Ofelimos: Tối ưu hóa Tổ hợp thông qua Bằng chứng Công việc Hữu ích

Một giao thức Blockchain an toàn đáng tin cậy

Matthias Fitzi
IOHK
matthias.fitzi@iohk.io

Aggelos Kiayias
Đại học Edinburgh & IOHK
akiayias@inf.ed.ac.uk

Giorgos Panagiotakos
IOHK
giorgos.panagiotakos@iohk.io

Alexander Russell
Đại học Connecticut & IOHK
acr@cse.uconn.edu

Ngày 14 tháng 10 năm 2021

Tóm lược

Giảm thiểu chi phí năng lượng và lượng khí thải Carbon của Blockchain Bitcoin và các giao thức liên quan là một trong những câu hỏi mở được xác định rộng rãi nhất trong lĩnh vực Crypto. Việc thay thế bằng chứng công việc (PoW) nguyên thuỷ trong giao thức chuỗi dài nhất của Nakamoto bằng bằng chứng công việc hữu ích (PoUW) về mặt lý thuyết từ lâu đã như một giải pháp lý tưởng ở nhiều khía cạnh. Nhưng cho đến nay, khái niệm này vẫn thiếu cách xác thực an toàn một cách thuyết phục.

Trong nghiên cứu này, chúng tôi đã đưa ra Ofelimos, một giao thức Blockchain dựa trên PoUW mới, có cơ chế đồng thuận đồng thời nhận ra một trình giải quyết vấn đề tối ưu hóa phi tập trung. Giao thức của chúng tôi được xây dựng dựa trên một thuật toán tìm kiếm cục bộ mới, mà chúng tôi gọi là Tìm kiếm cục bộ song song kép (DPLS-Doubly Parallel Local Search). Nó được thiết kế đặc biệt để phù hợp với việc triển khai dưới dạng thành phần PoUW của giao thức

Blockchain. Chúng tôi cung cấp phân tích bảo mật kỹ lưỡng về giao thức và trình bày thêm các chỉ số thể hiện tính hữu ích của hệ thống. Như một ví dụ minh họa, chúng tôi cho thấy cách DPLS có thể triển khai một biến thể của WalkSAT và thực nghiệm chứng minh khả năng cạnh tranh của nó đối với việc triển khai WalkSAT đơn giản. Bằng cách này, công việc của chúng tôi mở đường cho việc sử dụng an toàn các hệ thống Blockchain, làm công cụ tối ưu hóa chung cho nhiều vấn đề tối ưu hóa khó mà mong muốn có một giải pháp có thể xác minh công khai.

Nội dung

1. Giới thiệu

Các giao thức Blockchain dựa trên Proof of Work (PoW) tận dụng công việc được thực hiện bởi những người tham gia giao thức, được gọi là thợ đào, để đảm bảo tính bảo mật của sổ cái giao dịch được duy trì. Trong các thiết kế Blockchain nổi bật nhất theo mô hình của Bitcoin [46], công việc được thực hiện không phục vụ mục đích nào khác ngoài việc duy trì bảo mật. Đó là các giao thức không được phép và thúc đẩy. Do đó sẽ có phần thưởng cho những thợ đào tiềm năng quyết định tham gia giao thức và cam kết nỗ lực tính toán. Điều này dẫn đến việc chi tiêu năng lượng ngày càng tăng trong các hệ thống như Bitcoin.

Tại thời điểm viết bài này, Bitcoin có mức chi tiêu năng lượng hàng năm ngang bằng với nhiều quốc gia vừa và nhỏ (xem ví dụ: Chỉ số tiêu thụ điện Bitcoin của Cambridge, https://cbeci.org).

Xu hướng trên đã được xác định sớm như một khía cạnh quan trọng trong hệ sinh thái Bitcoin, dẫn đến hai con đường chính để cải thiện tiềm năng của giao thức Blockchain cơ bản. Đầu tiên là nhằm mục đích thay đổi cơ chế PoW thành một tài nguyên khác có thể có các đặc điểm "xanh hơn", ví dụ: bằng chứng cổ phần [17, 29, 37], bằng chứng không gian [21, 50], bằng chứng không gian-thời gian [44], và các cơ chế tương tự. Tuy nhiên, mối quan tâm chung đối với những cách tiếp cận này là sự thay đổi của bảo mật nguyên thuỷ cơ bản (từ "công việc" sang thứ khác) và tác động không thể tránh khỏi của sự thay đổi này đối với các đảm bảo an ninh của hệ thống. Hướng thứ hai cải thiện vấn đề này và là trọng tâm của nghiên cứu này. Đó là sử dụng lại nỗ lực tính toán đã đầu tư để giải quyết các vấn đề trong thế giới thực. Do đó, hướng này làm nổi bật cách tiếp cận thiết kế bằng chứng công việc hữu ích (PoUW) cho các giao thức Blockchain.

Các thiết kế và nỗ lực triển khai ban đầu như Noocoin [16] và Primecoin [38] đã làm nổi bật vấn đề cơ bản mà từ đó sẽ tiến tới một hệ thống PoUW mạnh mẽ. Nếu công việc được giải quyết thực sự hữu ích thì những kẻ tấn công

có thể đưa hệ thống theo hướng giải quyết các trường hợp vấn đề dễ dàng đối với chúng (ví dụ: do tính toán trước hoặc lợi thế riêng tư khác do cấu trúc không gian thể hiện cơ bản). Do đó các đảm bảo bảo mật là không rõ ràng. Đồng thời, việc giảm thiểu khả năng thao túng hệ thống của kẻ tấn công có thể khiến các tính toán của hệ thống trở nên vô dụng trong thực tế (ví dụ: Primecoin [38] và Gapcoin [23] tính toán chuỗi các số nguyên tố Cunningham và khoảng cách giữa các số nguyên tố tương ứng - cả hai đối tượng toán học có tính hữu dụng đáng ngờ).

Đóng góp của chúng tôi. Chúng tôi đề xuất giao thức Blockchain dựa trên PoUW đầu tiên đi kèm với phân tích tính hữu ích và bảo mật kỹ lưỡng. Trọng tâm trong việc xây dựng là một thuật toán có mục đích chung mới cho tìm kiếm cục bộ ngẫu nhiên được gọi là Tìm kiếm cục bộ song song kép (DPLS). Kỹ thuật quan trọng để thiết kế giao thức là đưa toàn bộ việc thực thi giao thức Blockchain vào một công cụ DPLS thực hiện các bước của thuật toán một cách rõ ràng và có thể xác minh công khai. Hoạt động PoUW trong giao thức đồng thuận có các thợ đào chạy chung DPLS trên các phiên bản do các khách hàng quan tâm đóng góp. Kết quả chi tiết hơn của chúng tôi như sau.

(I) Tìm kiếm cục bộ song song kép. Chúng tôi đưa ra một thuật toán ngẫu nhiên mới cho tìm kiếm cục bộ. Với DPLS, chúng tôi đạt được mục tiêu theo hai hướng sau: (i) Thuật toán thể hiện một cách thích hợp trong cấu trúc, các thuộc tính ngẫu nhiên của hoạt động Blockchain không được phép cơ bản để có thể coi việc thực thi Blockchain như một máy ảo chạy thuật toán; (ii) Tìm kiếm cục bộ ngẫu nhiên là một mô hình thuật toán chung và mạnh mẽ để giải quyết các vấn đề tối ưu hóa khó tính toán. Do đó, DPLS có thể được sử dụng trong nhóm rộng các biến thể thuật toán tìm kiếm cục bộ ngẫu nhiên và được đánh giá tính hữu ích đối với các vấn đề có giá trị thực cao.

DPLS là một thuật toán tìm kiếm cục bộ ngẫu nhiên có mục đích chung dựa trên thuật toán cơ bản M, được gọi là thuật toán thăm dò, kiểm tra một tập hợp các điểm nhất định trong các giải pháp và tạo ra một điểm khác, sau một số

khám phá cục bộ yêu cầu nỗ lực tính toán vừa phải. Dựa trên *M*, DPLS tuân theo chiến lược tìm kiếm song song kép trong đó một số đường dẫn được theo đuổi song song và trong mỗi đường dẫn, một số luồng thăm dò qua *M* được thực hiện trong đó đường dẫn tốt nhất theo chức năng tính điểm sẽ được chọn.

(II) Tính toán DAG vừa phải. Để xem xét khả năng sử dụng bộ giải DPLS trong cài đặt bằng chứng công việc, điều cần thiết là có thể biểu thị các điều kiện độ cứng mà theo đó việc chạy bài toán phụ thăm dò cơ bản thể hiện *độ cứng vừa phải* (Moderate Hardness - MH). Thuộc tính này là yêu cầu cần thiết để một bài toán tính toán có thể áp dụng trong cài đặt Blockchain.

Điều làm cho mô hình trở nên thách thức hơn so với trường hợp của thuật toán PoW của Bitcoin là chúng ta không thể sử dụng một mô hình lý tưởng hóa (chẳng hạn như Mô hình Oracle ngẫu nhiên) và chúng ta phải thể hiện thuộc tính độ cứng vừa phải theo cách có thể phù hợp được sử dụng trong các đối số bảo mật của giao thức Blockchain.

Để nắm bắt được điều này và đồng thời tái hiện bản chất có thể song song hóa của DPLS, chúng tôi tập trung vào trừu tượng tính toán DAG đã được sử dụng rộng rãi trong mô hình tính toán song song (ví dụ, xem [49, 2]). Trong thiết lập DPLS, đơn vị tính toán chính là bước thăm dò *M* và chúng tôi muốn thể hiện MH của các tính toán DAG tùy ý so với *M*. Phần tinh tế của mô hình này là thể hiện lợi thế *€*^ của đối thủ so với các bên trung thực như một chức năng của khả năng "nghiền nát" (Grind) tính ngẫu nhiên của tính toán DAG cũng như tận dụng bất kỳ lợi thế nào thu được từ việc quan sát các bước được công bố trước đó trong tính toán.

(III) Giao thức Blockchain dựa trên PoUW. Ở cấp độ cao, giao thức hoạt động bằng cách yêu cầu các bên đăng các trường hợp cho các vấn đề quan tâm trong sổ cái. Đồng thời khóa các khoản tiền có giá trị bằng Token gốc của sổ cái để khuyến khích các thợ đào tìm cách giải quyết chúng. Việc duy trì chuyển đổi Blockchain liên quan đến việc thực hiện các bước của thuật toán DPLS cho các trường hợp trong sổ cái và được thưởng cho điều đó với tầm nhìn xa. Chúng tôi nhấn mạnh rằng việc giải quyết các trường hợp đó trực tiếp (hoặc đăng các

trường hợp đã được giải trước) sẽ không giúp mở rộng sổ cái. Người đặt vấn đề có thể tiếp tục cấp vốn cho một tính toán DPLS cụ thể bất cứ khi nào tiền của nó cạn kiệt.

Nền tảng của giao thức là quy trình PoUW hoạt động theo ba giai đoạn. Trong giai đoạn Pre-Hash (trước khi băm), một chuỗi ngẫu nhiên được tạo thông qua Hash và kiểm tra xem giá trị Hash nhỏ đã đạt được như trong PoW tiêu chuẩn hay chưa. Chuỗi này sẽ tạo thành hạt giống (Seed) ngẫu nhiên cho bước thăm dò M của DPLS và sẽ là yếu tố cần thiết để kiểm soát các cuộc tấn công "nghiền nát". Khi giai đoạn thăm dò kết thúc, bước Post-Hash (sau khi băm) xác định bằng một truy vấn Hash duy nhất liệu giá trị kết quả có đủ điều kiện là PoUW hay không. Việc "kẹp" (Sandwiching) M giữa hai hàm Hash nhỏ này là điều cần thiết để bảo mật vì nó buộc thợ đào tấn công phải gieo hạt tính toán với một hạt giống được chọn ngẫu nhiên, một Block thành công chỉ có thể được phát hành sau khi bước thăm dò M hoàn tất. Tuy nhiên, nếu chúng ta áp dụng ý tưởng này không đúng cách, có hai nhược điểm lớn: thứ nhất, một số bước thăm dò hữu ích sẽ bị lãng phí, vì chúng sẽ không dẫn đến một Block. Thứ hai, điều chỉnh độ cứng của quá trình sản xuất Block (đó là cần thiết cho bảo mật Blockchain) sẽ ảnh hưởng đến tính hữu ích (vì các thợ đào sẽ tốn quá nhiều tiền để cố gắng tạo ra các Hash nhỏ như vậy).

