Hash của Block trước đó và Block đầu tiên (hoặc Block Genesis) được mã hóa cứng trong mã nguồn của các ứng dụng khách Bitcoin.

2.2.3. Khả năng không thể thu hồi có thể xảy ra

Mặt khác, tính không thể thu hồi của các giao dịch chỉ được đảm bảo cuối cùng bằng giả định rằng ít nhất 50% sức mạnh khai thác Bitcoin đến từ những người dùng trung thực. Những người dùng trung thực sẽ chấp nhận Blockchain dài nhất là Blockchain thật và xây dựng dựa trên nó. Do đó, giả sử phần lớn sức mạnh khai thác là trung thực, ngày càng có nhiều khả năng rằng một đại lý không trung thực sẽ có thể tạo ra một đợt Fork từ một Block cũ để đợt Fork mới được dài hơn chuỗi thật. Nếu không đúng như vậy, thì một người dùng độc hại sẽ có thể hoàn nguyên các giao dịch bằng cách tạo một Fork, nhưng ngay cả trong trường hợp này, vẫn không thể làm cho các Block không hợp lệ được chấp nhận bởi những người dùng trung thực nhờ vào khả năng xác minh.

Bởi vì độ khó của việc tạo Block là thay đổi, khi chúng ta nói về chuỗi hoặc Blockchain dài nhất, chúng ta thực sự muốn nói đến chuỗi có độ khó kết hợp cao nhất, không phải là chuỗi có số Block nhiều nhất. Nếu không, người dùng độc hại có thể tạo ra một chuỗi riêng biệt mà chỉ anh ta mới có thể đóng góp và do đó, độ khó của chuỗi này sẽ thấp hơn (vì nó thích ứng với tổng sức mạnh xử lý của các thợ đào trong chuỗi). Bằng cách có một chuỗi có độ khó thấp hơn, kẻ tấn công sẽ có thể tạo các Block rất nhanh và có khả năng có được một chuỗi có nhiều Block

hơn so với chuỗi được chấp nhận trên toàn cầu.

2.3. Khai thác

Có hai ưu đãi dành cho thợ đào. Mặt khác, mỗi Block cung cấp một lượng Bitcoin mới cho thợ đào đã tạo ra nó. Mặt khác, mỗi giao dịch thường chứa một khoản phí giao dịch (được ngầm xác định là Bitcoin chưa được sử dụng trong giao dịch), thuế của tất cả các giao dịch trong Block cũng sẽ được chuyển cho thợ đào. Trong trường hợp của Bitcoin, phần thưởng cho việc tạo Block ngoài phí giao dịch, bị giảm một nửa sau mỗi 210.000 Block (khoảng bốn năm một lần), ban đầu là 50 Bitcoin và tại thời điểm viết bài là 12,5 Bitcoin.

2.4. Cây Merkle

Để có thể chứng minh rằng một giao dịch tồn tại trong Blockchain mà không cần chuyển toàn bộ Blockchain, các giao dịch được lưu trữ trong một cấu trúc được gọi là *cây Merkle*. Một cây như vậy lưu trữ các giao dịch tại các Node lá và các Node bên trong chứa các Hash kết hợp của các cây con ngay lập tức của chúng. Điều này có thể chứng minh rằng một giao dịch đã được Hash mà không cần phải Hash lại tất cả các giao dịch của một Block, chỉ chuỗi Hash được gọi là đường dẫn Merkle, thu được bằng cách đi ngang cây từ gốc đến Node lá có liên quan.

2.5. Giao dịch

Block trong Blockchain thu thập các giao dịch. Các giao dịch này chuyển Bitcoin hoặc *đầu ra giao dịch chưa chi tiêu* (UTXO) giữa những người dùng. Nhưng chủ sở hữu hiệu quả của một lượng Bitcoin cụ thể không được xác định trong giao dịch; đúng hơn, giao dịch khai báo, đối với mỗi đầu ra chưa chi tiêu, một chương trình được viết bằng ngôn ngữ Script (xem Phần 2.6); ai muốn chi tiêu đầu ra đó phải cung cấp đầu vào cho chương trình làm cho nó thành công (nghĩa là trả về số 0).
Thông thường, chương trình này cung cấp một thử thách mật mã mà chỉ chủ sở hữu mới có thể giải quyết, chẳng hạn như cung cấp chữ ký được tạo bằng khóa riêng của chủ sở hữu, nhưng nó có thể là bất kỳ chương trình nào. Ví dụ: một chương trình có thể kiểm tra xem một số người đã ký một giao dịch, một tập hợp con của một nhóm người đã ký một giao dịch hoặc thậm chí rằng một giao dịch cung cấp lời giải cho một câu đố.

2.6. Script

Ngôn ngữ lập trình được sử dụng để tạo các tập lệnh trong Bitcoin được gọi là "Script". Script là một ngôn ngữ dựa trên ngăn xếp Bytecode giống ngôn ngữ lập trình Forth nhưng, không giống như Forth, Script được thiết kế có chủ đích để việc thực thi của nó được đảm bảo hoàn thành. Các tập lệnh bao gồm một chuỗi các lệnh và chúng được thực thi một cách tuyến tính, không có bước nhảy ngược lại; do đó, thời gian thực thi được giới hạn ở trên bởi độ dài của tập lệnh sau con trỏ lệnh. Hạn chế này ngăn chặn các cuộc tấn công từ chối dịch vụ vào các Node xác thực các Block.

Ngôn ngữ Script cung cấp một tập hợp hơn một trăm nguyên thủy cho phép [46]:

- Việc thêm các hằng số vào ngăn xếp.
- Một số điều khiển luồng có điều kiện cơ bản, đó là không lặp lại, nhưng lười biếng (không yêu cầu đánh giá cả hai lựa chọn thay thế).
- Thao tác ngăn xếp (bao gồm cả truy cập ngăn xếp thay thế cơ bản).
- Thao tác chuỗi (hầu hết bị vô hiệu hóa trong quá trình triển khai máy khách tiêu chuẩn[17]).
- Thao tác Bitwise (hầu hết bị vô hiệu hóa).
- Một số cơ bản 32-Bit số học với sự tràn ra (nhân và chia bị vô hiệu hóa).
- Một số mật mã nguyên thủy cơ bản để Hash và xác minh chữ ký.
- Hai nguyên tắc cơ bản để trì hoãn và hết hạn các giao dịch chưa cam kết, liên quan đến theo thời gian hoặc chiều dài hiện tại của Blockchain.

Một số đặc điểm có sẵn trong Forth nhưng không có trong Script bao gồm biến, mảng, hàm và vòng lặp.

Thông thường, các hệ thống dựa trên ngăn xếp chỉ cho phép truy cập vào một số mục hàng đầu trong ngăn xếp, điều này buộc các chương trình phải tính toán lại các giá trị có thể đã được đánh giá.

Trong Forth, có thể khai báo các biến và mảng, vì vậy chúng cho phép truy cập ngẫu nhiên, nhưng các chức năng này không có sẵn trong Script. Tuy nhiên, Script cung cấp một cặp hướng dẫn cho phép truy cập đọc ngẫu nhiên vào ngăn xếp:

- OP_PICK: sao chép một mục từ độ sâu tùy ý trên ngăn xếp lên trên cùng.

- OP_ROLL: di chuyển một mục từ độ sâu tùy ý trên ngăn xếp lên trên cùng.

Các nguyên thủy này cho phép truy cập đọc ngẫu nhiên vào các giá trị đã được tính toán trước đó trong tập lệnh, nhưng ngôn ngữ Script không cung cấp bất kỳ nguyên thủy nào cho phép truy cập ghi ngẫu nhiên vào một trong các ngăn xếp.

Bằng cách sử dụng các hướng dẫn truy cập ngẫu nhiên này và ngăn xếp thay thế, có thể viết một tập lệnh trong ngôn ngữ Script mô phỏng hiệu quả một máy Turing cụ thể thực hiện một số bước hữu hạn bị giới hạn bởi độ dài của tập lệnh. Tuy nhiên, có vẻ như không thể mô phỏng hiệu quả một máy RASP và mô phỏng này không có nghĩa ngôn ngữ là Turing Complete: để thực hiện một chương trình lặp, cần phải giải nén số lần nó lặp lại. Điều này chỉ có thể được thực hiện cho một đầu vào cụ thể, thay vì một cấu hình đầu vào tùy ý trên băng.

2.6.1. Hạn chế

Trong thực tế, nhiều hạn chế đã được đặt ra để tránh việc xây dựng các tập lệnh khác với những tập lệnh được coi là tiêu chuẩn, một số hạn chế là:

- Giới hạn về kích thước của các Block (được đặt thành khoảng 1MB tại thời điểm đang viết bài [6]).
- Giới hạn về dung lượng của các tập lệnh (được đặt thành 512 Byte cho mỗi phần tử và 10.000 Byte cho mỗi tập lệnh tại thời điểm viết bài [29]).
- Giới hạn về số lượng mã Opcode (được đặt thành 201 cho hầu hết các Opcode tại thời điểm viết bài [32]).
- Nhiều Opcode bị vô hiệu hóa [46].

Tuy nhiên, dường như không có bất kỳ giới hạn rõ ràng nào đối với kích thước của ngăn xếp, điều này về mặt hình thức là không cần thiết vì nó bị hạn chế ngầm bởi các giới hạn về độ dài của tập lệnh.

2.6.2. Trả cho Hash của tập lệnh (P2SH)

Có một phần mở rộng cho Bitcoin cho phép thanh toán cho các tập lệnh bằng cách cung cấp Hash của nó làm địa chỉ đích. Điều này giúp người dùng sử dụng các tập lệnh không phải tiêu chuẩn dễ dàng hơn vì họ có thể chuyển tiền cho chúng như một địa chỉ bình thường và tập lệnh có thể được tạo bởi một người dùng khác. Nó có thể được sử dụng để chia các tập lệnh thành một số giao dịch và cho phép các loại hợp đồng phức tạp hơn, nhưng trên thực tế, có một hạn chế là cấm sử dụng đệ

quy P2SH (nó không hoạt động để trỏ một tập lệnh P2SH sang một tập lệnh P2SH khác).

3. Ethereum

Ethereum là một hệ thống tiền mã hoá dựa trên Blockchain nhằm mục đích cung cấp một máy tính đa năng phi tập trung. Tiền tệ cơ bản của nó được gọi là Ether. Các chương trình chạy trên máy tính phi tập trung này thường được gọi là hợp đồng thông minh và được thực thi tự động thông qua quy trình xác thực Blockchain được thực hiện bởi tất cả các Node đầy đủ một cách độc lập.

Các Node đầy đủ là những Node tải xuống và xác thực toàn bộ Blockchain, các Node này không cần tin tưởng bất kỳ Node nào khác, vì chúng có thể xác thực toàn bộ lịch sử giao dịch. Ngược lại, vì kích thước của Blockchain là đáng kể (khoảng 89 GB tại thời điểm viết bài), các thiết bị di động thường sử dụng ứng dụng khách nhẹ (Lightweight Client), chỉ lưu trữ một phần của Blockchain và dựa vào các Node đầy đủ để xác thực giao dịch.

3.1. Cấu trúc chung

Không giống như Bitcoin, cơ chế tập lệnh Ethereum cho phép thực hiện hành vi lặp thông qua việc sử dụng cả bước nhảy và lệnh gọi đệ quy. Nếu đây là điểm khác biệt duy nhất với Bitcoin, thì kẻ tấn công độc hại có thể thực hiện một cuộc tấn công DoS (Từ chối dịch vụ) thành công bằng cách gửi một giao dịch lặp lại mãi mãi, vì các hợp đồng thông minh được xác thực bởi mọi Node. Để tránh vấn đề này, Ethereum cũng đưa ra giới hạn về thời gian thực hiện của mỗi giao dịch được gọi là phí Gas.

3.1.1. Phí Gas

Gas là một lượng Ether được trả trước khi giao dịch được phát hành để trang trải chi phí thực hiện giao dịch. Nếu một giao dịch hết phí Gas trong khi nó đang được thực hiện, giao dịch đó sẽ được khôi phục nhưng lượng phí Gas đã tiêu thụ sẽ không được trả lại.

Vì việc tạo giao dịch yêu cầu người tạo phải chỉ định và phân bổ lượng phí Gas tối đa mà họ sẵn sàng trả, nên các thợ đào có cơ hội phát hiện các giao dịch sẽ mất quá nhiều thời gian để xác thực mà không thực sự phải tính toán kết quả của chúng.

3.1.2. Hợp đồng

Ngoài việc thực hiện tính toán và chuyển Ether, các giao dịch cũng có thể tạo ra các hợp đồng độc lập được lưu trong Blockchain và có thể lưu trữ Ether, dữ liệu, mã Code thực thi, có thể giao tiếp với các hợp đồng khác và tạo các hợp đồng mới lần lượt.

Về cơ bản, các hợp đồng hoạt động với tư cách là người dùng, với sự khác biệt là họ không thể bắt đầu các hành động chuyển đổi (chúng mang tính phản ứng). Giới hạn này được áp dụng để tránh các cuộc tấn công DoS chống lại các hợp đồng hiện có. Với thiết kế này, phí Gas cần thiết để thực hiện mã Code được kích hoạt bởi một giao dịch phải được thanh toán ban đầu bởi người dùng đã phát hành giao dịch ngay từ đầu. Nhưng các hợp đồng có thể được lập trình chọn để hoàn lại tiền cho người dùng hợp pháp, do đó, một cách hiệu quả, có thể có các hợp đồng mà người dùng có thể sử dụng miễn phí.

3.1.3. Blockchain

Cấu trúc và hoạt động của Blockchain trong Ethereum rất giống với cấu trúc của Bitcoin, nhưng có hai điểm khác biệt cơ bản: thông tin trạng thái (State Information) và khuyến khích chú ý (Uncle Incentivization).

Thông tin trạng thái: Trong Bitcoin, tất cả lịch sử giao dịch đều được lưu trữ trong Blockchain và để tìm xem liệu một lượng Bitcoin chưa được sử dụng, có thể cần phải tham khảo các Block nằm sâu trong Blockchain. Do đó, để biết liệu một giao dịch có hợp lệ hay không, cần phải có một bản sao hoàn chỉnh của Blockchain hoặc hỏi ai đó. Trong Ethereum, mọi Block đều chứa một ảnh chụp nhanh với thông tin về tất cả các Ether chưa được sử dụng, các hợp đồng đang hoạt động, v.v. Vì lý do này, có thể cắt bỏ các Node cũ để chỉ lưu trữ tiêu đề của chúng, điều này cho phép tiết kiệm quan trọng và giảm nhu cầu về các Node nhẹ.

Khuyến khích chú ý: Xác thực các Block có khả năng mang lại cho các thợ đào lợi thế cạnh tranh vì họ mất nhiều thời gian hơn để bắt đầu Hash. Điều này kết hợp với việc giảm thời gian tạo Block của Ethereum (hiện tại khoảng 15 giây) có thể khiến một số thợ đào bỏ qua việc xác thực một số hoặc tất cả các giao dịch.

Để giảm bớt vấn đề này, Ethereum tặng một phần thưởng cho các Uncle, đó là các Block không lọt vào Blockchain nhưng vẫn hợp lệ.

3.2. Viết tập lệnh

Mã Code các hợp đồng thông minh của Ethereum được viết bằng Bytecode và được thực thi trong một máy ảo gọi là EVM. EVM có kích thước cố định là 32 Bit và không được định kiểu để đơn giản hóa.

3.2.1. Lưu trữ

Không giống như Bitcoin, EVM của Ethereum cung cấp một ngăn xếp duy nhất giới hạn ở 1024 phần tử, nhưng nó cung cấp hai loại lưu trữ bổ sung:

- Bộ nhớ tạm thời (bộ nhớ), là một mảng Byte và được xóa ở cuối việc thực hiện mỗi giao dịch.
- Lưu trữ vĩnh viễn, là từ một điển khóa giá trị được lập chỉ mục từ và được lưu trữ trên Blockchain giữa các lần thực thi, nhưng có thể được phân bổ một cách rõ ràng.

3.2.2. Hoạt động nhảy (Jump Operation)

Ngôn ngữ EVM của Ethereum cung cấp cả hoạt động nhảy có điều kiện và không điều kiện. Để cho phép triển khai dễ dàng và hiệu quả hơn các trình biên dịch JIT [16], các hoạt động nhảy này chỉ có thể nhắm mục tiêu các phần mã Code được đánh dấu là đích nhảy (Jump Destination).

3.2.3. Hoạt động hợp đồng

Hợp đồng là các thực thể ảo nằm trong Blockchain, có thể lưu trữ Ether và Bytecode, có thể gửi và nhận thông điệp và Ether, cũng như tạo các hợp đồng khác. Không giống như các tài khoản bên ngoài do người dùng quản lý, các hợp đồng không thể bắt đầu giao dịch: chúng có tính phản ứng.

Khởi tạo (Creation): Các hợp đồng mới có thể được tạo trực tiếp bởi người dùng hoặc các hợp đồng khác thông qua hoạt động CREATE Bytecode.

Lệnh gọi (Call): Các hợp đồng có thể gửi thông điệp đến các hợp đồng hoặc tài khoản khác thông qua việc sử dụng các thao tác lệnh gọi. Chúng cho phép gửi Ether, để thực thi mã Code của một hợp đồng khác, quy định lượng phí Gas tối đa cho mã Code được thực hiện bởi lệnh gọi (có thể nhỏ hơn phí Gas có sẵn tại thời điểm lệnh gọi) và có thể chuyển và nhận thông tin.

Các hợp đồng trong Ethereum có một Block Bytecode duy nhất được thực thi bởi các lệnh gọi, nhưng các ngôn ngữ cấp cao như Solidity sẽ tự động xác định một bộ chọn

chức năng ở đầu Block chuyển hướng các lệnh gọi đến phần thích hợp của Bytecode.

Tự xoá (Suicide): Để tiết kiệm không gian lưu trữ, Ethereum cho phép các hợp đồng tự xóa khi không còn cần thiết. Để thúc đẩy điều này, một phần chi phí tạo hợp đồng sẽ được hoàn lại khi hoạt động tự xoá được gọi.

3.2.4. Hoạt động kiểm tra

EVM của Ethereum cho phép các hợp đồng truy cập một số loại thông tin Meta về Blockchain, về bản thân các hợp đồng và thậm chí về mã Code của các hợp đồng khác, có thể được sao chép vào bộ nhớ.

3.2.5. Hoạt động ghi nhật ký

Một chức năng khác được cung cấp bởi EVM là ghi nhật ký. Có một tập hợp các Bytecode cho phép các hợp đồng thông minh ghi lại các giá trị. Các bản ghi này được trả lại dưới dạng "biên lai" là kết quả của việc xử lý giao dịch, nhưng chúng không được lưu trữ rõ ràng trong Blockchain [51].

3.2.6. Ngôn ngữ cấp cao

Ngoài EVM Bytecode, một số ngôn ngữ cấp cao được biên dịch cho nó cũng tồn tại. Những ngôn ngữ phổ biến nhất có lẽ là:

- Solidity: Solidity là một ngôn ngữ cấp cao, hướng hợp đồng có cú pháp tương tự như JavaScript. Solidity được định kiểu tĩnh, hỗ trợ kế thừa, thư viện và các kiểu định nghĩa phức tạp của người dùng cùng với các tính năng khác [48].
- Serpent: theo như tên gọi, Serpent được thiết kế để rất giống với Python; nó được thiết kế để trở nên sạch sẽ và đơn giản tối đa, kết hợp nhiều lợi ích hiệu quả của một ngôn ngữ cấp thấp với tính dễ sử dụng trong phong cách lập trình, đồng thời bổ sung các tính năng đặc biệt dành riêng cho miền cụ thể cho lập trình hợp đồng [47].

4. Nxt

Nxt được thiết kế để trở thành nền tảng chung cho các giao dịch kinh tế dựa trên DLT. Nó được lấy cảm hứng từ sự thành công của Bitcoin, nhưng nhằm mục đích cung cấp nhiều hiệu suất và khả năng mở rộng hơn thông qua việc dựa trên bằng chứng cổ phần thay vì bằng chứng công việc.

NRS (Nxt Reference Software) sử dụng kiến trúc máy khách - máy chủ. Máy chủ NRS là một ứng dụng Java có hai giao diện: một để giao tiếp với các máy chủ khác thông

qua Internet (tạo thành một mạng lưới các Node) và một để phản hồi các yêu cầu từ máy khách thông qua API (Application Program Interface - Giao diện Chương trình Ứng dụng). Thành phần máy khách của NRS là một giao diện thân thiện với người dùng dựa trên trình duyệt với máy chủ NRS (thông qua API), thường được gọi là Ví Nxt.

Như với các hệ thống Blockchain khác, bất kỳ ai cũng có thể cài đặt và chạy phần mềm. Hệ thống là mã nguồn mở và cũng như các dự án khác, các nhà phát triển cốt lõi hoạt động như những người gác cổng (Gatekeeper) cho những thay đổi của hệ thống. Người dùng cá nhân có thể sửa đổi các phiên bản của phần mềm, nhưng cũng như với các hệ thống Blockchain khác, các thay đổi trên tập hợp các phiên bản đã cài đặt với hơn 50% cổ phần sẽ được yêu cầu để gắn với tính toàn vẹn của toàn bộ hệ thống.

Có thể viết các Plugin JavaScript cho máy khách, nhưng chúng không mở rộng chức năng cốt lõi ở tất cả, và tạo điều kiện thuận lợi, ví dụ như trực quan hóa hoặc khám phá Block.

Có những đề xuất cho người kế nhiệm Nxt, Ardor, và điều này dự kiến sẽ được phát hành vào nửa đầu năm 2017. Điều này sẽ tách biệt khái niệm về Token cho "Forging" và cho (người dùng) "giao dịch"; điều này nhằm hỗ trợ hoạt động hiệu quả hơn của Blockchain. Tình hình hiện tại với một Blockchain duy nhất sẽ trở thành một trong đó có cả một Blockchain "cơ sở hạ tầng" duy nhất chăm sóc việc Forging Token cũng như tính nhất quán và tiến độ tổng thể của chuỗi, cùng với nhiều Childchain, có thể thực hiện các hoạt động trong tiền tệ của riêng chúng.

4.1. Khả năng lập trình

Điều quan trọng, khả năng lập trình của hệ thống được cung cấp thông qua một API cấp cao "Fat", có thể truy cập được từ các ứng dụng khách Nxt thông qua giao diện REST. API cung cấp chức năng hỗ trợ nhiều loại giao dịch khác nhau và các giao dịch được phân loại thành các kiểu (Type) và kiểu phụ (Subtype) "để tăng trưởng và phát triển theo mô-đun của giao thức Nxt". Mỗi loại chỉ ra các tham số bắt buộc và tùy chọn, cũng như "phương pháp xử lý" của hoạt động.

Sách trắng (Whitepaper) nói rõ ràng rằng "phần mềm cốt lõi không hỗ trợ bất kỳ dạng ngôn ngữ tập lệnh nào"; thay vào đó, người dùng dự kiến sẽ làm việc với các loại giao dịch và giao dịch được tích hợp sẵn hỗ trợ khoảng 250 hoạt động sơ khai trong một số

lĩnh vực, bao gồm cả các khoản thanh toán cơ bản; một hệ thống bí danh (cho các chuỗi có thể được lưu trữ trên Blockchain, đại diện cho các URI); thông điệp (thông điệp có thể được gửi giữa các tài khoản, nhưng cũng thể hiện dữ liệu có cấu trúc bằng các đối tượng JSON); trao đổi tài sản; mua và bán thông qua cửa hàng hàng hóa kỹ thuật số; cơ sở hạ tầng; các hoạt động theo giai đoạn, v.v..[1] Khả năng tập lệnh của API được cung cấp bởi <script> ở phía máy khách, nhưng điều này không cho phép bất kỳ quyền truy cập nào vào bất kỳ cấp độ cơ bản nào của việc triển khai.[2]

Tóm lại, mặc dù không có chức năng tập lệnh nào trong cốt lõi, có thể xây dựng chức năng bằng cách kết hợp các lệnh gọi API trong JavaScript. Điều này sẽ tạo ra một chuỗi các giao dịch trong API. Ngược lại với các hệ thống có VM cấp thấp hơn, một cuộc tấn công sẽ phải thực hiện bằng các hàm API này, chứ không phải là một chương trình mã Code VM chung.