Chúng tôi giải quyết những vấn đề này bằng hai cơ chế. Đầu tiên, tận dụng chức năng tính điểm g, chúng tôi yêu cầu các thợ đào công bố giá trị tốt nhất mà họ đã tạo ra dựa trên tất cả các nỗ lực Post-Hash; bằng cách này, tiến trình tính toán DPLS không bị mất. Lưu ý rằng việc đi chệch khỏi chiến lược này có thể chỉ ảnh hưởng đến tính hữu ích, còn tính bảo mật của giao thức vẫn được duy trì để chống lại bất kỳ sự sai lệch Byzantine nào. Thứ hai, bằng cách điều chỉnh cơ chế 2 cho 1 (two-for-one) của PoW [24], cho phép tạo ra hai loại Block chỉ với một lần Hash duy nhất: hoặc là "khối đầu vào" (Input Block), trong trường hợp đó, nó được chèn vào Blockchain dưới dạng giao dịch hoặc "khối xếp hạng" (Ranking Block) mở rộng Blockchain và chứa bất kỳ số lượng Input Block nào. Sử dụng cơ chế phân tách này, chúng tôi có thể duy trì tiến độ ổn định của tính toán DPLS và điều

chỉnh độ cứng cơ bản của Ranking Block một cách độc lập. Tính chất quan trọng mới nổi ở đây là, khi càng nhiều thợ đào tham gia giao thức, thì tính toán DPLS sẽ tăng tốc tương ứng. Trong khi việc sản xuất Ranking Block có thể được duy trì ổn định theo yêu cầu đối với tính bảo mật của giao thức Blockchain cơ bản. Bằng cách này, càng nhiều trường hợp vấn đề hữu ích trong thế giới thực được gửi đến hệ thống (bằng chứng là số tiền tăng lên bị khóa với mỗi trường hợp và sự đánh giá cao Token gốc của nền tảng), thì công cụ DPLS sẽ càng có nhiều sức mạnh tính toán hơn để giải quyết chúng.

Chúng tôi chứng minh giao thức của mình an toàn theo kiểu giả định "đa số trung thực" tiêu chuẩn gợi nhớ đến phân tích giao thức Bitcoin, trong đó khoảng cách từ 1/2 phụ thuộc, trong số các tham số khác, cũng như lợi thế *€*^ của MH (chúng tôi lưu ý rằng ngay cả khi *€*^ = 1, tức là MH hoàn toàn sụp đổ, giao thức vẫn an toàn với giới hạn gần = 1/4).

(IV) Các chỉ số đo lường mức độ hữu ích. Nói tóm lại, giao thức Blockchain của chúng tôi có thể được coi như một bộ giải DPLS phi tập trung. Điều này gợi ý hai số liệu bổ sung sau đây để đo lường mức độ sử dụng hiệu quả của nó. Chỉ số đầu tiên là xem việc thực thi Blockchain như một công cụ DPLS tốt như thế nào. Điều này có thể được thực hiện bằng cách đo tỷ lệ trên một đơn vị thời gian của số bước mà giao thức Blockchain sử dụng trong tính toán DPLS so với tổng số bước của nó. Chúng tôi gọi chỉ số này là U_{eng}, vì nó có thể được coi là hiệu quả của giao thức Blockchain như một công cụ chạy DPLS. Chỉ số thứ hai là xem các tính toán DPLS hữu ích như thế nào và chúng tôi ký hiệu nó là U_{alg}. Đối với một phân phối nhất định, chúng tôi coi chỉ số này là tỷ lệ giữa số bước dự kiến của thuật toán tốt nhất cho phân phối phiên bản đó chia cho số bước dự kiến mà DPLS thực hiện. Lưu ý rằng việc xác định thuật toán tốt nhất cho một vấn đề có thể không khả thi dựa trên tình trạng kỹ thuật hiện tại. Vậy nên trong trường hợp này, thuật toán tốt nhất có thể được thay thế đơn giản bằng thuật toán đã biết tốt nhất cho vấn đề hiện tại. Kết hợp hai chỉ số trên, chúng ta có thể thu được một thước đo tổng thể về mức độ hữu ích của sản phẩm $U_{eng} \cdot U_{alg}$.

Với hình thức ở trên, đối với giao thức chúng tôi quan sát thấy rằng (i) $U_{eng} \leq 1/2$, điều này bắt nguồn từ thực tế là chúng tôi cân bằng xác suất thành công của Pre-Hash để yêu cầu nỗ lực tương tự như độ phức tạp thời gian trong trường hợp xấu nhất của *M*. Điều này cho phép chúng tôi chứng minh tính bảo mật cho bất kỳ lợi thế *€*^ nào trong giả định MH cơ bản. (ii) U_{eng} sẽ gần bằng =1/2, nếu lợi thế *€*^ cho thấy độ nhạy nhỏ đối với việc "nghiền nát" tăng lên. Chúng tôi lưu ý rằng giới hạn 1/2 có thể được vượt qua bằng cách tính đến độ nhạy của *€*^ và đặt độ khó Pre-Hash một cách thích ứng. Tuy nhiên, hướng thực hiện như vậy sẽ chỉ khả thi nếu chúng ta hạn chế thuật toán thăm dò *M* ở những độ cứng đã được hiểu rõ. Ước tính U_{alg} yêu cầu một số đường cơ sở trong thế giới thực. Chúng tôi khám phá điều này bằng cách triển khai trong công cụ DPLS WalkSAT [55, 35], một thuật toán tìm kiếm cục bộ phổ biến cho các vấn đề về khả năng thỏa mãn và so sánh cách triển khai DPLS với việc chạy WalkSAT một cách riêng lẻ. Việc phân phối cá thể cũng là một yếu tố quan trọng cần xem xét; để minh họa, chúng tôi tập trung vào *Lập kế hoạch Thế giới Block (Blocks World Planning)*, một vấn đề nan giải nổi tiếng trong trí tuệ nhân tạo [32] mà có rất nhiều bộ dữ liệu công khai. Sử dụng WalkSAT làm đường cơ sở, chúng tôi cho thấy việc triển khai DPLS đơn luồng hoạt động khá tốt so với WalkSAT, đầu tư nhiều gấp đôi các bước tính toán, tức là một cái gì đó tương đương với ước tính $U_{alg} \approx 1/2$. Các kết quả tương tự thu được từ các thí nghiệm bổ sung tạo ra độ lệch đối nghịch và ví dụ của sự song song hóa.

Các kết quả trên là bằng chứng cho tính hữu ích không thể bỏ qua trong thế giới thực của giao thức Blockchain dựa trên PoUW. Chúng tôi dự đoán việc nghiên cứu sâu hơn về công cụ Blockchain DPLS như một công cụ giải quyết tối ưu hóa sẽ là một hướng nghiên cứu thú vị từ góc độ thuật toán. Có một khía cạnh quan trọng khác của việc sử dụng giao thức Blockchain như một bộ giải DPLS: tối ưu hóa được thực hiện một cách cộng tác theo cách có thể xác minh công khai. Tùy thuộc vào nhiệm vụ, khả năng xác minh công khai có tính hữu ích nội tại. Đây có thể được coi là cái giá mà hệ thống phải trả cho tỷ

lệ còn lại 1 - $U_{eng} \cdot U_{alg}$. Ví dụ: các nhiệm vụ tối ưu hóa như lên lịch thi đấu thể thao hoặc các vấn đề đối sánh khác nhau (ví dụ: phân bổ cư dân đến bệnh viện hoặc đấu giá tần số vô tuyến) có thể được hưởng lợi từ khả năng xác minh công khai; xem Phần 6 để thảo luận và tham khảo thêm.

Công việc liên quan. Ngoài công việc ban đầu được đề cập trong Coin và thiết kế PoUW [16, 38, 23], một số công trình khác đã nghiên cứu khái niệm này. Một dòng công việc được coi là công trình kết hợp trong đó thợ đào có thể lựa chọn giữa việc áp dụng PoW tiêu chuẩn hoặc thực hiện một số tính toán hữu ích tiềm năng [48, 12, 60]. Loe et al. [41], và Dotan et al. [20], và gần hơn với công việc của chúng tôi, Baldominos et al. [8], và Lihu et al. [39], được đề xuất dựa trên PoUW dựa trên các vấn đề tìm kiếm ngẫu nhiên và học máy (Machine-learning). Trong tất cả các cách tiếp cận trước đây, tính bảo mật của hệ thống không được phân tích chặt chẽ. Trong nhiều trường hợp, các cuộc tấn công cụ thể bởi kẻ tấn công trực tiếp tạo ra các trường hợp dễ giải quyết là khả thi.

Ngược lại với những điều trên, một cách tiếp cận bảo mật chính thức đã được thực hiện trong [9] nhưng phiên bản đã xuất bản của tác phẩm đã rút lại kích thước "hữu ích" của bài nghiên cứu gốc. Ngoài ra, việc xây dựng bằng chứng công việc của họ không phù hợp với sổ cái không được phép vì nó không đưa ra bất kỳ phương sai nào trong thời gian hoàn thành câu đố.

Cuối cùng, một số cách tiếp cận thay thế cho vấn đề đang được đề cập trong bối cảnh của chúng tôi là khái niệm "khai thác hợp nhất" (Merged Mining), kỹ thuật được sử dụng trong một số Crypto, trong đó nỗ lực khai thác cho Blockchain có công dụng kép là khai thác Bitcoin và nó rất hữu ích theo nghĩa này; Permacoin [43] trong đó, thông qua các bằng chứng về khả năng truy xuất, kích thước hữu ích đang duy trì một kho lưu trữ tệp công khai; và công việc hữu ích được thực thi thông qua một môi trường thực thi đáng tin cậy [59] trong đó, trái ngược với các giải pháp trên, cần phải tin tưởng hoàn toàn vào một nhà sản xuất phần cứng cụ thể.

Trong phạm vi hiểu biết của mình chúng tôi muốn lưu ý rằng, trước đây không có giao thức Blockchain dựa trên PoUW phi tập trung hoàn toàn nào được công bố cùng với phân tích bảo mật (hoặc tính hữu dụng) kỹ lưỡng.

Tổ chức của bài nghiên cứu. Trong Phần 2, chúng tôi mô tả mô hình tính toán và một số ký hiệu cơ bản. DPLS và khái niệm tính toán vừa phải được trình bày trong Phần 3. Trong Phần 4, chúng tôi trình bày giao thức Blockchain, có tính bảo mật và hữu ích mà chúng tôi phân tích trong Phần 5. Các ứng dụng và kết quả thử nghiệm được đưa ra trong Phần 6, trong khi một số mã Code được trình bày trong Phụ lục.

2. Sơ bộ

Kí hiệu. Với $k \in \mathbb{N}^+$, $[k]$ biểu thị tập $\{1, \ldots, k\}$. Chúng tôi ký hiệu các dãy bằng $(a_i)_{i \in I}$, với I là một tập hợp chỉ số có thể đếm được. Đối với một tập hợp X, $x \leftarrow X$ biểu thị việc lấy mẫu ngẫu nhiên một phần tử từ X một cách đồng nhất. Đối với phân phối $\mathcal{U}$ trên một tập X, $x \leftarrow X$ biểu thị lấy mẫu một phần tử của X theo $\mathcal{U}$. Với $\mathcal{U}_m$, chúng tôi biểu thị phân phối đồng đều trên $\{0, 1\}^m$. Chúng tôi biểu thị một số hàm f là không đáng kể trong λ bởi $f(\lambda) < negl(\lambda)$. Chúng tôi đặt λ biểu thị tham số bảo mật.

Mô hình bảo mật. Chúng tôi áp dụng mô hình tính toán của [25], là một biến thể của mô hình được trình bày trong [24]. Ở đó, tập hợp các bên $\{P_1, \ldots, P_n\}$ đang chạy giao thức được cố định và các bên, môi trường Z, đối thủ A và chương trình điều khiển C điều phối việc thực thi đều được mô hình hóa dưới dạng IRAMs. Đối thủ A đang hoạt động và có thể làm hỏng tối đa t để phá vỡ bảo mật.

Mô hình giao tiếp. Chúng tôi tuân theo mô hình giao tiếp được sử dụng bởi hầu hết các công trình trước đây [51, 7] phân tích các giao thức Blockchain trong cài đặt mật mã, trong đó thời gian là rời rạc và mạng đồng bộ (một

phần). Chi tiết hơn, giao thức tiến bộ trong các vòng và giao tiếp diễn ra thông qua chức năng khuếch tán. Các bên trung thực có thể sử dụng nó để gửi các thông điệp, có thể bị kẻ tấn công trì hoãn một cách thích ứng lên đến Δ vòng, nhưng được đảm bảo rằng tất cả mọi người trong mạng lưới đều nhận được. Thông tin liên lạc không được xác thực, theo nghĩa là chức năng không cung cấp bất kỳ đảm bảo nào về nguồn gốc của các thông điệp đã gửi. Cuối cùng, kẻ tấn công đang gấp rút và cũng có thể chọn gửi thông điệp của riêng mình chỉ đến một tập hợp con của các bên.

Thiết lập. Tất cả các bên đều có quyền truy cập vào *chuỗi tham chiếu chung* (CRS - Common Reference String), được lấy mẫu từ một phân phối có thể lấy mẫu rõ ràng đã biết, được sử dụng để khởi tạo hệ thống đối số không tương tác ngắn gọn (SNARG - Succinct Non-interactive Argument) [30] SNARG = (S, P, V). Lưu ý rằng có một số cách để thiết lập CRS cho SNARG một cách an toàn trong môi trường Blockchain không được phép. Đặc biệt, giả sử khái niệm mạnh hơn một chút về *chuỗi tham chiếu có cấu trúc* (SRS - Structured Reference String) có thể cập nhật [31, 42], cấu trúc của [36] cho phép thu được một chuỗi tham chiếu chung.

Oracle ngẫu nhiên. Các bên có quyền truy cập vào chức năng Oracle ngẫu nhiên (RO – Random Oracle) [10]. Chúng tôi sử dụng cả những vấn đề có độ cứng vừa phải dựa trên RO và không dựa trên RO. Để tranh luận về tính bảo mật, chúng tôi cần phải so sánh chi phí tính toán của chúng. Do đó, chúng tôi giả định một truy vấn tới RO thực hiện các bước tính toán c_H cho cả bên trung thực và kẻ tấn công.

Mô hình ràng buộc. *A* và *Z* có giới hạn cụ thể là $t \cdot c_H$, bước mà họ có thể thực hiện mỗi vòng cũng như giới hạn trên θ về số lượng thông điệp mà họ có thể gửi mỗi vòng.

3. Tìm kiếm cục bộ song song kép

Một cách để thiết kế Blockchain PoUW cho các vấn đề tối ưu hóa là: (i) trước tiên chọn thuật toán tối ưu hóa yêu thích của bạn, sau đó (ii) cố gắng thiết kế một giao thức Blockchain xung quanh nó. Điểm không thuận lợi của cách tiếp cận như vậy là bất kỳ thay đổi nào trong vấn đề tối ưu hóa mục tiêu đều có thể dẫn đến những thay đổi quan trọng đối với hệ thống Blockchain và cơ chế đồng thuận, yêu cầu các bằng chứng bảo mật mới. Thay vào đó, ở đây, chúng tôi áp dụng phương pháp mô-đun. Đầu tiên chúng tôi xây dựng một Blockchain PoUW dựa trên một thuật toán tối ưu hóa chung. Sau đó, với chi phí tối thiểu, khởi tạo nó bằng các tham số cụ thể của vấn đề. Điều này cho phép sử dụng lại phân tích Blockchain để tạo ra các mô tả khác nhau về phương pháp tối ưu hóa.

Trong Phần 3.1, chúng tôi bắt đầu bằng cách đưa ra tổng quan cấp cao về DPLS, thuật toán tối ưu hóa chung mà giao thức Blockchain của chúng tôi đang triển khai theo quan điểm của khách hàng, tức là bỏ qua các chi tiết nội bộ của thuật toán Blockchain. Trong Phần 3.2, chúng tôi mở rộng khái niệm về độ cứng vừa phải của tính toán hữu ích, dựa trên tính bảo mật của giao thức Blockchain.

3.1. Mô tả thuật toán

Tổng quan về DPLS. Khách hàng của giao thức gửi lên Blockchain các vấn đề tối ưu hóa mà họ muốn các thợ đào giải quyết. Mặt khác, các thợ đào chạy thuật toán Tìm kiếm cục bộ song song kép (DPLS) mà chúng tôi giới thiệu để giải quyết những vấn đề này.

Trước tiên, hãy lưu ý rằng việc giải quyết các vấn đề tối ưu hóa lớn có thể đòi hỏi nhiều công việc hơn những gì một Node có thể tính toán được trong quá trình khai thác một Block duy nhất. Do đó, chúng tôi đã chọn DPLS là một *thuật toán phân tán* trong đó kết quả tính toán thu được bằng nhiều lần cập

nhật trạng thái, một số trong số chúng có thể xảy ra đồng thời. Cập nhật đồng thời là nguồn song song đầu tiên của thuật toán song song kép.

Trong cốt lõi của nó, DPLS tìm kiếm không gian giải pháp X bằng cách liên tục khám phá vùng lân cận của một vị trí / điểm hiện đang được chọn, tìm kiếm một điểm lân cận hứa hẹn tiến tới một giải pháp tối ưu. Cụ thể hơn, dựa trên mô tả của một trường hợp vấn đề Λ, DPLS dần dần xây dựng một DAG G ghi lại các vị trí đã được khám phá trong X. Sau đó, một bước thăm dò duy nhất bao gồm việc gọi một thuật toán thăm dò chung M trên G, tạo ra một vị trí mới trong X, với mục tiêu mở rộng G bởi một Node đại diện cho một vị trí mới có chất lượng tốt hơn (được tính toán bằng thuật toán tính điểm g_Λ), do đó tiến hành khám phá.

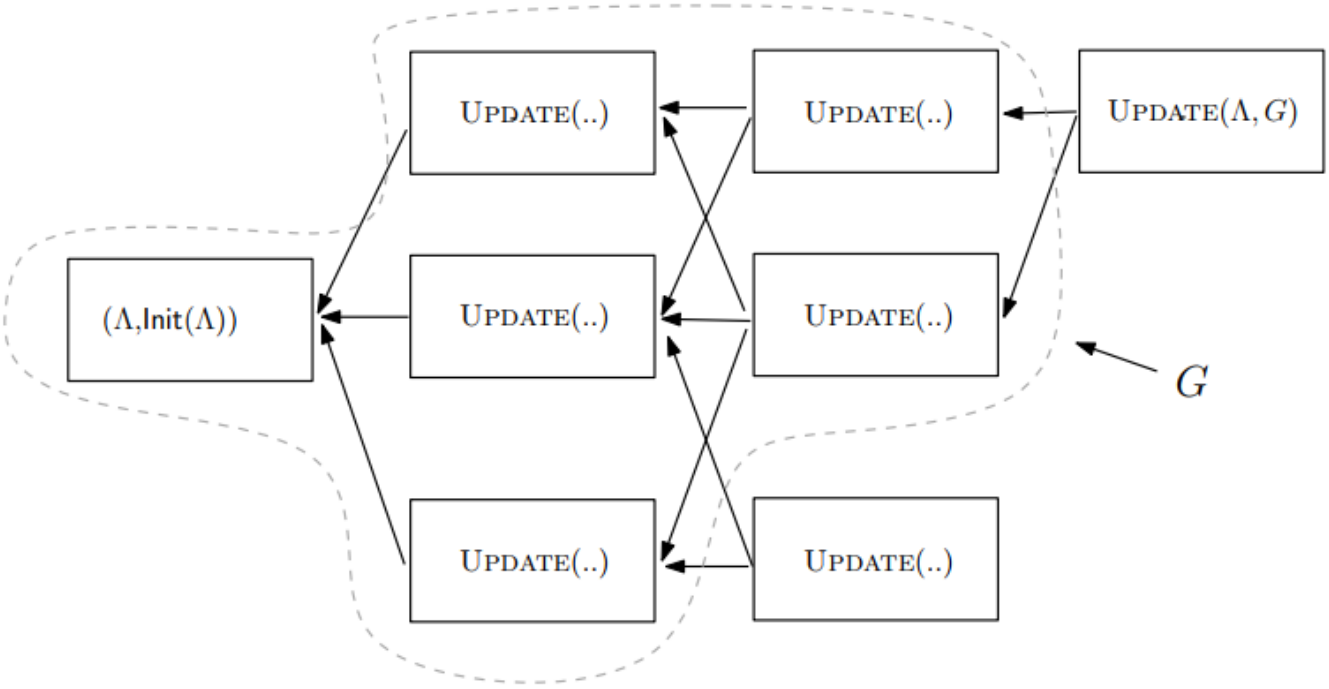
Lưu ý rằng, trong một thực thi tuần tự nghiêm ngặt, đồ thị G "tuyến tính" có thể phù hợp. Tuy nhiên, việc duy trì DAG các vị trí đã khám phá cho phép tìm kiếm cục bộ tổng quát hơn, nhiều chủ đề được khám phá đồng thời bởi các bên khác nhau. Việc thực hiện như vậy được trình bày trong Hình 1.

Khi thuật toán tìm kiếm được phân phối, với nỗ lực giảm thiểu trao đổi và tính toán trước cục bộ, chúng tôi không thể xuất bản mọi bản cập nhật vi mô. Vì lý do này, mỗi bên tính một số lượng lớn các bước thăm dò cục bộ theo lô (Batch), chỉ công bố kết quả thăm dò tốt nhất từ lô. Cuối cùng, thuật toán thăm dò M được tham số hóa bởi một *trạng thái bên trong z* xác định trạng thái chung của lô thực thi, ví dụ: vị trí bắt đầu chung trong G để tập trung tìm kiếm theo lô. Tìm kiếm theo lô là nguồn song song thứ hai của thuật toán song song. Với những điều trên, DPLS được tham số hóa bởi các thuật toán phụ sau:

- Thuật toán khởi tạo Init(Λ): Một thuật toán xác suất lấy đầu vào là mô tả phiên bản Λ và đầu ra là một DAG G.
- Thuật toán tiêu điểm F (Λ, G): Một thuật toán xác suất lấy đầu vào là Λ, G và đầu ra ra là giá trị bên trong chuỗi trạng thái z.
- Thuật toán thăm dò M_Λ (G, z, r): Một thuật toán xác định lấy đầu vào là một DAG G,một trạng thái bên trong z, và một hạt giống r, và đầu ra là

một điểm $x \in X$.

- Thuật toán tính điểm $g_\Lambda(x)$: Một thuật toán xác định lấy đầu vào Λ và $x \in X$, và đầu ra là điểm $y \in \mathbb{R}$ của x.
- Thuật toán kết thúc Finish (Λ, G): Một thuật toán xác định lấy đầu vào Λ, G và đầu ra là 1 nếu thuật toán đã hoàn thành và 0 trong trường hợp khác.