Do đó, tính bảo mật của hệ thống phụ thuộc vào việc triển khai (mã nguồn mở) của API này: mỗi hoạt động có an toàn không và không có cách nào kết hợp một cuộc tấn công thông qua một chuỗi lệnh gọi API? Mặc dù VM cho phép phạm vi rộng hơn các tập lệnh tiềm năng, nhưng có một chi phí khái niệm trong việc hiểu phạm vi các khả năng được trình bày bởi một API cấp cao lớn và phức tạp hơn.

5. Hệ thống DLT khác

Kể từ khi Bitcoin phát triển, nhiều loại Crypto hoặc Altcoin khác nhau đã được tạo ra. Chúng có thể được phân loại theo mối quan hệ với Bitcoin [50]:

- Chúng có thể được triển khai trong Bitcoin và được gọi là *Meta-Coins*.
- Chúng có thể có Blockchain của riêng mình nhưng được liên kết với Blockchain của Bitcoin (hoặc Crypto khác). Chúng bao gồm *Sidechain* và Blockchain dựa trên khai thác hợp nhất.
- Chúng có thể được lấy cảm hứng từ Bitcoin nhưng được triển khai theo hướng hoàn toàn độc lập.

[1]API được ghi lại đầy đủ tại đây: http://nxtwiki.org/wiki/The_Nxt_API

[2]Một số chi tiết về mã hóa được tìm thấy ở đây: https://nxt.org/category/coding/, cung cấp một số ví dụ về <script>s.

5.1. Meta-Coins

Meta-Coins là cơ chế được thực hiện trên Bitcoin và có thể được coi là những cách cụ thể để sử dụng và diễn giải Bitcoin. Chúng được hưởng lợi từ sự ổn định từ Blockchain Bitcoin, điều này đảm bảo tính không thể hủy ngang của chúng. Nói chung chúng sẽ có các quy tắc bổ sung giải thích ý nghĩa của các giao dịch Bitcoin và có thể bỏ qua những quy tắc đó, điều đó không có ý nghĩa (giao dịch không hợp lệ theo quan điểm của Meta-Coin), nhưng chúng minh bạch đối với những người dùng Bitcoin khác. Chúng có thể lưu trữ thông tin như một phần của các giao dịch Bitcoin (điều này có thể được thực hiện theo một số cách [14]) hoặc chúng có thể chỉ lưu trữ một Hash của dữ liệu và lưu trữ dữ liệu thực tế ở một nơi khác. Trong phần này, chúng tôi xem xét ba ví dụ về Meta-Coin.

5.1.1. Coloured Coins

Coloured Coins [9] xác định thêm ý nghĩa cho Bitcoin, mặc dù ý tưởng này có thể được áp dụng cho các loại Crypto khác và thậm chí cho các loại tiền tệ vật chất. Ví dụ: chúng có thể được sử dụng như một phương tiện để nhận UTXO Bitcoin đại diện cho quyền sở hữu một vật phẩm trong thế giới thực (ví dụ: một mảnh đất, một phần của doanh nghiệp hoặc một vé xem một buổi hòa nhạc). Tất nhiên, Coloured Coins tự nó không thực thi “ý nghĩa bổ sung” này, mà đòi hỏi một tổ chức hoặc công ước có thẩm quyền nào đó phải công nhận chúng.

5.1.2. Type-Coin

Type-Coin [13] là một cơ chế cho cam kết liên kết có mục đích chung. Nó liên quan đến Coloured Coins (xem Phần 5.1.1) ở chỗ nó có thể được sử dụng để đại diện cho các tài nguyên liên kết tương tự như cách Coloured Coins đại diện cho tài sản. Nhưng biểu đạt hơn vì nó có thể biểu thị các vị từ bằng cách sử dụng Logic Affine. Type-Coin được thực hiện trên Bitcoin và trạng thái của nó được giải thích bằng cách áp dụng một bộ quy tắc cho Blockchain của Bitcoin.

5.1.3. Counterparty

Counterparty [11, 12] cung cấp chức năng của Ethereum như một Meta-Coin. Đơn vị tiền tệ nội bộ (XCP) được phân bổ thông qua "bằng chứng đốt" (Proof of Burn) của Bitcoin, nghĩa là để có được các đơn vị XCP, người dùng gửi Bitcoin đến một địa chỉ đặc biệt không thể sử dụng được. Các giao dịch được mã hóa như các giao dịch Bitcoin

thông thường và chúng được xác thực bằng cách sử dụng các quy tắc được chỉ định trong mã nguồn của khách hàng của Counterparty.

5.2. Sidechain và khai thác hợp nhất

Một số hệ thống có Blockchain riêng nhưng nó được liên kết với Bitcoin thông qua các kỹ thuật như 2way-pegging hoặc khai thác hợp nhất. Những kỹ thuật này nhằm mục đích nâng cao Bitcoin bằng cách cung cấp các chức năng mới trong khi sử dụng lại một số thuộc tính mạnh mẽ của nó:

2way-pegging cho phép chuyển đổi trực tiếp Bitcoin sang một loại tiền tệ thay thế và quay trở lại, bằng cách tạm thời khóa tiền thông qua một tập lệnh. Điều này hoạt động tương tự như "bằng chứng đốt" nhưng nó có thể được đảo ngược và do đó, cho phép trao đổi tiền tệ theo cả hai hướng. Cơ chế này chưa được áp dụng rộng rãi; nó đã được thực hiện đáng chú ý trong RootStock và tồn tại một số dự án thử nghiệm khác.

Khai thác hợp nhất nhằm mục đích tái sử dụng sức mạnh khai thác của Bitcoin để tăng cường bảo mật cho một Blockchain ít phổ biến hơn. Nó hoạt động bằng cách thêm Hash của một số hoặc tất cả các Block trong Blockchain của Crypto thay thế vào của Bitcoin.

5.2.1. RootStock

Rootstock [42] hoạt động như một Sidechain của Bitcoin với 2way-pegging, Bitcoin có thể được chuyển sang Blockchain Rootstock và chúng trở thành “Rootcoins” (RTC) và Rootcoins có thể được chuyển trở lại vào Blockchain Bitcoin. Rootstock cung cấp chức năng tương tự như của Ethereum. Trong bài nghiên cứu [43], các tác giả tuyên bố rằng Roostock VM (RVM) là cấp độ Opcode tương thích với EVM (Ethereum VM). Đơn vị tiền tệ gốc của Rootstock (RTC) được sử dụng để thanh toán cho các thợ đào của Blockchain Rootstock để thực hiện các hợp đồng. Rootstock cũng hỗ trợ hợp nhất khai thác Bitcoin và Rootstock đồng thời, đồng thời cung cấp một số cơ chế bảo vệ, như các điểm kiểm tra và tăng thời gian đáo hạn cho các đồng Coin đã khai thác (khoảng thời gian mà các đồng Coin mới khai thác gần đây không thể được sử dụng), chống lại DoS của những thợ đào Bitcoin.

5.2.2. Namecoin

Namecoin [35] bắt đầu như một nhánh của phần mềm Bitcoin và nhằm mục đích cung cấp một hệ thống đăng ký tên phi tập trung. Nó có thể được sử dụng thay thế cho DNS

và hiện có thể được sử dụng để phân giải các miền .bit. Nó tính một khoản phí nhỏ cho việc đăng ký tên và nó yêu cầu chủ sở hữu cập nhật chúng khoảng 250 ngày một lần, nếu không chúng sẽ hết hạn. Namecoin có Blockchain riêng nhưng nó được liên kết với Bitcoin ở chỗ nó cung cấp khả năng khai thác hợp nhất.

5.3. Tezos

Tezos [22, 21] là một loại Crypto đang được phát triển tại thời điểm viết bài, nhưng các tác giả cung cấp thông số kỹ thuật chi tiết về ngôn ngữ tập lệnh của nó và giải thích về hệ thống.

Tezos tuyên bố là Crypto đầu tiên có thể sửa đổi một cách dân chủ. Nó cung cấp một cơ chế rõ ràng để quyết định các sửa đổi trong tương lai giao thức riêng của nó. Ban đầu nó xem xét cơ chế bỏ phiếu và thời gian dùng thử.

Ngôn ngữ tập lệnh của Tezos dựa trên ngăn xếp nhưng bao gồm các nguyên bản cấp cao như Lambdas, Sets, Maps và một số cho các nhiệm vụ cụ thể theo ngữ cảnh. Nó cung cấp một thông số kỹ thuật đầy đủ để hỗ trợ việc xác minh chính thức các hợp đồng. Nó cũng cung cấp các chức năng nhằm mục đích tăng khả năng đọc của Bytecode như việc gắn nhãn các phần tử trong ngăn xếp và lồng các biểu thức nguyên thủy. Để giải quyết vấn đề của DoS dựa trên tập lệnh, Tezos xác định một giới hạn cố định cho số bước tính toán trên mỗi chương trình.

Tezos cũng cho phép tạo các hợp đồng được lưu trữ trong Blockchain có thể lưu trữ một lượng tiền tệ và dữ liệu (lên đến 16KB) và được kiểm soát bởi người dùng hoặc "người quản lý". Hợp đồng không có tiền tệ sẽ tự động bị hủy.

Ngôn ngữ Contract Script của Tezos là ngôn ngữ bảo thủ nhất trong số các ngôn ngữ Blockchain mới. Đó là một ngôn ngữ ngăn xếp, giống như Bitcoin Script. Tuy nhiên, không giống như Bitcoin Script, ngôn ngữ của Tezos được nhập tĩnh. Ngoài ra, nó có đặc điểm kỹ thuật chi tiết, bao gồm ngữ nghĩa hoạt động chính thức cho ngăn xếp máy (Stack Machine). Điều này giúp dễ dàng xây dựng các xác minh chính thức trong Coq và các công cụ liên quan. Hạn chế chính của ngôn ngữ Tezos là, giống như tập lệnh Bitcoin và Forth, rất khó để lập trình. Các ngôn ngữ ngăn xếp nổi tiếng là khó sử dụng, khiến nó không được coi là ngôn ngữ dành cho người dùng.

5.4. Hawk

Hawk [30] là một hệ thống hợp đồng thông minh phi tập trung, ẩn danh các giao dịch và cung cấp các cơ chế giúp đơn giản hóa nhiệm vụ ẩn các đầu vào và người tham gia vào các hợp đồng thông minh (xem Phần 8.3), thông qua việc sử dụng các cam kết mật mã, bằng chứng không kiến thức (Zero-Knowledge Proof) (xem Phần 8.4), và việc sử dụng bên thứ ba hoặc trình quản lý.[3]

Công việc trong [30] có thể được coi là cung cấp nền tảng cho các hợp đồng thông minh Ethereum, nhưng với khả năng làm cho những người tham gia và các giá trị trong các hành động chuyển đổi ở chế độ riêng tư, theo phong cách của Zerocash, với các hoạt động *đúc (Mint)* và *đổ (Pour)*. Nó trình bày một cách trừu tượng về những gì mà Blockchain có thể được nhìn thấy để cung cấp (trong Phần II-A) và cách tính toán tiến hành trong đó (Phần II-C), bao gồm các tính toán *ngay lập tức* và tính toán *chậm trễ*, được thể hiện thông qua các "Tick" trên chuỗi.