Hình 1: Thực thi thuật toán DPLS. Do các bản cập nhật trạng thái không đồng bộ hóa có thể được tính toán trên một phần của việc thực thi (khung nhìn G).

DPLS được mô hình hóa trong một thiết lập Blockchain. Việc giải quyết vấn đề bắt đầu bằng việc người đặt vấn đề đăng một mô tả ví dụ Λ cùng với đầu ra của Init(Λ) trong Blockchain dưới dạng một giao dịch đặc biệt.[1] Thợ đào làm việc trên một phiên bản như vậy bằng cách chạy quy trình với hàm UPDATE (Thuật toán 1), sử dụng các thuật toán phụ được giới thiệu ở trên. Các đầu ra được tạo ra được gửi lên Blockchain và lần lượt được các bên khác sử dụng để tạo ra các bản cập nhật bổ sung. Thuật toán tìm kiếm kết thúc khi vị từ Finish(Λ, G) bằng 1.

Hàm UPDATE nhận làm đầu vào mô tả phiên bản đã chọn Λ và chế độ xem của bên hiện tại về DAG G. Trạng thái bên trong z, được truyền dưới dạng tham số trong hàm UPDATE, được tạo bằng thuật toán F(Λ, G). Trong khi số k của các lệnh gọi khác nhau

của *M* được phân phối theo phân bố hình học, với các tham số chính xác của phân phối do người thiết kế giao thức thiết lập. Việc lấy mẫu *k* từ các mô hình phân phối hình học tích hợp nó vào quy trình khai thác công việc hữu ích trong đó mỗi phép tính của *M* đủ điều kiện cho sản xuất Block với xác suất p_2. Để xuất bản cập nhật trạng thái, thợ đào tìm một Block. Sau khi *k* được cố định, nhiều hạt giống $(r_i)_{i\in[k]}$ được lấy mẫu ngẫu nhiên và thuật toán $M(G, z, r_i)$ được gọi *k* lần, với kết quả cho điểm tốt nhất (theo hàm *g*) được đưa ra bằng cách sử dụng hàm UPDATE.

Thuật toán 1: Thủ tục cập nhật trạng thái.

funtion UPDATE (Λ, G)	
z ← F (Λ, *G*)	-Tính toán trạng thái bên trong
k ← $G_{eom\ (p2)}$	-Mẫu từ hình học
$(ri)_{i\in[k]}$ ← U^{km}	-Mẫu đồng nhất
S: = {(z, r_i ,x_i) | x_i : = *M* (*G*, z, r_i), i € [k]}	-Gọi *M*
(z, r, *x*): = arg max $_{(z, r, x)\in S}$ g(*x*)	-Lựa chọnh tốt nhất
return (z, r, x)	

[1]Để tránh thêm sự thao túng của đối thủ, chúng tôi có thể yêu cầu các bên tự chạy chức năng khởi tạo bằng cách sử dụng Coin được tạo ra bằng cách Hash Block bao gồm trường hợp sự cố đã đăng.

Một ví dụ. Chúng tôi trình bày một biến thể DPLS của thuật toán WalkSAT cổ điển [55, 35] cho bài toán SAT. Đầu tiên, chúng tôi mô tả thuật toán WalkSAT ban đầu. Bắt đầu từ một số quan điểm ban đầu, ở mỗi bước, WalkSAT chọn một biến để sắp xếp (Thuật toán 2) như sau: Với quy cách hiện tại, một trong các mệnh đề không thỏa mãn được chọn ngẫu nhiên. Đối với mỗi biến liên quan đến mệnh đề, điểm được tính bằng số mệnh đề sẽ bị phá vỡ (nghĩa là chuyển từ phù hợp sang chỉnh sửa không thỏa mãn) nếu biến đã chọn được bổ sung. Nếu tồn tại các biến có cấp 0, thì một trong số chúng sẽ được chọn ngẫu nhiên và sắp xếp. Nếu không, một biến được chọn (và sắp xếp) một cách ngẫu nhiên, với xác suất *wp* đến từ mệnh đề đã chọn và với xác suất 1 - *wp* đến từ các biến có cấp tốt nhất. Thuật toán Walk tiếp tục cho đến khi tìm ra giải pháp (Thuật toán 3), hoặc một số điều kiện khác được đáp ứng, ví dụ: đạt tới giới hạn trên của tổng số lần sắp xếp. Nếu không tìm thấy giải pháp nào, thuật toán có thể được khởi động lại từ một số điểm khác trong không gian giải pháp.

Trong biến thể DPLS, mô tả cá thể Λ mã hóa mô tả của cá thể SAT, tức là số lượng biến và các mệnh đề khác nhau, với không gian giải pháp X bằng với các cấu trúc có thể có của các biến SAT. Để tận dụng mức độ song song đầu tiên, Init(Λ) xuất ra một số hàm ban đầu khác nhau trong X; trong mỗi lần gọi của hàm UPDATE, thợ đào chọn ngẫu nhiên vị trí/cấu hình trong G để làm việc và mã hóa thông tin này bằng z. Với điều kiện này, thuật toán thăm dò $M(G, z, r)$ tương đương với việc chạy WalkSAT với số lần sắp xếp cố định. Lưu ý rằng quy tắc bắt đầu giống nhau đối với các lần chạy khác nhau của M trong một lần gọi hàm UPDATE duy nhất, cho phép các thợ đào tập trung vào việc tìm kiếm. Mặt khác, tính ngẫu nhiên được sử dụng bởi các lệnh gọi WalkSAT khác nhau đến từ các hạt tương ứng (r), dẫn đến việc khám phá các điểm khác nhau trong không gian giải pháp. Để chọn điểm tốt nhất trong số các điểm này, g đếm số lượng các mệnh đề phù hợp trong các hàm kết thúc tương ứng có độ sâu tối đa trong DAG. Do đó, hàm UPDATE cho kết quả đầu ra cấu hình tối đa hóa g. Sau đó có thể sẽ được một thợ đào khác sử dụng làm điểm bắt đầu của một lần chạy hàm UPDATE khác. Thuật toán chấm dứt sau khi một số lượng bản cập nhật trước đó đã được đăng. Chúng tôi hướng người đọc đến Phần 6 để đánh giá thử nghiệm hiệu suất của thuật toán này.

Thuật toán 2: Chức năng chọn biến của WalkSAT. Nó được tham số hóa bởi xác suất *wp*, tập hợp các mệnh đề *C* và hàm cấp độ $grade_{A,C}(x)$, đếm số mệnh đề trong *C* sẽ bị phá vỡ nếu biến *x* được xếp theo hàm A.

1: **funtion** PickVariable (*A*, *C*)
2: c ←{c | c là mệnh đề không thỏa mãn trong *C*}
3: s: = min $\{grade_{A,C}(x) \mid x$ € Var (c)$\}$ Var xuất ra biến trong c
4: **If** s=0 **then**
5: *V*: = {x € Var (c) | $grade_{A,C}(x)$ = s}
6: **else**
7: với xác suất *wp* thì *V*: = Var (c)
8: ngược lại thì V: = {xVar (c) | gradeA, C (x) = s}
9: *x* ← *V* *x* là biến được chọn để sắp xếp
10: **return c**

Thuật toán 3: Thuật toán WalkSAT. Nó được tham số hóa bởi hợp thức *A*, tập hợp các mệnh đề *C*, và một giới hạn *m* trên số lần thử nghiệm. Tính ngẫu nhiên được WalkSAT sử dụng được chuyển vào tham số *r*.

1: **funtion** WalkSAT (*A*, *C*, *m*; *r*)
2: i: = 0, *f*: = false
3: **while** ((f = false) ∧ (i <m)) **do**
4: x ← PickVariable (*A*, *C*)
5: A: = *flip* (*A*, *x*) Sắp xếp biến *x* trong *A*
6: f: = (A thỏa mãn tất cả các mệnh đề *C*)
7: *i* = *i* + 1
8: **if** *f* = false **then return**
9: **else return** *A*

Tiếp theo, chúng tôi đưa ra mô tả chi tiết về tính toán DAG liên quan đến thuật

toán, cũng như phần còn lại của các hàm liên quan đến thuật toán DPLS: Init, F, Finished (Thuật toán 4). Đối với tính toán DAG, các tập *X*, *Z* đại diện cho tập hợp các cấu hình có thể có của cá thể SAT, trong khi $g(x)$ bằng số mệnh đề được thỏa mãn bởi hàm *x*. Hàm Init tạo ra nhiều cấu hình ban đầu. Mỗi cấu hình được sử dụng như điểm bắt đầu của một thuận toán Walk khác nhau. Một bên bây giờ chọn ngẫu nhiên một điểm bắt đầu A_z bằng cách sử dụng hàm F, sau đó *M* chọn điểm kết thúc tốt nhất kéo dài A_z và ở độ sâu tối đa, để sử dụng làm điểm bắt đầu trong thủ tục WalkSAT được gọi bởi *M*. Thuật toán WalkSAT chỉ được chạy cho một số lần sắp xếp giới hạn (tham số *m*). Tiếp theo lược đồ được trình bày trong Phần 3, quá trình hàm UPDATE gọi *M* nhiều lần, chỉ với câu lệnh tốt nhất trong số các lệnh gọi này được xuất ra. Thuật toán kết thúc khi tìm thấy một điều khoản thỏa mãn tất cả các mệnh đề.

3.2. Tính toán DAG vừa phải

Trong DPLS, phần lớn công việc được dành để chạy thuật toán thăm dò *M*. Do đó, mức độ an toàn sẽ dựa trên độ cứng vừa phải của phép tính này. Bây giờ chúng tôi mô tả chi tiết cú pháp của *M* và các thuộc tính bảo mật liên quan cần thiết để sử dụng trong giao thức PoUW.

Như đã giải thích trước đó, một khía cạnh quan trọng là các cập nhật trạng thái trong DPLS được thực hiện theo cách phân tán và không có nhiều sự phối hợp. Hơn nữa, các thông số tính toán được thực hiện sẽ có thể bị kẻ tấn công cản trở, nghĩa là họ có thể cố gắng gửi một vấn đề khách hàng cần được giải quyết,

Thuật toán 4: Các thuật toán WalkSAT được sửa đổi. Hàm $\text{depth}_G(v)$ xuất ra độ sâu của *v* trên DAG *G*.

1: **funtion** Init(Λ; *G*)
2: $(A_i)_i \leftarrow X^l$ -Mẫu điểm *l* từ không gian giải pháp
3: **return** $\{(i; \perp; A_i)\}_i$ -Trả lại DAG với số điểm ban đầu
4:
5: **funtion** F(Λ; *G*)
6: $z \leftarrow [l]$ -Điểm tiếp theo được chọn sẽ mở rộng A_z
7: **return** *z*
8:
9: **funtion** M(Λ; *G*; *z*; *r*)

10: C; $m := \Lambda$ -Đọc mô tả phiên bản

11: S := …

12: {x'| ∃r' : ∄r'', x'': (z, r', x''), (z, r'', x'') ∈ V (G)…

13: ….^ $depth_G$ (z; r'; x'') < $depth_G$ (z; r''; x'')}

-Chọn các điểm độ sâu tối đa mở rộng A_z

14: x := arg $\max_{x \in S}$ g(x) -Lựa chọn tốt nhất

15: A' ← WalkSAT(x; C; m; r) -Chạy m lần sắp xếp

16: **return** A'

17:

18: **funtion** Finished(Λ; G)

19: C; $m := \Lambda$

20: f := (∃(z; r; x) ∈ V(G) : x thoả mãn tất cả mệnh đề C)

21: **return** f

---

chỉ với mục đích lật đổ giao thức Blockchain cơ bản. Vì tính bảo mật của Blockchain phụ thuộc vào độ cứng các tính toán riêng lẻ của *M,* chúng tôi phải đảm bảo rằng chúng vẫn ở mức độ cứng vừa phải ngay cả khi các tham số được chọn một cách độc hại.