Việc mô tả các hợp đồng được thực hiện bằng cách sử dụng một phần trừu tượng cho phép khai báo các phần riêng tư và công khai, đồng thời có thể được biên dịch thành khái niệm triển khai cấp thấp hơn sử dụng các bằng chứng không kiến thức hoặc SNARK để ẩn đầu vào của các phần riêng tư. Các giao dịch được lập trình theo 3 giai đoạn: *đóng băng (Freeze), tính toán (Compute)* và *hoàn thiện (Finalize)*. Hệ thống cung cấp 2 cấp độ mô tả cho các tính toán: *các chương trình* trừu tượng hơn và *chức năng* cụ thể hơn, với một trình bao bọc (Wrapper) đi từ cái trước đến cái sau. Trình bao bọc này có thể được coi là đóng gói một số ngữ cảnh chung, bao gồm bộ hẹn giờ, bút danh và sổ cái. Các chương trình "lý tưởng" này có thể được xem theo một cách nào đó như một đặc tả của các chương trình cấp thấp hơn.

Các giao thức được mô tả trong bài nghiên cứu [30] chính thức được chứng minh là an toàn theo khuôn khổ Khả năng Tương tác Toàn cầu, chống lại sự trừu tượng của Blockchain được cung cấp.

Tại thời điểm viết bài, không có triển khai công khai nào về Hawk.

[3]Lưu ý, tuy nhiên, trình quản lý không thể ảnh hưởng đến kết quả của việc thực thi các tập lệnh ngoài việc hủy bỏ ở giữa quy trình, xem Phần 7.5.

6. Ý tưởng khác để viết kịch bản

Có thể có các cách tiếp cận khác đối với tập lệnh: đặc biệt, có thể tạo tập lệnh từ các đồ tạo tác khác, bao gồm các ngôn ngữ lập mô hình quy trình nghiệp vụ (BMP - Business Process Modeling) và các máy trạng thái (State Machine) hữu hạn; chúng tôi nhìn vào những điều này ở đây.

6.1. Biên dịch từ ngôn ngữ cấp cao hơn

Trong khi các ngôn ngữ lập trình hoặc máy ảo là một phương tiện lập trình phù hợp cho một số người, thì trong các lĩnh vực khác chuyên biệt hơn, các ngôn ngữ dành riêng cho miền được sử dụng trong thực tế. Một ví dụ chính là mô tả quy trình kinh doanh và có công việc dịch các thông số kỹ thuật được viết bằng các ngôn ngữ như vậy sang các ngôn ngữ tập lệnh Blockchain, ví dụ như ngôn ngữ Solidity.

6.1.1. BPMN: Mô hình và ký hiệu quy trình kinh doanh OMG

BPMN được sử dụng để mô tả các quy trình kinh doanh trong thế giới thực, chẳng hạn như chuỗi cung ứng, đòi hỏi sự hợp tác của một số thực thể. Theo dõi và xác thực quy trình như vậy có thể đạt được với DLT, sử dụng Blockchain như một phương thức đồng bộ hóa ban đầu và một dấu vết kiểm tra bất biến. Các hệ thống như thế này thường được mô tả bằng một ngôn ngữ như BPMN [8]. Công việc gần đây [49] cho thấy cách các thông số kỹ thuật BPMN có thể được chuyển thành các hợp đồng thông minh trong Solidity, tạo thành một phần của hệ thống thời gian chạy lớn hơn có thể giám sát và điều phối việc thực hiện quy trình, cũng như cung cấp các dịch vụ khác cho quy trình, chẳng hạn như thanh toán ký quỹ.

6.1.2. Quy trình nhận biết dữ liệu

Tiếp tục với công việc trong phần trước, một đánh giá khác về khả năng sử dụng Blockchain trong lĩnh vực này [26] lập luận rằng loại phương pháp tiếp cận 'dựa trên đồ tạo tác' này cũng phù hợp với các quy trình trong thế giới thực giàu dữ liệu và đặc biệt nó có thể được xem là cung cấp các khả năng cho việc xác minh và mô hình hóa khái niệm phong phú. Đặc biệt, các tác giả lập luận rằng các hệ thống thực tế liên quan đến dữ liệu, đưa các mô hình của chúng vượt ra ngoài máy trạng thái hữu hạn thuần túy, yêu cầu bằng chứng có sự hỗ trợ của máy móc hoặc công nghệ tự động như kiểm

tra mô hình. Những cách tiếp cận này, vốn đã được sử dụng để xác minh hệ thống BPM, nên có thể mở rộng cho các hệ thống dựa trên Blockchain.

6.2. Máy trạng thái hữu hạn

DSL cho FSM mô tả các giao dịch Blockchain được thảo luận trong một diễn đàn Nxt [19]; Không rõ là điều này đã được thực hiện thêm hay không, nhưng có thể liên kết với đề xuất của Bamboo trong Phần 7.2.

6.3. Hyperledger: Tài liệu hóa Chaincode

Tập đoàn Hyperledger [27], bao gồm IBM và Linux Foundation cùng các tổ chức khác, nhằm mục đích xây dựng công nghệ DLT cho mục đích chung, với việc làm cho nhiều khía cạnh của hệ thống có thể kết nối được, ví dụ như tính bảo mật và (thậm chí là) cơ chế đồng thuận.

Tập lệnh trong mô hình này sử dụng *mã Chaincode*, mã này đầu tiên có liên kết trong ngôn ngữ Go của Google (và cũng được thiết lập để có liên kết Java và JavaScript). Điều này làm cho mô hình tương tự như của Nxt, nhưng sự khác biệt trong Hyperledger là Chaincode được tách biệt thành các vùng chứa Docker, do đó cung cấp một số đảm bảo tự động. Mỗi cá thể Chaincode có thể xác định các biến trạng thái liên tục được lưu trữ trên Blockchain và được cập nhật khi giao dịch được gọi; tổng thể của kho lưu trữ này được gọi là *trạng thái toàn cầu*.

6.4. Quy tắc Logic

Có một lịch sử sử dụng nhiều Logic khác nhau để mô tả các hợp đồng và [28] dựa trên điều này để kiểm tra tính khả thi của việc sử dụng Logic để mô tả các hợp đồng thông minh dựa trên Blockchain. Điều này được thực hiện thông qua việc khám phá một ví dụ điển hình, đầu tiên trong mã giả (Pseudocode) và sau đó trong Logic hợp đồng chính thức (FCL - Formal Contract Logic), một Logic Deontic và khả thi được thực hiện trong công cụ Logic khả thi SPINdle. Khi phương pháp tiếp cận được thiết lập, bài nghiên cứu cũng kiểm tra sự cân bằng giữa việc thực hiện (các phần của) tính toán On-Chain và Off-Chain. Nó kết luận bằng cách lập luận rằng một trường hợp về tính khả thi đã được đưa ra, nhưng lưu ý rằng một thách thức đáng kể đối với việc áp dụng là sự kém hiệu quả tương đối của các cơ chế thực thi Logic.

6.5. Quy trình tính toán (Process Calculi)

Các ngôn ngữ khác cũng đã xuất hiện. Rholang của Synereo dựa trên một hệ thống được hiểu rõ ràng hơn Quy trình tính toán, và nó có một đặc điểm kỹ thuật đã được công bố [41]. Điều này làm cho nó có khả năng thích hợp để xác minh chính thức bằng các công cụ như Coq và các hệ thống đánh máy phong phú như các loại phiên tuyến tính. Tuy nhiên, Rholang cũng phản ánh [33], theo nghĩa là dữ liệu có thể được diễn tả bằng các quy trình, điều này có thể gây ra vấn đề cho cả hai lựa chọn đó. Ngoài ra, nhiều lập trình viên không quen thuộc với quy trình tính toán, điều này sẽ làm tăng thêm chi phí học tập và chi phí tìm hiểu.

Mặt khác, việc sử dụng các quy trình giao tiếp trong Rholang nhấn mạnh quan điểm (cũng được đưa ra bởi đề xuất) rằng các hợp đồng "xã hội" vốn dĩ là những đồ tạo tác đồng thời nên mong muốn rằng sự đồng thời này không bị mất đi thông qua việc tuần tự hóa khi mô hình hóa chúng.

7. Phê bình các hệ thống hiện có

Khi lập trình, các nhà phát triển thường tập trung vào cách sử dụng thông thường; tuy nhiên, trong một tình huống có liên quan đến tiền và các chương trình được phát triển phải tương tác với các tác nhân độc hại tiềm ẩn, điều quan trọng là phải hiểu các khả năng không lường trước được mà ngôn ngữ mang lại, bao gồm bất kỳ hành vi mặc định nào của chương trình và bất kỳ khía cạnh nào có khả năng không xác định. Trong phần này, chúng tôi xem xét một loạt các vấn đề bảo mật được quan sát tuân theo mô hình này.

7.1. Reentrancy

Reentrancy là hành vi chính xác của một hàm hoặc phương thức trong trường hợp nó được gọi ở giữa quá trình thực thi của chính nó. Reentrancy được đảm bảo cho các chức năng đó minh bạch về mặt tham chiếu, nghĩa là chúng không có tác dụng phụ hoặc quyền truy cập vào các biến toàn cục.

Trong trường hợp một hàm gọi một hàm khác mà chúng ta không có quyền kiểm soát, chúng ta phải xem xét khả năng hàm từ xa sẽ gọi lại hàm của chúng ta, do đó chúng ta không thể giả định rằng lệnh gọi hàm của chúng ta sẽ hoạt động nguyên tử [3, 31].

Trong Ethereum, các khoản thanh toán được thực hiện thông qua một lệnh gọi đến một

hàm không xác định. Bởi vì chúng ta thường không có dự định dùng hàm được gọi để thực thi bất kỳ mã Code nào, chúng ta muốn giới hạn phí Gas của lệnh gọi ở một giá trị thấp. Không làm như vậy, cho phép người nhận độc hại thực thi mã tùy ý sau khi thanh toán.

Một ví dụ về thời điểm đây sẽ là một vấn đề là nếu chúng ta có chức năng "rút tiền" lấy tiền từ tài khoản và thực hiện điều này bằng cách gửi tiền đầu tiên cho người dùng và sau đó trừ số tiền khỏi số dư. Người dùng độc hại có thể khiến lệnh gọi thanh toán thực hiện hàm "rút tiền" một cách đệ quy và anh ta có thể làm như vậy cho đến khi ngân hàng hết tiền vì số dư sẽ không được cập nhật. Vấn đề bảo mật này đã được minh họa bằng cuộc tấn công DAO [31].

Có một số vấn đề trong ví dụ này: một là sự thiếu chỉn chu, một vấn đề khác là việc thực thi mã tùy ý ngoài ý muốn.

Tuy nhiên, chúng ta có thể tưởng tượng rằng việc tách biệt Logic khỏi các tác dụng phụ sẽ giúp tránh được loại vấn đề này.

7.2. Bamboo: cải thiện Reentrancy theo phong cách Ethereum

Yoichi Harai đã phát triển một ngôn ngữ cải tiến là Bamboo, [23] dựa trên cách tiếp cận Ethereum đối với các hợp đồng; Đặc biệt, ngôn ngữ của anh ấy cải thiện hành vi Reentrancy. Anh ấy cho rằng những khiếm khuyết của Ethereum là do không có hàng đợi thông điệp cho các hợp đồng, ví dụ như được triển khai trong Erlang.

> "Trong EVM, khi hợp đồng A gọi hợp đồng B, hợp đồng B có thể gọi lại thành hợp đồng A. Tại thời điểm này, hai bản thực thi của hợp đồng A tồn tại, chúng chia sẻ bộ nhớ nhưng không chia sẻ bộ đếm chương trình. Vì vậy, việc lập luận gần như phức tạp giống như việc có hai luồng trên cùng một chương trình. Cả EVM và Solidity đều có đặc tính khó chịu này".
>
> "Ở Bamboo, việc thực hiện chương trình luôn ở một điểm duy nhất, ngay cả trong trường hợp Reentrancy. Đoạn mã

```
sleep_ after_calling (B)
        with reentrancy {... code_reentrant ...};
                ... code_continued ...
```

> gửi một thông điệp đến Callee B. Thông thường, khi Callee B quay trở lại, việc thực thi tiếp tục trong code_continued. Nếu Callee B gọi lại hợp đồng

của chúng tôi, việc thực thi sẽ bị mắc kẹt trong code_reentrant. Điều này ngăn một số mã Code ở xa thay đổi trạng thái trong khi hợp đồng của chúng tôi đang gọi là Callee B". [24]

Mặc dù cách tiếp cận này cho phép mã Code được gọi hoạt động tốt hơn, nhưng vấn đề của việc Callee nhập lại một hàm đã được gọi là vẫn còn và có thể nói đây vẫn là điều cần được giải quyết.

7.3. Các ngoại lệ thời gian chạy ngầm định

Các trường hợp ngoại lệ thể hiện hành vi không phải là dự định. Vì bản chất của chúng, các trường hợp ngoại lệ là những ứng cử viên sáng giá để gây ra các hành vi trong các chương trình không được các nhà phát triển của chúng xem xét. Nếu chúng tôi muốn giúp các nhà phát triển làm cho mã Code của họ có thể dự đoán được, chúng tôi phải nghiên cứu cách giúp họ nhận thức được tất cả các loại ngoại lệ có thể xảy ra.

EVM của Ethereum xử lý các ngoại lệ bằng cách đơn giản là trả về 0 bất cứ khi nào lệnh gọi thất bại [38]. Các nhà phát triển hợp đồng phải xác minh các giá trị trả về trong hàm gọi.

Trong phần này, chúng tôi nghiên cứu một số ví dụ về các trường hợp ngoại lệ đã khiến các cuộc tấn công trên Ethereum xảy ra trong quá khứ. Chúng tôi đề xuất một số ý tưởng mà các ngôn ngữ tập lệnh có thể triển khai để giảm thiểu loại vấn đề này.

7.3.1. Tràn ngăn xếp (Stack Overflow)

Trong Ethereum, hành động gửi một thông điệp (có thể do một lệnh gọi hàm trong Solidity gây ra) làm tăng kích thước của ngăn xếp. Bởi vì việc thực thi một hàm có thể là kết quả của một lệnh gọi hàm khác, chúng tôi không thể đưa ra giả định về lượng không gian ngăn xếp còn lại, do đó, bất kỳ lệnh gọi nào cũng có thể tạo ra một ngoại lệ. Hơn nữa, nếu ngoại lệ không được tự động lan truyền và lập trình viên quên kiểm tra lỗi, việc thực thi sẽ tiếp tục như thể nó là một thực thi bình thường, điều này có thể gây ra các cuộc tấn công nguy hiểm tiềm ẩn [15].

7.3.2. Ngoại lệ về phí Gas

Như chúng ta đã thấy trong Phần 3.1.1, nỗ lực thực hiện các hợp đồng trong Ethereum được kiểm soát bằng phí Gas. Khi bạn phát hành một giao dịch, bạn có thể dự đoán phí

Gas cần thiết (giả sử không có điều kiện chạy đua). Nhưng mã Code là một phần của hợp đồng được lưu trữ trong Blockchain có thể có sẵn bất kỳ lượng phí Gas nào khi được gọi bởi một tác nhân bên ngoài. Do đó, quá trình thực hiện có thể gặp phải ngoại lệ hết phí Gas tại bất kỳ thời điểm nào.

Ethereum thực hiện một dự phòng tốt cho trường hợp này, vì các giao dịch được khôi phục khi chúng hết phí Gas. Nhưng nó tạo ra sự không chắc chắn khi gọi các hàm ngoại lai có thể không thành công nếu không cung cấp đủ phí Gas [15]. Chúng ta không thể biết chắc chắn lượng phí Gas cần thiết để thực hiện lệnh gọi bên ngoài. Điều này cho thấy có thể hữu ích khi tìm ra các cơ chế để giới hạn thời gian thực hiện mà không sử dụng phí Gas. Nó sẽ giảm bớt sự không chắc chắn để có thể đảm bảo các điều kiện tiên quyết khi thực hiện giao dịch (để ngăn chặn các điều kiện chạy đua).

7.3.3. Xử lý các trường hợp ngoại lệ

Có một số ý tưởng cho rằng một ngôn ngữ tập lệnh có thể triển khai để cố gắng tạo ra các lỗi bắt nguồn từ các ngoại lệ thời gian chạy không mong muốn trở nên khó khăn hơn, một số giải pháp này ảnh hưởng đến thiết kế của ngôn ngữ tập lệnh cấp thấp vì các ngôn ngữ bậc cao có thể không thực thi được tính nguyên tử [31]:

- [31] đề xuất tuyên truyền các ngoại lệ thông qua người gọi và hoàn nguyên trạng thái, và cung cấp một cơ chế thích hợp để kiểm soát các trường hợp ngoại lệ.
- Chúng ta có thể muốn có một cách mô tả các trường hợp ngoại lệ sẽ được xử lý cho mỗi tình huống. Chúng ta có thể muốn việc thực thi tiếp tục và bỏ qua các hoạt động không thành công hoặc chúng ta có thể muốn khôi phục mọi thứ nếu bất kỳ hoạt động nào không thành công, nhưng chúng ta có thể muốn xác định điều này một cách rõ ràng hoặc ít nhất có một dự phòng dễ dự đoán.
- Chúng ta có thể muốn buộc nhà phát triển xác định rõ ràng hành vi đặc biệt bất cứ khi nào có cơ hội để nó xảy ra.

7.4. Không hoàn lại

Một vấn đề có thể xảy ra, mặc dù ít nghiêm trọng hơn nhưng cũng có thể gây khó chịu cho người dùng, đó là việc xử lý các điều kiện tiên quyết không đầy đủ. Một lần nữa, thường xảy ra trường hợp các nhà phát triển tập trung vào quy trình thực hiện thông thường và khi các điều kiện tiên quyết của hợp đồng không được đáp ứng, ngay cả khi

việc thực thi bị hủy bỏ, người dùng có thể không được trả lại số tiền đã đầu tư vào hợp đồng [15, 38]. Hơn nữa, ngay cả khi người dùng kiểm tra xem các điều kiện tiên quyết có được đáp ứng hay không trước khi gửi giao dịch, có thể một điều kiện chạy đua thay đổi thực tế này, do đó buộc người dùng phải đối mặt với rủi ro khi thực hiện giao dịch. Làm cho việc thực thi giao dịch trở nên xác định [31] chắc chắn có thể tránh được sự không chắc chắn cho người dùng, nhưng không có cách nào dễ dàng thực thi để loại lỗi này không xảy ra. Một con đường nghiên cứu khả thi là tìm ra cách viết chương trình phân tích tổng thu nhập và kết quả cho mỗi người dùng và đưa ra các biến thể cho tổng này một cách rõ ràng.

7.5. Thất bại một bên

Trong các hợp đồng nhiều giai đoạn phức tạp, người dùng có thể ngừng cộng tác trong việc thực hiện hợp đồng nếu làm như vậy sẽ đi ngược lại lợi ích của họ [15]. Do đó, hợp đồng phải được thiết kế với khả năng xảy ra sự bất hợp tác cuối cùng cho mỗi bước. Thông thường, hành vi bất hợp tác có thể bị trừng phạt bằng cách lưu trữ một khoản tiền gửi cho mỗi người tham gia sẽ bị mất nếu họ chọn không hợp tác và bằng cách thiết lập thời hạn cho mỗi bước của giao thức [30].

Mặt khác, thời hạn nhất sẽ gây ra rủi ro cho người dùng, có thể bị DoS ngăn cản việc hợp tác và do đó, mất khoản tiền ký quỹ ngay cả khi họ sẵn sàng hợp tác.

Một lần nữa, không có cách rõ ràng nào để ngăn những sai lầm này xảy ra, nhưng có thể tìm ra cách thiết kế hợp đồng cho phép xác định tiền gửi và thời gian chờ cho người dùng, tự động trừng phạt người dùng khi họ không hợp tác và toàn bộ giao thức cho phần còn lại. Mặc dù bất kỳ giải pháp nào trong số này có thể sẽ yêu cầu nhiều cân nhắc và hiệu chuẩn cụ thể cho tình huống.

7.6. Trạng thái không thể đoán trước

Bởi vì các giao dịch được xử lý theo Block, thứ tự mà các giao dịch chưa được xử lý sẽ được xử lý là không xác định. Đơn đặt hàng được quyết định bởi thợ đào thành công tại một thời điểm nhất định. Vì các hợp đồng thông minh có thể phụ thuộc vào ngữ cảnh (trạng thái của Blockchain), nên không có gì lạ khi kết quả của chúng là không thể đoán trước [15, 3].

Một giải pháp khả thi cho sự không thể đoán trước có thể là thiết lập một bối cảnh cụ

thể như một điều kiện tiên quyết để các giao dịch được thực hiện, hoặc sử dụng kết quả mong đợi như một điều kiện tiên quyết [31]. Nhưng điều này có thể làm chậm các hợp đồng tương tác cao và buộc người dùng phải gửi lại giao dịch nhiều lần.

Một hậu quả khác là, nếu có bất kỳ lợi ích nào khi là người đầu tiên tạo giao dịch chứa bí mật, kẻ tấn công có thể chặn giao dịch bằng bí mật và thực hiện giao dịch của chính họ bằng cách sử dụng bí mật, và có khả năng khiến giao dịch của họ được ghi lại đầu tiên trong Blockchain. Chúng ta có thể tưởng tượng đây sẽ là một vấn đề quan trọng đối với một hệ thống đăng ký tên như Namecoin (xem Phần 5.2.2) vì những kẻ tấn công có thể thấy rằng người dùng đang cố gắng đăng ký một cái tên và tự đăng ký tên đó, và sau đó tính tiền chuộc cho người dùng vì đã phát hành nó.

Những lý do tại sao kẻ tấn công có thể có được một bản sao của một giao dịch xuất hiện sớm hơn trong Blockchain so với bản gốc bao gồm: may mắn, trả phí cao hơn và kiểm soát đủ mạng lưới hoặc sức mạnh khai thác và sử dụng nó để loại bỏ hoặc gây khó khăn cho việc lan truyền giao dịch của người dùng.

Nhưng đó không phải là hậu quả duy nhất của việc không thể đoán trước được, một số thuộc tính có thể bị thao túng trực tiếp bởi thợ đào mà không cần kiểm soát mạng lưới, giống như trường hợp của dấu thời gian [31] trong Ethereum, được phép có biên độ lên đến 900 giây do độ trễ và khả năng thiếu đồng bộ giữa các Node khác nhau. Cho phép ngôn ngữ cung cấp loại thông tin không đáng tin cậy này có thể đánh lừa các nhà phát triển về cảm giác an toàn sai lầm, điều này có thể dẫn đến các lỗi như sử dụng dấu thời gian làm hạt giống cho sự ngẫu nhiên [31], vì vậy có thể là một quyết định thiết kế khôn ngoan không cung cấp chúng [31], đề xuất sử dụng số chiều cao Block để thay thế và dịch các biểu thức sử dụng dấu thời gian về số chiều cao Block (vì đã biết thời gian dự kiến giữa các Block).

Vấn đề ăn cắp bí mật có thể được giải quyết bằng cách sử dụng các cam kết mật mã hoặc các bằng chứng không kiến thức (xem Phần 8.4).

7.7. Bí mật

Cả Ethereum và Bitcoin đều không cung cấp tính năng ẩn danh cho người dùng theo mặc định, ngoài tùy chọn sử dụng bút danh, nhưng điều đó thường không tự ngăn cản việc sử dụng các kỹ thuật phân tích để phân tích dữ liệu.