Dựa trên những hạn chế này, chúng tôi áp dụng cấu trúc DAG cho các phép tính của *M*, trong đó mỗi phép tính tương ứng với một đỉnh trên DAG và phụ thuộc vào nhiều đỉnh trước đó. Khái niệm của chúng tôi có thể được xem là sự tổng quát của mô hình tính toán lặp lại [11, 26], trong đó mỗi phép tính phụ thuộc vào một đỉnh duy nhất. Các đỉnh mới được tạo dựa trên chế độ xem hiện tại của DAG, một chuỗi trạng thái bên trong và một hạt giống không thể đoán trước. Như đã giải thích trước đó, trạng thái bên trong cho phép các bên tập trung vào công việc trong bối cảnh của DPLS. Trong khi hạt giống tính toán ngẫu nhiên để buộc đối phương thực hiện công việc phức tạp trong trường hợp trung bình. Trái ngược với việc có thể chọn các phiên bản "rẻ" để đạt được lợi thế trong sản xuất Block. Tiếp theo, chúng tôi chính thức giới thiệu khái niệm tính toán DAG.

Định nghĩa 1. (Tính toán DAG/bản ghi). Một đặc điểm tính toán DAG là một chuỗi các mô tả cá thể $I = (\Lambda_\lambda)_\lambda$. Đối với mọi giá trị của tham số bảo mật $\lambda \in \mathbb{N}$, một mô tả thực thể Λ:

1. Một tập hợp hữu hạn, không rỗng tập Z (trạng thái bên trong);

2. Một tập hợp hữu hạn, không rỗng X (đầu ra);

3. Một mối quan hệ R được mô tả dưới đây.

Một *bản ghi* của phép tính DAG Λ tương ứng với DAG G được gắn nhãn trong đó mỗi đỉnh $u \in V(G)$ được gắn nhãn với một bộ $(z^{(u)}, r^{(u)}, x^{(u)}) \in Z \times \{0, 1\}^{\lambda} \times X$ (các cạnh không có nhãn). R liên quan đến bản ghi với các bộ ba trong $Z \times \{0, 1\}^{\lambda} \times X$. Cho một bản ghi G, một đỉnh $u \in V(G)$ là *R-compliant*[2] nếu nó không có cạnh đến hoặc nó giữ $(((z^{(ui)}; r^{(ui)}; x^{(ui)}))i; (z^{(u)}; r^{(u)}; x^{(u)})) \subseteq R$, trong đó $\{u_i\}_i$ là tập hợp các đỉnh bắt đầu của các cạnh tới. Chúng ta nói rằng G là R-compliant nếu tất cả các đỉnh của nó đều như vậy. Chúng ta viết $\Lambda [Z, X, R]$ để chỉ ra rằng Λ xác định Z, X, R như trên.

Về cơ bản, R mô tả cách một tập các đỉnh, tương ứng với các phép tính của M, có thể được mở rộng bằng một phép tính mới. Bằng ứng dụng đệ quy, R phù hợp để chính thức hóa khái niệm về một bảng điểm R-compliant của một phép tính DAG. Tính toán DAG cũng cung cấp hai thuật toán. Với mục đích này, chúng tôi yêu cầu rằng các mô tả cá thể, cũng như các phần tử của tập Z, X, có thể được mã hóa duy nhất dưới dạng chuỗi Bit có độ dài đa thức tính bằng λ. Tất cả các thuật toán được tham số hóa bởi $\Lambda[Z, X, R]$:

- Thuật toán *xác minh* $V_{\Lambda}(G)$: Một thuật toán xác định lấy đầu vào là bản ghi G và đầu ra là 1 nếu nó là R-compliant và là 0 trong trường hợp khác; và
- Thuật toán *thăm dò* $M_{\Lambda}(G, z, r)$: Một thuật toán xác định lấy đầu vào là một DAG G, một bên trong trạng thái z, một hạt giống r, và đầu ra là một điểm $x \in X$.

Để đơn giản, chúng tôi sẽ bỏ qua việc viết Λ như một tham số của V, M khi nó

[2]Chúng tôi áp dụng một thuật ngữ tương tự như mô hình dữ liệu thử nghiệm (PCD) [13].

rõ ràng từ ngữ cảnh. Hơn nữa, chúng ta giả sử rằng M đúng, tức là, đối với $x \leftarrow M(G, z, r)$ trong đó G là R-compliant, điều đó cho rằng nếu chúng tôi thêm một đỉnh u trên G với nhãn (z, r, x) được kết nối với tất cả các đỉnh khác, khi đó bản ghi kết quả là R-compliant. Chúng tôi biểu thị việc mở rộng một DAG có nhãn G với một Node mới có nhãn (z, r, x) và được kết nối với tất cả các Node khác bằng $G \oplus (z, r, x)$ và hợp nhất hai bản ghi G, G' bằng $G \cup G$'.

Độ cứng vừa phải. Tiếp theo, chúng tôi giới thiệu khái niệm độ cứng vừa phải (MH) cho các tính toán DAG. Chúng tôi xây dựng khái niệm dựa trên những ý tưởng được tìm thấy trong [25, 26]. Vì chúng tôi muốn xây dựng một giao thức có thể giải quyết nhiều vấn đề tối ưu hóa, MH được biểu thị bằng một họ các tính toán DAG (trên mỗi mức tham số bảo mật), mỗi giao thức tương ứng với một phiên bản khác nhau của thuật toán DPLS. Mặc dù giao thức cho phép các thuật toán thăm dò có độ phức tạp theo thời gian khác nhau, nhưng để đơn giản, chúng tôi giả định rằng tất cả chúng đều có độ phức tạp xấp xỉ trong *trường hợp xấu nhất*, tức là, chúng tôi tính $t\hat{}$ bằng $\max_{\Lambda, G, z, r}\{\text{Steps}_{M\Lambda}(G, z, r)\}$.

Ở mức độ cao, chúng tôi yêu cầu thời gian cần thiết để tạo ra một số đỉnh mới nhất định trong DAG, tỷ lệ với số lượng của chúng cũng như $t\hat{}$, độ phức tạp trong trường hợp xấu nhất của M. Chi tiết hơn, trong thử nghiệm bảo mật, kẻ tấn công có quyền truy cập vào ba Oracle $\mathcal{O}$, $\mathcal{M}$, $\mathcal{V}$. Mục tiêu của nó là tính m đỉnh mới cho các hạt được tạo ngẫu nhiên từ Oracle $\mathcal{O}$ trong ít hơn $(1 - \epsilon\hat{}) \cdot mt\hat{}$ bước, tạo ra lợi thế của kẻ tấn công so với M. Kẻ tấn công được phép truy vấn Oracle $\mathcal{O}$ nhiều hơn m lần và có thể sử dụng Oracle $\mathcal{M}$ và $\mathcal{V}$ để mô phỏng các đỉnh được tính toán trung thực mới và xác minh xem tính toán DAG có là R-compliant tương ứng hay không. $\epsilon\hat{}$ được tham số hóa bởi tỷ lệ truy vấn tương ứng $q\mathcal{O}/m$, $q\mathcal{M}/m$, $q\mathcal{V}/m$ để tạo ra lợi thế đối đầu có thể có. Hãy lưu ý rằng các

Oracle $\mathcal{M}$ và $\mathcal{V}$ được cung cấp để hỗ trợ thành phần;[3] Cuối cùng, chúng tôi yêu cầu thuộc tính giữ với xác suất áp đảo và đối với m lớn hơn một số tham số $k\hat{}$.

Định nghĩa 2. Cho $I = ((\Lambda_{\lambda,i})_i)_\lambda$ là một họ các tính toán DAG. I là $(t\hat{}, \epsilon\hat{}, k\hat{})$ – độ cứng vừa phải (MH) nếu với bất kỳ PPT RAM $A = (A_1, A_2)$, $\lambda \in \mathbb{N}$, và tất cả các đa thức lớn $m \geq k\hat{}$, điều đó cho thấy kẻ tấn công thắng với xác suất $negl(\lambda)$trong $\text{Exp}^{MH}_{A,I,\epsilon\hat{}t\hat{}}(1^\lambda, m)$

$$\left\{\begin{array}{l} st \leftarrow \mathcal{A}_1(1^\lambda); ((\Lambda_i, G_i, z_i, r_i, x_i))_{i\in[m]} \leftarrow \mathcal{A}_2^{\mathcal{O},\mathcal{V},\mathcal{M}}(st); \\ b_1 := \mathsf{Steps}_{\mathcal{A}_2^{\mathcal{O},\mathcal{M},\mathcal{V}}}(1^\lambda, st) < (1 - \hat{\epsilon}(\frac{q_{\mathcal{O}}}{m}, \frac{q_{\mathcal{V}}}{m}, \frac{q_{\mathcal{M}}}{m}))m \cdot \hat{t}; \\ b_2 := \bigwedge_{i=1}^{m}((G_i, z_i, r_i) \in Q_{\mathcal{O}} \wedge V_{\Lambda_i}(G_i \oplus (z_i, r_i, x_i)) = 1); \\ \text{ret } b_1 \wedge b_2 \end{array}\right\}$$

nơi các truy vấn $q_{\mathcal{O}}$ được thực hiện cho Oracle

$$\mathcal{O}(\Lambda, G, z) = \left\{r \leftarrow \{0,1\}^\lambda;\ \text{return } r\right\},$$

các truy vấn $q_{\mathcal{V}}$ được thực hiện cho Oracle

$$\mathcal{V}(\Lambda, G) = \left\{\text{if } V_\Lambda(G) = 0,\ \text{then return } 0,\ \text{else return } 1 \quad\right\},$$

[3]Trong cài đặt Blockchain, đối thủ nhìn thấy các Block do các bên khác tạo ra, được Oracle $\mathcal{M}$ mô phỏng và gửi ra các Block mà các bên khác có thể bỏ hoặc chấp nhận tùy thuộc vào việc chúng có hợp lệ hay không, được Oracle $\mathcal{V}$ mô phỏng.

Và các truy vấn q$\mathcal{M}$ được thực hiện cho Oracle

$$\mathcal{M}(\Lambda, G, z) = \left\{ \begin{array}{l} \text{if } V_\Lambda(G) = 0, \text{ then return } \perp \\ \text{else, } r \leftarrow \{0,1\}^\lambda; \text{ return } (r, M_\Lambda(G, z, r)) \end{array} \right\}.$$

Như đã gợi ý ở trên, chúng tôi yêu cầu tính toán MH cũng mạnh mẽ để chống lại các cuộc tấn công "nghiền nát" lợi dụng hạt giống của Oracle $\mathcal{O}$. Kẻ tấn công có thể lấy mẫu nhiều hạt từ $\mathcal{O}$, và chỉ chọn thực hiện phép tính đối với những hạt dễ dàng hơn trong số chúng. Khả năng bảo mật chống lại các cuộc tấn công như vậy chỉ tập trung vào giới hạn trên của tốc độ tăng tốc mà đối thủ có được bằng cách thực hiện thêm các truy vấn tới $\mathcal{O}$.

Vì kẻ tấn công luôn có thể thực hiện tính toán DAG cho hạt giống được tạo bởi $\mathcal{O}$ trong $t\hat{}$ bước, các truy vấn bổ sung tới $\mathcal{O}$ chỉ có thể tăng tốc A lên tối đa $t\hat{}$ bước, tức là với ≥ 0

$$\begin{aligned} &(1 - \hat{\epsilon}(1, b, c))m\hat{t} \leq (1 - \hat{\epsilon}(1 + a, b, c))m\hat{t} + ma\hat{t} \qquad (1) \\ &\Leftrightarrow \hat{\epsilon}(1 + a, b, c) \leq \hat{\epsilon}(1, b, c) + a. \end{aligned}$$

Điều này thể hiện bất kỳ tính toán DAG nào đều có khả năng chống lại các cuộc tấn công "nghiền nát" liên quan đến thời gian chạy trong trường hợp xấu nhất của M. Chúng tôi chính thức phát biểu điều này trong bổ đề sau.

Bổ đề 3. Cho I là một họ các tính toán DAG là ($t\hat{}$, $\epsilon\hat{}$, $k\hat{}$) - MH. Khi đó, với bất kỳ hàm $\bar{\epsilon}$ nào với bất kỳ $a \geq 0$, nó cho rằng $\bar{\epsilon}$ (1 + a, b, c) ≤ $\epsilon\hat{}$ (1, b, c) + a, I là ($t\hat{}$, $\bar{\epsilon}$, $k\hat{}$) - MH.

Chứng minh. Để mâu thuẫn, giả sử rằng I không phải là ($t\hat{}$, $\bar{\epsilon}$, $k\hat{}$) - MH. Điều này

thể hiện tồn tại một kẻ tấn công $A = (A_1; A_2)$, $m \geq k$ và $a \geq 0$, am là đa thức lớn trong λ, sao cho A thắng trong $\text{Exp}^{\text{MH}}_{A,I,}$ $\epsilon t(1^{\lambda}, m)$ với xác suất không đáng kể. Vì tổng số truy vấn tới Oracle $\mathcal{O}$ bị giới hạn bởi một đa thức, nên bằng một đối số trung bình, nó cho rằng tồn tại một số $a \geq 0$ sao cho sự kiện A thắng trong $\text{Exp}^{\text{MH}}_{A,I,}$ $\epsilon t(1^{\lambda}, m)$ và số truy vấn đến $\mathcal{O}$ chính xác là $(1 + a^{-})m$ xảy ra với xác suất không đáng kể. Gọi a^{-} là giá trị nhỏ nhất như vậy. Nếu $a^{-} = 0$, thì A có thể được sử dụng để phá vỡ trực tiếp thuộc tính (t, ϵ, k) - MH của I. Nếu không, chúng ta sử dụng A để tạo ra một đối thủ $A' = (A'_1; A'_2)$ phá vỡ thuộc tính (t, ϵ, k) - MH, như được mô tả trong đoạn sau.

A'_1 hoạt động chính xác như A_1 trong giai đoạn đầu của thí nghiệm và chuyển trạng thái được tạo ra thành A'_2. A'_2 chạy bên trong A_2. Truy vấn Oracle $\mathcal{M}$, $\mathcal{V}$ được tạo bởi A_2 được trả lời bởi A'_2 bằng cách sử dụng các Oracle thích hợp. Mặt khác, chỉ m truy vấn đầu tiên đến Oracle $\mathcal{O}$ do A_2 thực hiện được trả lời bằng Oracle $\mathcal{O}$ trong khi tất cả các phản hồi tiếp theo cho các truy vấn tới $\mathcal{O}$ được mô phỏng bằng A'_2.

Trong trường hợp A_2 thành công và tạo ra các nhân chứng hợp lệ cho m trong số các truy được yêu cầu. A'_2 thu thập các truy vấn chỉ được mô phỏng và tự giải quyết các vấn đề tương ứng của chúng bằng cách sử dụng bộ giải chuẩn M. Cuối cùng, nó xuất ra các nhân chứng cho m truy vấn được thực hiện cho $\mathcal{O}$. Nếu không, nó xuất ra $\perp$.

Bây giờ chúng tôi phân tích xác suất chiến thắng của A'. Đầu tiên, hãy lưu ý rằng A'_1 chạy trong thời gian đa thức và tạo ra kết quả chính xác giống như A_1. A'_2 chạy bộ giải M ở hầu hết các thời điểm $a^{-}m$, vì nhiều nhất là $a^{-}m$ trong số các truy vấn hỏi $\mathcal{O}$ bởi A_2 được mô phỏng. Theo giả định, việc thực hiện A_2 tốn ít hơn $(1 - \epsilon (1 + a^{-}, b, c))mt < (1 - \epsilon (1, b, c) - a^{-})mt$ bước; và các lần thực thi bổ

sung của M theo A'_2 có giá hầu hết các bước $a\bar{}mt\hat{}$. Do đó, sử dụng $q\mathbb{O} = m$ truy vấn tới $\mathbb{O}$, các bước tổng thể được thực hiện bởi A'_2 trong trò chơi MH ít hơn

$$(1 - \hat{\epsilon}(1, b, c) - \bar{a})m\hat{t} + \bar{a}m\hat{t} = (1 - \hat{\epsilon}(1, b, c))m\hat{t},$$

thể hiện I không phải là $(t\hat{}, \epsilon\hat{}, k\hat{})$ - MH, do đó kết luận được chứng minh.

Về việc đạt được độ cứng vừa phải. Điều quan trọng cần lưu ý là $(t\hat{}, \epsilon\hat{}, k\hat{})$ - MH, đối với các thông số có thể lý do, là (có thể) không thể đạt được đối với tất cả các họ tính toán DAG. Đặc biệt, để đáp ứng độ cứng vừa phải, chúng tôi không thể cho phép tính toán DAG chung chung.

Để minh họa điều này, hãy xem xét một họ các phép tính DAG cho phép một phiên bản được tạo theo cách sau: một cặp khóa của hoán vị cửa sập (Trapdoor Permutation) được tạo ra, khóa công khai được nhúng trong phiên bản và bước thăm dò được thiết kế như vậy theo cách tính toán hình ảnh trước của một Nonce ngẫu nhiên. Rõ ràng, việc tính toán DAG như vậy sẽ không khó vừa phải theo bất kỳ cách hợp lý nào.

Mặc dù ví dụ này là cực đoan, nhưng nó chứng tỏ rõ ràng sự cần thiết phải hạn chế việc mô tả đặc tính của họ tính toán DAG. Ngược lại, độ cứng vừa phải dường như là một giả định hợp lý cho một tính toán lớn với các thuật toán xác minh và thăm dò đơn giản, ví dụ như đối với các bài toán WalkSAT lớn (vì WalkSAT chỉ liên quan đến các bước tính toán rất cơ bản). Kẻ tấn công bây giờ vẫn có thể tạo ra các vấn đề để cố gắng đạt được lợi thế tính toán trong tính toán DAG, nhưng việc ngẫu nhiên hóa có thể giúp giảm thiểu điều này ở một mức độ lớn.

Vẫn cần nghiên cứu thêm về đặc điểm của các họ tính toán DAG với bằng chứng mạnh mẽ về độ cứng vừa phải (theo các tham số hợp lý).

4. Giao thức Blockchain PoUW

Trong phần này, chúng tôi mô tả và chứng minh an toàn cho một giao thức

Blockchain PoUW chung thực hiện thuật toán DPLS.

4.1. Mô tả giao thức

Trước tiên, chúng tôi tóm tắt một số yêu cầu không chính thức mà giao thức phải đáp ứng để đủ điều kiện là một giao thức ứng cử viên cho việc khai thác công việc hữu ích. Sau đó, chúng tôi mô tả giao thức của mình trong khi thúc đẩy các lựa chọn thiết kế theo các yêu cầu. Các yêu cầu được thúc đẩy từ cả hai phía: bảo mật Blockchain và hiệu quả của thuật toán DPLS:

1. Bảo mật Blockchain:

(a) Không "nghiền nát": Kẻ tấn công không thể đạt được lợi thế khai thác bằng các bước thăm dò "hái anh đào" (Cherry-picking) có độ phức tạp thấp.

(b) Khả năng phục hồi trước tính toán: các trường hợp vấn đề không thể được tạo ra một cách bất lợi để đối thủ có được quyền truy cập vào quá trình sản xuất Block nhanh hơn. Việc tính toán trước khi thấy phần đầu của chuỗi được mở rộng không thể góp phần vào việc tính toán PoUW tương ứng.

(c) Độ khó khai thác có thể điều chỉnh: Độ khó của Block có thể được điều chỉnh theo sức mạnh khai thác được áp dụng bởi mạng lưới.

2. Hiệu quả của thuật toán DPLS:

(a) Cập nhật thường xuyên: Kết quả về các điểm mới đã khám phá được công bố (tương đối) nhanh.

(b) Chi phí nhỏ: Chi phí tính toán của việc tích hợp thuật toán thăm dò M vào PoUW là nhỏ (thể hiện việc khai thác trung thực thực hiện công việc hữu ích).

Kiến trúc cấp cao của giao thức tương tự như Bitcoin, tức là, các Block được liên kết với nhau bằng cách tham chiếu nhau bằng Hash. Trong mỗi vòng, thợ đào chọn chuỗi dài nhất từ chế độ xem và cố gắng mở rộng nó bằng một Block. Hai sửa đổi được áp dụng: PoW tiêu chuẩn được thay thế bằng PoUW và chúng tôi áp dụng PoW 2 cho 1 [24] để phù hợp với các loại Block khác

nhau vì những lý do được giải thích bên dưới. Xem Hình 2 để tham khảo thêm.

Cốt lõi của thuật toán khai thác bao gồm việc áp dụng thuật toán thăm dò *M*, tính toán "phần hữu ích" của PoUW. Để bảo vệ chống lại tính toán trước (Yêu cầu 1a), việc tính toán *M* được bổ sung trước bằng cách Hash Block ứng viên (xem hộp *H* đầu tiên trong Hình 2), do đó việc ngẫu nhiên hóa phép tính được thực hiện bởi *M*. Hơn nữa, tương tự như cơ chế đồng thuận của Nakamoto, Pre-Hash của Block phải nằm dưới mục tiêu ban đầu T_1, để chống lại việc "nghiền nát" các tham số của *M* dẫn đến độ phức tạp tính toán thấp hơn mức trung bình: việc lấy lại mẫu các tham số mới phải đắt hơn độ phức tạp trong trường hợp xấu nhất của *M*.

Theo Yêu cầu 1c, chúng tôi có thể giảm xác suất khai thác thành công xuống dưới xác suất Hash thành công so với T_1 hiện được xác định đầy đủ bởi các đặc tính tính toán của trường hợp vấn đề và không liên quan đến việc tham gia khai thác trong mạng lưới. Một khả năng để giải quyết vấn đề này là hạ thấp hơn nữa mục tiêu T_1 để thực hiện Pre-Hash theo yêu cầu đối với bảo mật sổ cái. Tuy nhiên, điều này sẽ dẫn đến sự mất mát lớn về tính hữu ích, vì các thợ đào sẽ dành phần lớn thời gian của họ để thực hiện Hash. Thay vào đó, chúng tôi yêu cầu thợ đào cung cấp đầu ra của *M* vào một vòng sau khi Hash (Post-Hash) (xem hộp *H* thứ hai trong hình) để quyết định xem Block có đủ điều kiện để xuất bản hay không so với ngưỡng T_3. Ngưỡng thứ hai điều chỉnh độ khó khai thác tổng thể đến mức yêu cầu của phân tích bảo mật để đảm bảo các đặc tính tốt và an toàn của Blockchain. Lưu ý phần bổ sung của quá trình Post-Hash để thích ứng với khó khăn trong khai thác: thợ đào chỉ biết liệu nỗ lực PoUW có thành công hay không sau khi thực hiện *M*, tức là không thể rút ngắn quá trình tính toán để tăng tốc độ tạo Block.