Ngoài ra, nhiều hợp đồng thực hiện trò chơi có nhiều người chơi, yêu cầu một số trường

dữ liệu được giữ bí mật trong một thời gian: ví dụ, nếu một trường dữ liệu lưu trữ bước đi tiếp theo của một người chơi, việc tiết lộ nó cho những người chơi khác có thể giúp họ lựa chọn nước đi tiếp theo của mình.

Trong những trường hợp như vậy, để đảm bảo rằng một trường dữ liệu vẫn còn bí mật cho đến khi một sự kiện nhất định xảy ra, hợp đồng phải khai thác các kỹ thuật mật mã phù hợp [3]. Một số kỹ thuật này đã được đề cập trong Phần 5.4 và 8.4.

7.8. Lỗi bất biến

Tính bất biến có thể được coi là mong muốn và không mong muốn tùy thuộc vào hoàn cảnh. Rõ ràng, dữ liệu trong Blockchain là bất biến, là một lựa chọn thiết kế có ý thức, nhưng nó sẽ trở thành một vấn đề khi những gì bất biến là một lỗi [3]. Thách thức không thực sự là kỹ thuật mà là quan liêu, ở chỗ cần phải tìm ra một cách thuận tiện và thuyết phục để quyết định cách quyết định về những thay đổi, nâng cấp và sửa chữa của các hợp đồng và nói chung về các quy tắc trong một hệ thống Crypto phi tập trung. Chúng ta đã thấy trong Tezos (xem Phần 5.3) một cách tiếp cận sau này, giải pháp mặc định được cung cấp cho các bản sửa lỗi trong hợp đồng là có một người quản lý được chỉ định cho các hợp đồng. Nhưng chúng ta có thể thấy rằng có một người quản lý là một hạn chế nếu chúng ta không muốn tin tưởng ai hoặc có thể chúng ta quan tâm đến việc tin tưởng một lược đồ kiểu N trong số M, sẽ yêu cầu một số tập lệnh thực hiện xác thực.

7.9. Mất Ether

Một hệ quả khác của tính bất biến là tiền được gửi đến các địa chỉ không xác định được khóa [3] do nhầm lẫn. Nếu giả định mật mã tiêu chuẩn được giữ nguyên, thì số tiền này về mặt thống kê là không thể phục hồi và bị mất vĩnh viễn (bị đốt). Ngoài tổn thất kinh tế, điều này tạo ra gánh nặng cho Blockchain bởi vì tiền chưa sử dụng không thể bị lãng quên, vì không có bằng chứng nào cho thấy nó thực sự không thể sử dụng (theo như mọi người đều biết ai đó thực sự có thể có chìa khóa).

Để giảm khả năng điều này xảy ra, trong ngữ cảnh của Bitcoin, Base58Check thường được sử dụng để bổ sung dự phòng cho các địa chỉ, do đó một lỗi đánh máy nhỏ không thể tạo ra một địa chỉ hợp lệ khác [2].

7.10. Tính không ngẫu nhiên

Một số tập lệnh yêu cầu nguồn ngẫu nhiên an toàn trong quá trình thực thi [3]. Khi môi trường thực thi là một Blockchain, vấn đề chuyển thành việc tìm kiếm một nguồn ngẫu nhiên có sẵn trên toàn cầu, có thể xác minh độc lập và không thể đoán trước được. Blockchain dường như là một nguồn tốt cho sự ngẫu nhiên như vậy, vì các Hash của các Block mới rất khó dự đoán. Nhưng việc sử dụng nó như một nguồn ngẫu nhiên sẽ mở ra cánh cửa cho sự thao túng tiềm năng của những thợ đào. Do đó, việc tìm ra một cách an toàn để sử dụng Entropy trong Blockchain như một nguồn ngẫu nhiên là một vấn đề mở và được nghiên cứu tích cực [40].

7.11. Turing Complete hay Incomplete?

Một ngôn ngữ tập lệnh Turing Complete có khả năng - ít nhất là về nguyên tắc - cho phép tất cả các hàm có thể tính toán được viết bằng ngôn ngữ này. Điều này có nghĩa là việc thực thi một số tập lệnh sẽ không kết thúc (đối với một số đầu vào) và ngay cả đối với những tập lệnh kết thúc, thời gian thực thi có thể lớn vô thời hạn. Vì vậy, có vẻ như việc chọn một ngôn ngữ Turing Incomplete sẽ là thích hợp. Mặc dù điều này đúng, nhưng bản thân tính không đầy đủ Turing là không đủ.[4] Ví dụ, thời gian tính toán có thể là siêu cấp số nhân ngay cả đối với đệ quy nguyên thủy (với các loại cao hơn), được chứng minh bằng hàm Ackermann. Ngay cả khi không có đệ quy, việc đặt các giới hạn ưu tiên về thời gian thực thi có thể là vấn đề trong các tình huống mà các hợp đồng có thể gọi lẫn nhau và các lệnh gọi có thể mang tính biểu tượng.

8. Công nghệ

8.1. Tính toán có thể xác minh

Tính toán có thể xác minh [37] là một kỹ thuật cho phép tạo ra các bằng chứng tính toán - đó là bằng chứng cho việc đánh giá một chương trình có một kết quả cụ thể - có thể được xác minh nhanh hơn thời gian thực hiện tính toán thực tế. Điều này cho phép các bên không đáng tin cậy thuê ngoài tính toán.

Trong Pinoccio [37], các tác giả tuyên bố đã đạt được một cách "gần như thực tế" để thực hiện điều này, trái ngược với những cách trước đó, mặc dù tiệm cận sẽ hoạt động

[4]Lập luận này được đưa ra trong ngữ cảnh của Ethereum trong Whitepaper: https://github.com/ethereum/wiki/wiki/White-Paper#computation-and-turing-completeness.

đối với một số đầu vào, nhưng trên thực tế, chúng có một hệ số không đổi khả thi (hàng trăm năm cho tính toán nhỏ).

Trong Geppetto [10], các tác giả mô tả các kỹ thuật bổ sung để nâng cao hiệu quả của cách tiếp cận. Công việc trong bài nghiên cứu dành cho một trình biên dịch C thực tế, dựa trên LLVM và giới thiệu một số kỹ thuật tiên tiến nhằm mục đích giảm chi phí điện năng và tăng phạm vi công việc. Tác phẩm Geppetto được xây dựng dựa trên hệ thống Pinocchio, hệ thống này cũng làm nền tảng cho các bằng chứng không kiến thức (xem Phần 8.4).

Tính toán có thể xác minh ít nhất cũng có một ứng dụng thú vị cho các hợp đồng thông minh: chúng sẽ cho phép thợ đào thực hiện tính toán chỉ một lần và tạo ra bằng chứng tính toán, và người xác minh sẽ chỉ cần xác minh rằng bằng chứng đó là chính xác. Điều này sẽ cho phép các hợp đồng đắt tiền hơn được chạy, vì chúng sẽ không làm tăng thời gian cần thiết để xác thực Blockchain theo tỷ lệ.

Về khả năng, các phép tính có thể được thực hiện trực tiếp bởi người dùng gửi các giao dịch. Nhưng điều này có khả năng yêu cầu các giao dịch phải được tính toán lại trong trường hợp ngữ cảnh của chúng thay đổi (điều này có thể xảy ra do điều kiện chạy đua trong quá trình thêm chúng vào Blockchain, xem Phần 7.6).

8.2. Giới hạn có thể xác minh và thư viện có thể sử dụng lại

Ngay cả khi chúng ta không thể đưa ra bằng chứng tính toán, sẽ rất hữu ích nếu cung cấp giới hạn trên đã được chứng minh cho lượng tính toán mà một chương trình sẽ yêu cầu. Bằng cách này, chúng ta có thể tính toán lượng phí Gas tối thiểu đảm bảo hợp đồng được thực hiện, điều này sẽ tránh hoàn toàn ngoại lệ hết phí Gas (với các tác động bảo mật mà điều này có, xem Phần 7.3.2).

Các hợp đồng thường được thiết kế và sử dụng lại nhiều lần với những thay đổi nhỏ đối với các thông số và đầu vào của chúng. Ngay cả khi việc xác minh tự động các hợp đồng chung là không thể thực hiện được, một số hợp đồng có thể tự chứng minh hoặc cung cấp các công thức có thể xác minh tính toán lượng thời gian cần thiết để thực hiện một hợp đồng nhất định, hoặc thậm chí để chứng minh rằng việc thực hiện là chính xác (như là trường hợp với các bài toán NP, mà lời giải có thể được xác minh trong thời gian đa thức). Các công thức có thể tái sử dụng và có thể kiểm chứng như vậy có thể được lưu trữ trong Blockchain.

8.3. Tính toán đa bên

Các giao thức tính toán đa bên an toàn (MPC), cho phép một nhóm các bên ủy thác lẫn nhau tính toán một hàm chung *f* trên các đầu vào riêng tư của họ. Thông thường, tính bảo mật của các giao thức như vậy được xác định theo mô hình lý tưởng trong đó *f* được tính toán bởi một bên đáng tin cậy [1].

Các khía cạnh có thể được đảm bảo trong các cài đặt khác nhau sẽ khác nhau, các giao thức cơ bản cho phép mô phỏng tính toán đơn giản, nhưng Bitcoin đã được sử dụng thành công [1, 4] để thực thi kết quả tính toán và phạt trong trường hợp không hợp tác (xem Phần 7.5).

8.4. Bằng chứng không kiến thức

Bằng chứng không kiến thức (Zero Knowledge Proof) là một cơ chế mật mã để chứng minh việc sở hữu những kiến thức cụ thể mà không tiết lộ những kiến thức đó. [20] Người ta đã chỉ ra rằng có thể chứng minh tương tác mà không cần biết rằng sở hữu lời giải cho bài toán 3 màu của đồ thị và bằng cách mở rộng bất kỳ bài toán NP đầy đủ nào.

Zerocoin [34] và Zerocash [45] sử dụng bằng chứng không kiến thức để thêm tính ẩn danh cho các giao dịch Bitcoin mà không cần thêm bất kỳ bên đáng tin cậy nào. Thuật toán cho phép người dùng đúc một số lượng Zerocoin vào một Pool và sau đó đổi các đồng khác nhau, nhưng đảm bảo rằng mỗi người dùng không thể đổi nhiều đồng hơn số lượng mà họ đã đúc mà không tiết lộ bất kỳ liên kết nào giữa đầu vào và đầu ra. Các bằng chứng không kiến thức cũng được tích hợp trong Hawk (xem Phần 5.4) để đảm bảo tính ẩn danh.

8.5. Mã Code mang theo bằng chứng / bằng chứng tính toán

Trong *mã Code mang theo bằng chứng* [36] mã Code chương trình - trong ví dụ ban đầu, mã Code máy - được kèm theo bằng chứng rằng mã Code thỏa mãn các thuộc tính nhất định, chẳng hạn như kết thúc trong một khoảng thời gian nhất định. Bằng chứng được định nghĩa theo cách mà người nhận có thể kiểm tra được và việc kiểm tra như vậy về cơ bản rẻ hơn đáng kể so với việc người nhận tự làm bằng chứng. Hơn nữa, người nhận không cần phải tin tưởng vào người gửi: bằng chứng cung cấp bằng chứng thiết yếu cho thấy có điều gì đó xảy ra, thay vì phải tin tưởng

vào tính xác thực của một tác nhân khác trong hệ thống. Giả sử rằng định dạng bằng chứng được xuất bản, người nhận thực sự có thể viết trình kiểm tra bằng chứng (Proof Checker) cho mình và không cần tin tưởng bên thứ ba cung cấp định dạng bằng chứng.

Tuy nhiên, điều này có một nhược điểm lớn: không chỉ những người triển khai hệ thống Blockchain cần hiểu và triển khai một ngôn ngữ lập trình, họ cũng sẽ cần hiểu và triển khai một trình kiểm tra bằng chứng hoàn chỉnh. Việc chỉ sử dụng các trợ lý chứng minh như Coq được các nhà triển khai ngôn ngữ lập trình mô tả là yêu cầu bằng tiến sĩ, đừng bận tâm đến việc xây dựng một bằng tiến sĩ. Bây giờ, mặc dù điều này không thực sự đúng, nhưng nó truyền tải sự phức tạp của nhiệm vụ. Mặt khác, đây có thể là một điều tốt, vì nó tạo ra rào cản cho những người muốn phát triển trình xác thực và có thể ngăn cản tất cả trừ những người kiên quyết nhất (và có kỹ năng), do đó tránh việc sản xuất phần mềm trình xác thực được thiết kế kém có thể đe dọa mạng lưới Blockchain.