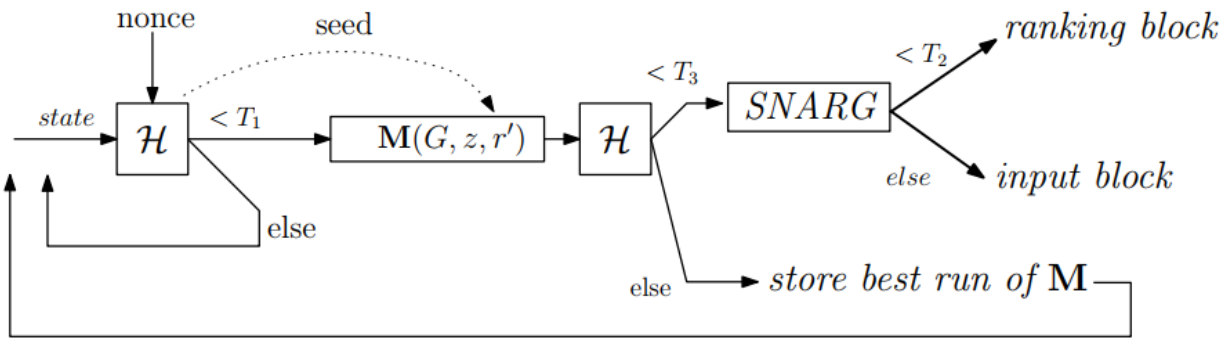
Một thợ đào lặp lại nhiều lần trình tự tính toán của Pre-Hash (so với T_1), công việc hữu ích và một Post-Hash, cho đến khi Post-Hash của một chuỗi nằm dưới T_3, cho phép Block được xuất bản. Để bảo toàn tiến trình, điểm tốt nhất (bằng thuật toán tính điểm g) từ tất cả các chuỗi tính toán gần nhau được lưu trữ để cuối cùng đưa vào một Block trong tương lai sẽ được xuất bản. Lưu ý

rằng việc tìm ra một điểm mới tốt được tách biệt khỏi thành công khai thác, do đó giúp thiết lập Yêu cầu 1b. Hơn nữa, chỉ xuất bản điểm tốt nhất từ một loạt các điểm mới, thay vì tham lam xuất bản tất cả chúng tăng dần, sẽ giúp đáp ứng Yêu cầu 2b.

Xem xét Yêu cầu 2a dưới thông số Bitcoin, chúng tôi không thể yêu cầu một thợ đào chờ đợi bản cập nhật của mình cho đến khi anh ta khai thác một Block. Vì lý do này, chúng tôi kết hợp PoW 2 cho 1 để cho phép xuất bản các loại Block khác nhau, Ranking Block là khối Bitcoin "tiêu chuẩn" có độ khó cao (mục tiêu T_2) và Input Block có độ khó thấp (mục tiêu T_3, tức là phạm vi Hash $T_2 < h \leq T_3$) không phải là một phần của chuỗi nhưng được xử lý giống như các giao dịch để cuối cùng được tham chiếu bởi một Ranking Block.

Bây giờ một thợ đào bao gồm điểm tốt nhất của anh ta được khám phá bất cứ khi nào anh ta chạm vào một trong hai loại Block; và bằng cách đặt độ khó đủ thấp cho Input Block, tỷ lệ cập nhật cho mỗi thợ đào đủ cao để phân phối tiến độ ở các điểm đã khám phá một cách nhanh chóng, trong khi không có tác động đáng kể đến các đặc tính của Blockchain.

Một khối chứa hai điểm được khám phá bằng cách sử dụng M: điểm "chiến thắng" dẫn đến Post-Hash nhỏ và điểm "tốt nhất" được đưa vào để phát triển thuật toán DPLS. Để phù hợp với 2b, chúng tôi giảm thiểu chi phí xác thực Block bằng cách yêu cầu thợ đào thêm SNARG chứng minh tính đúng đắn của cả hai điểm thăm dò đóng góp vào Block, tức là SNARG chứng minh tư cách thành viên cho ngôn ngữ sau: $L = \{((\Lambda; G; z; r';x'),(\Lambda_b;G_b;z_b;r'_b;x'))|V\Lambda(G\oplus(z,r',x'))=1 \wedge V\Lambda_b(G_b\oplus(z_b;r'_b;x'_b)) = 1\}$ trong đó $G\oplus(z,r',x')$ biểu thị đồ thị G mở rộng với đỉnh (z,r',x') như định nghĩa trong phần 3.2.



Hình 2: Sơ đồ quy trình khai thác PoUW.

Mô tả chi tiết về quy trình PoUW được đưa ra trong Thuật toán 5. Thuật toán khai thác được tham số hóa bởi Blockchain dài nhất nhận được *C*, thông báo được đưa vào Block *m*, trường hợp vấn đề Λ được thợ đào chọn để làm việc,[4] liên quan bản ghi *G* trích xuất từ *C*, và trạng thái bên trong được chọn *z*. Đầu vào Pre-Hash bao gồm các tham số này, Hash của Block *s* trước đó và một số ngẫu nhiên *r'* cho *M*. Tại thời điểm này, tất cả các tham số của *M*, Λ, *G*, *z*, và *r'*, được xác định đầy đủ dựa trên dữ liệu được Hash ban đầu, do đó thiết lập mỗi Pre-Hash nhỏ mà đối thủ tìm thấy chỉ có thể được sử dụng để thực hiện một lần thử Post-Hash phù hợp. Chúng tôi lưu ý rằng nếu trong một vòng, thợ đào không có đủ các bước để hoàn thành việc chạy quy trình PoUW, ví dụ như anh ta chỉ tìm được một Pre-Hash nhỏ, anh ta sẽ tiếp tục vòng tiếp theo kể từ thời điểm nó dừng lại.

Các Input Block được xử lý ở cấp ứng dụng của giao thức, vì chúng không cần thiết cho sự đồng thuận. Đặc biệt, để một Input Block được các thợ đào xem xét, nó phải đáp ứng các điều kiện nhất định: Pre-Hash và Post-Hash liên quan đủ nhỏ và việc tính toán *M* là chính xác. Lưu ý rằng các Ranking Block cũng được coi là Input Block và có thể được đưa vào trọng tải của các Ranking Block khác. Như trong [24], một Input Block có thể được đưa vào trọng tải của các Ranking Block

[4]Ngay cả khi khách hàng không đăng bất kỳ vấn đề nào như vậy, chúng tôi giả định rằng các thợ đào luôn có thể tạo ra sự cố MH dựa trên Hash của Block mà họ đang mở rộng. Ví dụ: có thể sử dụng PoW dựa trên băm theo thời gian cố định là [5, 14]. Điều này tương đương với một phép tính DPLS "dự phòng".

khác nhau trong các chuỗi phân kỳ. Điều này đảm bảo rằng tất cả các Input Block do một bên trung thực khai thác cuối cùng sẽ được đưa vào chuỗi chính và không có tiến trình nào bị mất. Quy trình đầy đủ được trình bày trong Phụ lục A.

Lưu ý 1. (Chi phí SNARG) Tính hữu ích của ghi chú không nhất thiết bị ảnh hưởng đáng kể bởi chi phí tính toán SNARG lớn vì mỗi bản cập nhật trạng thái bao gồm một số lượng lớn các bước thăm dò (trung bình) nhưng SNARG chỉ cho hai trong số các phép tính M được thực hiện. Do đó, số bước thăm dò trung bình này có thể được tăng lên để cân bằng với tần suất cập nhật trạng thái trong hệ thống, giúp thiết lập Yêu cầu 2b.

Thuật toán 5: Thủ tục PoUW được tham số hóa bởi các tham số độ cứng $T_1, T_2, T_3 \in \mathbb{N}$, hệ thống SNARG, hàm Hash $H(\cdot)$, thuật toán khám phá M và thuật toán tính điểm g.

1: var $(score_b, s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b) := (\infty, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp)$ -Các biến toàn cục liên quan đến lần chạy tốt nhất. Chúng được đặt lại khi thợ đào tìm thấy một Block hoặc trường hợp anh ta đang thực hiện các thay đổi.
2: var $(z, com) := (\perp, \perp)$
3: **funtion** PoUW (C, m, Λ, G)
4: $s := H(head(C))$
5: **if** $(z = \perp)$ **then** $z \leftarrow F(\Lambda, G)$ -Tính toán trạng thái bên trong
6: $r \leftarrow U_\lambda$ -Mẫu Nonce
7: $h := H(s, m, com, \Lambda, G, z, r)$ -Tính toán Pre-Hash
8: **if** $(h < T_1)$ **then**
9: $r' := H(s, m, com, \Lambda, G, z, r, h)$ -Tính toán hạt giống
10: $x' := M_\Lambda(G, z, r')$ -Tính toán DAG
11: $h' := H(s, m, com, \Lambda, G, z, r, h, x')$ -Tính toán Post-Hash
12: **if** $(h' < T_2$ or $T_2 \leq h' < T_3)$ **then** -Block mới
13: **if** $(s_b = \perp)$ **then** $(s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b) := (s, m, com, \Lambda, G, z, r, x')$
14: $\pi := SNARG.P(\Sigma, ((\Lambda, G, z, r', x'), (\Lambda_b, G_b, z_b, r'_b, x'_b)))$; -Tính toán chứng minh tính đúng đắn
15: $B := ((s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b), (s, m, com, \Lambda, G, z, r, x'), \pi)$
16: **if** $(h' < T_2)$ **then** $C = C\ B$ -Trả chuỗi mới về chức năng chính
17: **else** Diffuse$((input, B))$ -Khuếch tán Input Block khác
18: $(score_b, s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b) := (\infty, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp)$
19: **else** -Không có Block
20: **if** $(\Lambda \neq \Lambda_b$ or $g_\Lambda(x') > score_b)$ **then**
point

21: $(s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b) := (s, m, com, \Lambda, G, z, r, x')$ -Lưu trữ bản cập nhật tốt nhất

22: com := $H(s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b)$ -Cam kết cho nỗ lực tốt nhất trước đó

23: $score_b := g_\Lambda(x')$

24: return C

4.2. Xử lý nhiều trường hợp

Bây giờ chúng tôi mô tả cách mở rộng thiết kế để xử lý nhiều trường hợp sự cố hiệu quả hơn. Động lực chính là đối với một số hoạt động nhất định trong quá trình giải quyết, những người tham gia giao thức có thể muốn thiết lập một quan điểm chung về trạng thái của thuật toán đang thực thi. Ví dụ, người tạo ra vấn đề có thể muốn chi phối quá trình giải quyết bằng cách mở rộng thêm hoặc dừng nó, dựa trên chất lượng của các giải pháp được tìm thấy. Hơn nữa, thỏa thuận cho phép chúng tôi triển khai phức tạp hơn hành vi của chính thuật toán giải, ví dụ: khởi động lại quá trình giải nếu các tham số ban đầu dẫn đến một vùng xấu của không gian giải pháp, một kỹ thuật phổ biến trong tìm kiếm cục bộ ngẫu nhiên song song. Nếu không có thỏa thuận đầy đủ, các hoạt động như vậy có thể khó thực hiện hoặc dẫn đến thiếu hiệu quả.

Để đạt được thỏa thuận, chúng tôi đưa ra ý tưởng về *Pipelining*. Tương tự với Pipelining trong bộ xử lý kiến trúc, máy tính toán có hai giai đoạn: chạy thuật toán giải cho một vấn đề và thiết lập một quan điểm chung về trạng thái của thuật toán. Các trường hợp vấn đề khác nhau được xen kẽ để luân chuyển giữa hai giai đoạn này trong quá trình thực thi giao thức. Chi tiết hơn, giao thức xác định lịch trình dựa trên Epoch xác định vấn đề nào được giải quyết tại mỗi Epoch, mỗi Epoch được tính bằng m Block liên tiếp trong chuỗi chính. Ví dụ, Epoch đầu tiên (Block 1 đến m) được chỉ định vấn đề Λ_1, Epoch thứ hai (khối $m + 1$ đến $2m$) vấn đề Λ_2, Epoch thứ ba quay lại vấn đề Λ_1 , v.v. Một Epoch có một vấn đề nhất định không được chỉ định, cho phép các bên có đủ thời gian để đồng ý về tất cả các bản cập nhật trước đó được tạo cho vấn đề. Trong khi, trong ví dụ này, chỉ có hai vấn đề được giải quyết, lịch trình có thể chứa một số lượng lớn được chạy lần lượt, có thể với tần suất khác nhau. Chúng tôi lưu ý rằng các kịch bản lập lịch trình khác nhau có thể được thực hiện bằng cách sử dụng thông tin được đăng trong Blockchain, ví dụ: chọn làm trường hợp tiếp theo để giải quyết trường hợp mang lại phần thưởng lớn nhất. Hơn nữa, để đạt được thỏa thuận, chúng tôi cho phép các cập nhật liên quan chỉ được đưa

vào chuỗi chính đối với một số Block giới hạn *k* sau khi kết thúc Epoch mà sự cố đang được giải quyết một cách tích cực. Tham số *k* phải đủ lớn để đảm bảo rằng bất kỳ bản cập nhật trung thực nào được đảm bảo sẽ được đưa vào chuỗi chính.