Một khái niệm liên quan là *bằng chứng tính toán*, trong đó bằng chứng đi kèm với một giá trị: trong trường hợp này, giá trị được coi là kết quả của việc đánh giá một biểu thức nhất định và bằng chứng cung cấp bằng chứng cho thấy trường hợp này xảy ra.

8.6. Sử dụng bộ tổ hợp

Bộ tổ hợp là các hàm bậc cao hơn, có thể bằng một ngôn ngữ hàm không định kiểu (như λ-Calculus) hoặc ngôn ngữ đã định kiểu (chẳng hạn như Haskell). Các bộ hàm như vậy tạo thành các thư viện tổ hợp, và chúng đã được sử dụng để xác định các *ngôn ngữ dành riêng cho miền* (DSL) nhỏ; các ví dụ nổi tiếng bao gồm thư viện phân tích cú pháp và mô tả phần cứng. [39] Nghiên cứu cách các bộ tổ hợp có thể được sử dụng để đại diện cho các hợp đồng tài chính điển hình. Nó nhằm mục đích xác định một ngôn ngữ tổ hợp gần với thuật ngữ được sử dụng trong bối cảnh tài chính.

Mô hình là hợp đồng với hai bên: bên nắm giữ và bên phản đối. Được xem là tồn tại theo thời gian, với thời hạn (ngày hết hạn) và có thể có hạn chế về ngày mua. Các hợp đồng có thể được kết hợp thông qua kết hợp, tách rời và cái này đến cái kia (tuần tự), cũng như có thể "hoán đổi cực" như nó vốn có. Cho thấy các hợp đồng phức tạp có

thể được mô tả như thế nào bằng DSL: bao gồm các tùy chọn kiểu "Mỹ" và "Châu Âu".

Bài nghiên cứu [39] trình bày, ngoài bản thân thư viện bộ tổ hợp, việc xác định giá trị hợp đồng, dựa trên máy móc ngẫu nhiên và cách triển khai cụ thể của máy móc định giá này.

8.7. Sử dụng các kiểu phụ thuộc đa hình và tác dụng phụ đại số

Các kiểu phụ thuộc cho phép người dùng khai báo nhiều kiểu biểu đạt hơn cho chương trình của họ. Thật vậy, thông qua phép đẳng cấu "Curry-Howard", các loại có thể được xác định bằng các hàm mệnh đề trong Logic vị từ, và các thành viên của các loại đó bằng các bằng chứng về các mệnh đề đó. Vì những lý do này, một ngôn ngữ biểu đạt có thể là một phương tiện thích hợp để đảm bảo rằng các hợp đồng thông minh có một số đặc tính nhất định theo thiết kế. Bài nghiên cứu [38] nghiên cứu điều này, và đặc biệt xem xét cách Idris hoạt động, với sự kết hợp của các kiểu phụ thuộc và hiệu ứng đại số, có thể được sử dụng để tạo ra hiệu quả tốt trong không gian này. Đặc biệt, nó quản lý mô hình xử lý phí Gas và trạng thái toàn cầu theo cách an toàn hơn.

Chi tiết hơn, nó đề xuất sử dụng các kiểu phụ thuộc và đa hình, và khai báo hiệu ứng phụ đại số để ngăn chặn chúng. Mục đích là để tránh các kết quả không mong muốn bằng cách tuyên bố các kết quả mong đợi và tác dụng phụ một cách rõ ràng. Bài nghiên cứu cũng cung cấp bản dịch từ Idris sang Serpent, ngôn ngữ cấp cao kiểu Python cho Ethereum. Cuối cùng, bài nghiên cứu xem xét một số vấn đề phổ biến của việc phát triển hợp đồng, nhưng chúng được lấy trực tiếp từ [15], được thảo luận trong Phần 7. Khi đánh giá công việc, bài nghiên cứu kết luận rằng quy trình tính toán cộng với các loại hành vi có thể cung cấp một giải pháp tốt hơn.

8.8. Xác minh chính thức hợp đồng

"Việc chứng minh các chương trình là đúng" đã được thảo luận trong hơn 30 năm, nhưng đang dần trở nên phổ biến. Một số công việc đã được thực hiện khi áp dụng điều này cho các hợp đồng thông minh, với hai ví dụ đáng chú ý về việc xác minh các hợp đồng Ethereum. Cách tiếp cận đầu tiên [5] sử dụng F*[5], là một ngôn ngữ lập trình hàm ML-Like nhằm xác minh chương trình, để xác minh các hợp đồng Ethereum được viết

[5]https://www.fstar-lang.org

bằng Solidity. Nó trình bày ba cách khác nhau để giải quyết vấn đề.

- Đầu tiên, nó trình bày một hệ thống Solidity*, được viết bằng OCaml và dịch các chương trình được viết bằng Solidity sang F*, (cấu thành một cách nhúng "Shallow" của ngôn ngữ). Sau đó có thể xác minh các thuộc tính của các hàm đã dịch. Công việc này đã phát hiện một số vấn đề về sự hấp dẫn lại, cũng như các lệnh gọi chưa được kiểm tra để gửi và nó được đánh giá trên một số lượng nhỏ các tập lệnh cấp nguồn[6].
- EVM Bytecode có thể được dịch ngược thành F*, sử dụng một công cụ khác gọi là EVM*, cũng được viết bằng OCaml. Sau đó, có thể xác minh các thuộc tính của mã Code, ví dụ như tiêu thụ phí Gas, chạy trong một trình biên dịch nhẹ của một tập con của tập lệnh.
- Với hai bản dịch, cũng có thể cho thấy sự tương đương của các bản dịch Solidity* và EVM*, ít nhất là khi có mã nguồn Solidity.

Máy ảo Ethereum (EVM) cũng đã được chính thức hóa [25] trong trợ lý chứng minh Isabelle/HOL và các khía cạnh của hành vi của một hợp đồng điển hình - hợp đồng Chứng thư là một phần của Dịch vụ tên Ethereum đã được xác minh. Đặc biệt, đặc tính an toàn "chỉ có công ty đăng ký mới có thể giảm số dư " đã chính thức được thiết lập. Như được mong đợi, một trong những lợi ích của việc chính thức hóa như vậy là [25] chứa một tuyên bố rõ ràng về tập hợp các giả định mà theo đó kết quả nắm giữ, và những lợi ích này đại diện cho kết quả của một bài tập mô hình hóa khái niệm nghiêm túc. Báo cáo chỉ ra các công việc tiếp theo,[7] chẳng hạn như xác nhận việc thực hiện EVM và chính thức hóa của phí Gas, để các đặc tính tồn tại ("điều gì đó tốt sẽ xảy ra") có thể được chứng minh bên cạnh đặc tính an toàn hiện có ("không có gì xấu xảy ra").

8.9. Phân tích tĩnh các hợp đồng

Có thể dễ dàng kiểm tra một số lỗi bằng cách áp dụng các công cụ phân tích tĩnh. Cách tiếp cận này đã được khám phá rộng rãi cho các ngôn ngữ lập trình như C. Nhưng đặc biệt, chúng tôi muốn làm nổi bật ứng dụng của nó như một bản sửa lỗi nhanh cho các

[6]Hãy lưu ý rằng phần lớn các tập lệnh Ethereum trên Blockchain chỉ có sẵn dưới dạng mã Code EVM.

[7]Điều này được nói nhiều hơn trên Reddit:
https://www.reddit.com/r/ethereum/comments/59wr6w/formal_verification_of_deed_contract_in_ethereum/

ngôn ngữ Ethereum EVM thông qua công cụ Oyente [31], có thể được sử dụng cho cả việc tìm kiếm lỗi trong hợp đồng do người dùng viết và để cho phép người dùng tránh sử dụng các hợp đồng lỗi đã được triển khai.

8.10. Cây cú pháp trừu tượng Merkelized

MAST (Merkelized Abstract Syntax Trees) [44] là một đề xuất cho phép các tập lệnh xác thực giao dịch Bitcoin được lưu trữ ở dạng Hash một phần.

Cấu trúc hiện có của một tập lệnh xác thực Bitcoin chỉ đơn giản là một chương trình Script bao gồm một chuỗi các hoạt động Script. Tuy nhiên, có một cấu trúc hợp lý đối với các chương trình Script, như trong tất cả các chương trình có dạng cây. Cụ thể, mọi hoạt động có điều kiện đều khiến chương trình phân nhánh thành hai chương trình con Logic, một chương trình cho trường hợp điều kiện đúng và một cho trường hợp điều kiện sai.

Đối với bất kỳ tập lệnh nhất định nào, việc xác thực thành công liên quan đến việc truyền xuống một Block mã Code lá trong một cây như vậy, đã vượt qua hoặc thất bại trong các bài kiểm tra tương ứng. Tất cả các nhánh khác của cây đều không liên quan đến việc xác nhận được đề cập.

Đề xuất MAST tận dụng thực tế này và đề xuất thay thế tất cả các nhánh không liên quan bằng các Hash của chúng, do đó loại bỏ khỏi các tập lệnh xác thực tất cả các phần của tập lệnh không cần thiết để thực hiện xác thực. Việc triển khai MAST trong một Blockchain hoàn toàn mới phải đơn giản. Tuy nhiên, có một cảnh báo chính. MAST dựa trên thực tế rằng việc đánh giá mã Code chỉ liên quan đến việc chạy qua một danh sách các hoạt động để thực hiện một lần theo thứ tự rõ ràng. Các đường dẫn thực thi có thể tương ứng trực tiếp với các nhánh chương trình. Các cấu trúc lập trình phong phú hơn như vòng lặp, đệ quy, các hàm được xác định, v.v. khiến việc sử dụng MAST khó hơn, có thể là không thể. Điều này phát sinh bởi vì các phần của chương trình có liên quan đến tính toán. Do đó đối với việc xác nhận, không thể được mô tả ngay từ đầu như là một trong số ít các đường dẫn đến một Node lá trong cây mã Code. Thay vào đó, các đường dẫn liên quan được xác định bằng cách thực thi và có thể liên quan đến nhiều phần phân tán của chương trình. Tuy nhiên, vẫn có thể sử dụng MAST trong cài đặt này, nhưng đó là một câu hỏi mở về cách chính xác để thực hiện điều đó.

9. Kết luận

Trong bài nghiên cứu này, chúng tôi đã khảo sát một số ví dụ tiêu biểu về việc sử dụng nâng cao Crypto và Blockchain ngoài việc sử dụng cơ bản làm phương thức thanh toán. Chúng tôi đã tập trung vào các giải pháp tập lệnh hiện có, điểm mạnh và điểm yếu của chúng và một số giải pháp hiện có cho các vấn đề đã biết với chúng.

Chúng tôi thấy rằng, trong khi đã có nhiều nỗ lực đa dạng theo nhiều hướng khác nhau, vẫn còn nhiều câu hỏi mở, chưa có lời giải phổ quát và còn nhiều chỗ cho các nghiên cứu và thử nghiệm trong tương lai.

Chúng tôi rất biết ơn Input Output Hong Kong (https://iohk.io/) đã tài trợ cho công việc dẫn đến bài nghiên cứu này. Lior Yaffe vui lòng đánh giá chúng tôi về các chi tiết của hệ thống Nxt, và Maria Christakis và Valentin Wüstholz đã đưa ra một số nhận xét hữu ích trong phiên bản trước của bài nghiên cứu.

Tài liệu tham khảo

[1] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski và Lukasz Mazurek. Tính toán đa bên an toàn trên Bitcoin. Trong Hội nghị chuyên đề IEEE về Bảo mật và Quyền riêng tư năm 2014, trang 443 – 458. IEEE, 2014.

[2] Andreas M Antonopoulos. Mastering Bitcoin. O'Reilly Media, 2014.

[3] Nicola Atzei, Massimo Bartoletti và Tiziana Cimoli. Một cuộc khảo sát về các cuộc tấn công vào hợp đồng thông minh Ethereum. Kho lưu trữ ePrint của Cryptology: Báo cáo 2016/1007, https://eprint.iacr.org/2016/1007, 2016.

[4] Iddo Bentov và Ranjit Kumaresan. Cách sử dụng Bitcoin để thiết kế các giao thức công bằng. Trong Hội nghị Mật mã Quốc tế, trang 421 – 439. Springer, 2014.

[5] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Natalia Kulatova, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy và Santiago Zanella-Béguelin. Xác minh chính thức các hợp đồng thông minh: Bài viết ngắn. Trong Kỷ yếu Hội thảo ACM 2016 về Ngôn ngữ Lập trình và Phân tích Bảo mật, PLAS '16. ACM, 2016.

[6] Tranh luận về giới hạn kích thước Block. https://en.Bitcoin.it/wiki/Block_size_limit_controversy, 2010. [truy cập lần cuối ngày 21-11-2016].

[7] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A.

Kroll và Edward W. Felten. SoK: Quan điểm nghiên cứu và thách thức đối với Bitcoin và Crypto. https://eprint.iacr.org/2015/261.pdf, 2015. [truy cập lần cuối 27-11-2016].

[8] BPMN. http://www.bpmn.org, 2016. [truy cập lần cuối ngày 21-11-2016].

[9] Colored Coins. https://en.Bitcoin.it/wiki/Colored_Coins, 2014. [truy cập lần cuối ngày 21-11-2016].

[10] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno và Samee Zahur. Geppetto: Tính toán linh hoạt có thể xác minh. Trong Hội nghị chuyên đề IEEE về Bảo mật và Quyền riêng tư năm 2015, các trang 253–270. IEEE, 2015.

[11] Đối tác. http://counterparty.io/, 2014. [truy cập lần cuối ngày 21-11-2016].

[12] Đối tác tại Wikipedia. https://en.wikipedia.org/wiki/ Counterparty_ (công nghệ), 2014. [truy cập lần cuối ngày 21-11-2016].

[13] Karl Crary và Michael J Sullivan. Cam kết liên kết ngang hàng bằng Bitcoin. Thông báo ACM SIGPLAN, 50 (6): 479–488, 2015.

[14] Phương pháp lưu trữ dữ liệu tại wiki Colored Coins. https://github.com/ Coloured-Coins/Coloured-Coins-Protocol-Specification/wiki/ Data% 20Storage% 20Methods, 2015. [truy cập lần cuối 21-11-2016].

[15] Kevin Delmolino, Mitchell Arnett, Ahmed E Kosba, Andrew Miller và Elaine Shi. Từng bước hướng tới việc tạo ra một hợp đồng thông minh an toàn: Bài học và hiểu biết sâu sắc từ một phòng thí nghiệm Crypto. IACR Cryptology ePrint Archive, 2015: 460, 2015.

[16] Cơ sở lý luận về thiết kế tại Ethereum Wiki. https://github.com/ethereum/wiki/ wiki / Design-Rationale, 2014. [truy cập lần cuối 23-11-2016].

[17] Đã tắt Opcodes ở mã nguồn khách hàng Bitcoin. https://github.com/bitcoin/bitcoin/blob/57b34599b2deb179ff1bd97ffeab91ec9f90 4d85 /src/script/interpreter.cpp # L288, 2010. [truy cập lần cuối ngày 21-11-2016].

[18] Công nghệ sổ cái phân tán: ngoài Blockchain. https://www.gov.uk/goGovernment/news/distributed-ledger-technology-beyond-blockchain, 2016. [truy cập lần cuối 27-11-2016].

[19] FSM và Nxt. https://nxtforum.org/automated-transactions/ats with-fsm-based-dsl

/, 2014. [truy cập lần cuối 21-11-2016].

[20] O Goldreich, S Micali, và A Wigderson. Các bằng chứng không mang lại kết quả gì ngoài giá trị của khẳng định của họ. Preprint, 1986.

[21] LM Goodman. Tezos - Sách trắng về sổ cái Crypto tự sửa đổi. https://tezos.com/pdf/white_paper.pdf, 2014.

[22] LM Goodman. Tezos: Báo cáo định vị Crypto tự sửa đổi. https://tezos.com/pdf/position_paper.pdf, 2014.

[23] Yoichi Hirai. Bamboo: một ngôn ngữ hợp đồng thông minh phôi thai. https://github.com/pirapira/bamboo, 2016. [truy cập lần cuối 24-11-2016].

[24] Yoichi Hirai. Ethereum Reentrancy. Giao tiếp riêng, 2016.

[25] Yoichi Hirai. Xác minh Chính thức Hợp đồng Chứng thư trong Dịch vụ Tên Ethereum. https://yoichihirai.com/deed.pdf, 2016. [truy cập lần cuối 24-11-2016].

[26] Richard Hull, Vishal S. Batra, Yi-Min Chee, Alin Deutsch, Fenno F. Terry Heath, III và Victor Vianu. Hướng tới Ngôn ngữ Hợp tác Kinh doanh Sổ cái được Chia sẻ dựa trên Quy trình Nhận biết Dữ liệu. Trong Proc. Intl. Conf. on Service Oriented Computing (ICSOC), 2016.

[27] Hyperledger. https://www.hyperledger.org/blog, 2016. [truy cập lần cuối 21-11-2016].

[28] Florian Idelberger, Guido Governatori, Régis Riveret, và Giovanni Sartor. Đánh giá vị trí của các hợp đồng thông minh dựa trên Logic cho các hệ thống Blockchain. Trong Hội thảo Quốc tế về Quy tắc và Ngôn ngữ Đánh dấu Quy tắc cho Web Ngữ nghĩa, trang 167–183. Springer, 2016.

[29] Có kích thước tối đa của scriptSig/scriptPubKey không? tại Sàn giao dịch Bitcoin Stack. http://Bitcoin.stackexchange.com/questions/35878/is there-a-Maximum-size-of-a- scriptsig-scriptpubkey, 2015. [truy cập lần cuối ngày 21-11-2016].

[30] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, và Charalampos Papamanthou. Hawk: Mô hình Blockchain của mật mã và bảo vệ quyền riêng tư hợp đồng thông minh. Đại học Maryland và Đại học Cornell, 2015.

[31] Loi Luu, Chu Duc Hiep, Hrishi Olickel, Prateek Saxena, và Aquinas Hobor. Tạo hợp đồng thông minh thông minh hơn. Trong Kỷ yếu ACM SIGSAC 2016 Hội nghị về Bảo mật Máy tính và Truyền thông, trang 254–269. ACM, 2016.

[32] Số Op_code tối đa trong tập lệnh tại Bitcoin StackExchange. http://Bitcoin.stackexchange.com/questions/38230/maximum number-of-op-Code-in-script, 2015. [truy cập lần cuối ngày 21-11-2016].

[33] L. Gregory Meredith và Matthias Radestock. Một phép tính bậc cao phản chiếu. Electr. Notes Theor. Comput., 141 (5): 49–76, 2005.

[34] Ian Miers, Christina Garman, Matthew Green và Aviel D Rubin. Zerocoin: Crypto được phân phối ẩn danh từ Bitcoin. Trong Bảo mật và Quyền riêng tư (SP), Hội nghị chuyên đề IEEE 2013, trang 397–411. IEEE, 2013.

[35] Namecoin. https://namecoin.org/, 2011. [truy cập lần cuối ngày 21-11-2016].

[36] George C. Necula và Peter Lee. Các tác nhân an toàn, không đáng tin cậy sử dụng bằng chứng mang theo mã Code. Trong tác nhân và bảo mật di động, trang 61–91, Luân Đôn, Vương quốc Anh, Vương quốc Anh, 1998. Springer-Verlag.

[37] Bryan Parno, Jon Howell, Craig Gentry và Mariana Raykova. Pinocchio: Tính toán có thể kiểm chứng gần như thực tế. Trong Bảo mật và Quyền riêng tư (SP), Hội nghị chuyên đề IEEE 2013, trang 238–252. IEEE, 2013.

[38] Jack Pettersson và Robert Edström. Hợp đồng thông minh an toàn hơn thông qua phát triển theo kiểu. https://publications.lib.chalmers.se/records/ toàn văn bản /234939 /234939.pdf, 2106.

[39] Simon Peyton Jones, Jean-Marc Eber và Julian Seward. Soạn hợp đồng: Một cuộc phiêu lưu trong kỹ thuật tài chính (viên ngọc trai chức năng). Trong Kỷ yếu của Hội nghị quốc tế ACM SIGPLAN lần thứ năm về lập trình hàm, ICFP '00. ACM, 2000.

[40] Cécile Pierrot và Benjamin Wesolowski. Tính dễ uốn nắn của Blockchain. Cryptology ePrint Archive, 2016/370, https://eprint.iacr.org/2016/370.pdf, 2016.

[41] Hợp đồng, thành phần và quy mô Đặc điểm kỹ thuật của Rholang 0.1. https://docs.google.com/document/d/1gnBCGe6KLjYnahktmPSm_-8V4jX53Zk10J-KFQl7mf8/edit?Pref=2&pli=1#header=h.k4bk2akncduu, 2016. [truy cập lần cuối ngày 11-12-2016].

[42] Rootstock. http://www.rsk.co/, 2014. [truy cập lần cuối ngày 21-11-2016].

[43] Rootstock Whitepaper. http://www.the-Blockchain.com/docs/Rootstock-WhitePaper-Overview.pdf, 2015. [truy cập lần cuối ngày 21-11-2016].

[44] Naik Manali Rubin, Jerry và Nitya Subramanian. Cây cú pháp trừu tượng

Merkelized. http:// www.mit.edu/~jlrubin/public/pdfs/858report.pdf, 2016. [truy cập lần cuối ngày 11-12-2016].

[45] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer và Madars Virza. Zerocash: Thanh toán ẩn danh phi tập trung từ Bitcoin. Trong Hội nghị chuyên đề IEEE về Bảo mật và Quyền riêng tư năm 2014, trang 459–474. IEEE, 2014.

[46] Tập lệnh - Wiki Bitcoin. https://en.Bitcoin.it, 2010. [truy cập lần cuối ngày 21-11- 2016].

[47] Serpent tại Ethereum Wiki. https://github.com/ethereum/wiki/wiki/ Serpent, 2014. [truy cập lần cuối ngày 21-11-2016].

[48] Tài liệu về Solidity. https://solidity.readthedocs.io/en/ Develop /, 2016. [truy cập lần cuối ngày 21-11-2016].

[49] Ingo Weber, Xiwei Xu, Régis Riveret, Guido Governatori, và Alexander Pono marev và Jan Mendling. Giám sát và thực thi quy trình kinh doanh không đáng tin cậy bằng cách sử dụng Blockchain. Trong Intl. Conf. Business Process Mgmt. (BPM), năm 2016.

[50] Forks, Alt-coins, Meta-coins và Sidechains là gì? https://coincenter.org/entry /what-are-forks-alt-coins-meta-coins-and-sidechains, 2015. [truy cập lần cuối 21-11-2016].

[51] Các bản ghi sự kiện hợp đồng được lưu trữ ở đâu trong kiến trúc Ethereum? tại Ethereum StackExchange. http://ethereum.stackexchange.com/questions/1302/ where-do-contract-event-logs-get-Managed-in-the-ethereumarchitecture, 2013. [truy cập lần cuối ngày 21-11-2016].

Người dịch: Nguyễn Văn Tú

Telegram: https://t.me/Tulibra

Link gốc: https://iohk.io/en/research/library/papers/scripting-smart-contracts-for-distributed-ledger-technology/