Đáng ngạc nhiên, Pipelining cũng giúp đối phó với các cuộc tấn công DoS Input Block, đồng thời cho phép các thợ đào tiêu thụ các Input Block không ổn định trong quá trình khai thác, do đó tăng hiệu suất của DPLS. Chi tiết hơn, trong một cuộc tấn công DoS Input Block, kẻ tấn công khai thác trước một lượng lớn Input Block, và cuối cùng giải phóng chúng để tạo tác động đến mạng lưới. Để đối phó với một cuộc tấn công như vậy, Fruitchains [52] yêu cầu các Ranking Block hợp lệ chỉ được bao gồm các Input Block gần đây, tức là Input Block tham chiếu đến một Ranking Block gần đây. Cơ chế này không hoạt động trong cài đặt của chúng tôi, ở đó chúng tôi muốn thợ đào tham chiếu đến công việc bởi các Input Block không ổn định: kẻ tấn công có thể ngăn chặn một cách chắc chắn Input Block để ngăn chặn sự giải quyết cuối cùng của nó, trong khi một thợ đào trung thực vẫn sẽ tham chiếu nó trong Block của họ, do đó làm cho Block không hợp lệ. Thay vào đó, theo kiến trúc của chúng tôi, chúng tôi đối phó với các cuộc tấn công DoS bằng cách chỉ yêu cầu các Ranking Block hợp lệ bao gồm các Input Block (i) "mới", tức là chúng tham chiếu đến một Ranking Block của cùng một Epoch và (ii) tham chiếu các Input Block khác là một phần của chuỗi chính. Điều này có nghĩa là các Input Block cũ không phải là một phần của chế độ xem chung được thiết lập trước đó sẽ bị loại bỏ. Lưu ý rằng bất chấp những hạn chế này, những thợ đào trung thực vẫn có thể tham chiếu các Input Block mới chưa được giải quyết trong khi khai thác một Block mới, vì họ có thể chắc chắn rằng có đủ thời gian để các Block này được đưa vào chuỗi chính.

4.3. Cơ cấu khuyến khích

Phần thưởng cho công việc hữu ích trong giao thức được trao trong các khoảng thời gian thực thi giao thức theo Epoch được đo bằng các khoảng thời gian Ranking Block cố định. Khách hàng gửi các phiên bản DPLS Λ để xử lý sẽ khóa tiền để thực hiện thuật toán khởi tạo nhóm phần thưởng P_Λ cho bài toán. Vào cuối mỗi Epoch, đóng góp của mỗi thợ đào đối với việc giải quyết một phiên bản cụ thể

Λ sẽ được đo bằng phần Input Block mà họ đã đóng góp trong tổng số được hướng đến phiên bản đó. Sử dụng số liệu này, các thợ đào sẽ được thưởng tương ứng từ nhóm phần thưởng của mỗi bài toán. Lưu ý rằng phần P_Λ phân bổ cho một Epoch cụ thể sẽ được xác định bởi một hàm được khách hàng lựa chọn lấy làm đầu vào cho tổng công việc đầu tư vào vấn đề cho Epoch cụ thể đó. Khách hàng có thể tiếp tục tăng nhóm phần thưởng P_Λ khi nó cạn kiệt để đảm bảo sự quan tâm liên tục của các thợ đào đối với trường hợp vấn đề của họ.

Cơ chế trên về cơ bản dựa trên cơ chế 2 cho 1 và thực tế là bất kỳ bộ khai thác nào sẽ tạo ra một tập hợp con các bản cập nhật tỷ lệ thuận với sức mạnh tính toán. Theo cách này, giao thức của chúng tôi là công bằng và có thể được chứng minh như vậy dọc theo dòng Fruitchains [52]; chúng tôi bỏ qua các chi tiết khác. Chúng tôi lưu ý rằng, ngoài những điều trên, những thợ đào có thể được thưởng thêm bằng phí giao dịch và dựa trên nguồn cung tiền cơ sở bằng cách tạo ra một nhóm phần thưởng bổ sung không liên quan đến bất kỳ vấn đề tối ưu hóa nào.

Sử dụng thiết lập trên, có thể dễ dàng thấy rằng những thợ đào hợp lý có động lực để gắn bó với giao thức dưới giả định rằng phần thưởng bù đắp được chi phí hoạt động của họ. Lưu ý rằng thợ đào "thất bại" (Griefing) vẫn có thể đi chệch hướng phần nào bằng cách không xuất bản bản cập nhật tốt nhất của họ cho thuật toán hữu ích trong bước song song bên trong và thay vào đó chọn bản cập nhật được chọn ngẫu nhiên. Đây là một kiểu "thất bại" vì thợ đào không thu được gì khi làm điều này (vì công việc đã được hoàn thành). Mặc dù việc đối phó với các cuộc tấn công như vậy nằm ngoài phạm vi giải thích hiện tại, chúng tôi lưu ý rằng có thể đánh giá các Block đối với hành vi "thất bại" đó bằng cách đối chiếu thống kê giữa bản cập nhật được xác nhận "tốt nhất" với bản cập nhật "chiến thắng" được liên kết với Post-Hash nhỏ.

Việc phân phối Coin gốc của sổ cái được thực hiện với một lịch trình mở khóa Coin thông qua phần thưởng Block cho mỗi Ranking Block. Hãy nhớ rằng trong tình huống không có trường hợp sự cố nào khác, các công cụ khai thác sẽ hoàn nguyên về trường hợp sự cố mặc định (xem chú thích 4). Việc giải quyết trường hợp này không có bất kỳ giá trị bên ngoài nào nhưng nó cung cấp một cách để khởi động nền tảng (vì các thợ đào sẽ tham gia vào Genesis với ý định nhận Token từ các Ranking Block). Thị trường nổi lên xung quanh tài sản gốc

của nền tảng có những người đặt ra vấn đề khi mua Token trong thị trường mở từ các thợ đào hoặc nhà đầu cơ, để họ tài trợ cho các trường hợp có vấn đề mà họ đăng trên nền tảng.

5. Phân tích bảo mật

Tiếp theo, chúng tôi chính thức phân tích tính bảo mật của giao thức. Đầu tiên, chúng tôi chỉ ra rằng giả định việc tính toán DAG cơ bản là vừa phải và các bên trung thực kiểm soát phần lớn sức mạnh tính toán trong mạng lưới, giao thức thực hiện một sổ cái giao dịch mạnh mẽ. Sau đó, chúng tôi xác định và phân tích tỷ lệ hữu ích của giao thức.

5.1. Bảo mật sổ cái

Giả sử Π biểu thị cho giao thức Blockchain của chúng tôi. Việc phân tích tính nhất quán của quy tắc chuỗi dài nhất xuất hiện trong Π liên quan đến một số thách thức mới, bao gồm một chuỗi Markov kỳ lạ chi phối động lực khai thác và khả năng khởi động lại trong chuỗi này được tạo ra bởi việc phân phối một Block mới, có lẽ bởi đối thủ. Chúng tôi điều chỉnh ngôn ngữ của [37, 6] cho phù hợp với cài đặt này và sau đó phát triển các công cụ cần thiết cho phân tích xác suất liên quan. (Cách xử lý dưới đây của chúng tôi không yêu cầu bạn phải quen thuộc với các nghiên cứu trước đây)

Để đơn giản hơn, trong phần chính của nghiên cứu, chúng tôi thảo luận về trường hợp *không khởi động lại*, nghĩa là giao thức được thực hiện bởi các bên trung thực không bắt đầu lại quá trình khai thác khi nó biết về một chuỗi dài hơn, mà là hoàn thành tính toán hiện tại. Mộc cách trực giác, việc khởi động lại *cải thiện* các thuộc tính bảo mật của Blockchain, vì chúng giúp đảm bảo rằng các bên trung thực đang khai thác trên các chuỗi hiện tại. Tuy nhiên, tình hình hơi phức tạp bởi thực tế là việc khởi động lại cho phép kẻ tấn công tương quan với trạng thái của các bên trung thực trong chuỗi Markov. Đặc biệt, hãy lưu ý rằng một đối thủ nắm giữ một chuỗi vượt quá độ dài của những chuỗi mà các bên trung thực nắm giữ có thể phát hành chuỗi đó một cách chiến lược cho những người chơi trung thực. Họ có thể có kiến thức chi tiết về trạng thái hiện tại để đạt được một số quyền kiểm soát ngắn hạn đối với việc phân phối thành công khai thác trung thực. Mặc dù có những mối tương quan như vậy, chúng tôi cho thấy trong Phụ lục D rằng trực giác ở trên là đúng: lợi thế đối đầu đạt

được bằng cách hiển thị các Block gian lận cho những thợ đào trung thực bị lu mờ bởi thực tế là những tiếp xúc đó làm tăng độ dài của Blockchain được nắm giữ bởi người nhận trung thực; theo ngôn ngữ của phân tích bên dưới, một sự tiếp xúc như vậy có lợi ích cũng giống như một chiến thắng khai thác trung thực!

Chúng tôi áp dụng mô hình thời gian rời rạc, chia thời gian thành các "vòng" (Round) ngắn với thời lượng c_H bằng thời gian thực hiện truy vấn Hash. Chúng tôi đưa ra các sự kiện tạo Block thiết yếu của quá trình thực thi giao thức bằng một chuỗi *đặc trưng*: chuỗi này xác định đối với mỗi vòng, số lượng Ranking Block gian lận và trung thực được tạo ra. Do đó, các chuỗi đặc trưng của có cấu trúc $w = w_1, \ldots w_L$ với mỗi $w_i = (h_i, a_i) \in \mathbb{N}^2$, h_i và a_i biểu thị số lượng khám phá Ranking Block trung thực và gian lận tương ứng; ở đây L là thời gian tồn tại của giao thức.

Cuối cùng, giao thức của chúng tôi Π xác định một Blockchain của Ranking Block, bản thân chúng đề cập đến các Input Block. Cấu trúc như vậy xác định thứ tự tuyến tính trên tập hợp các Input Block được tham chiếu trong Blockchain của Ranking Block (bằng cách sắp xếp thứ tự các Input Block được tham chiếu trong một Ranking Block cụ thể theo thứ tự tham chiếu của chúng trong Ranking Block). Cuối cùng, chúng tôi muốn thiết lập hai thuộc tính cơ bản của sổ cái: *tính tồn tại* và *tính bền bỉ*.

Tính bền bỉ với tham số $k \in \mathbb{N}$. Khi một Node của hệ thống công bố một Input Block nhất định trong phần *ổn định* của sổ cái L, các Node còn lại hoặc báo cáo Input Block ở cùng vị trí của sổ cái của chúng hoặc báo cáo một sổ cái ổn định là tiền tố của L. Ở đây khái niệm ổn định là một vị từ được tham số hóa bởi tham số bảo mật k; cụ thể, một Input Block được tuyên bố là *ổn định* khi và chỉ khi nó nằm trong một (Ranking) Block có nhiều hơn k (Ranking) Block nằm sâu trong sổ cái.

Tính tồn tại với tham số $u \in \mathbb{N}$. Nếu tất cả các Node trung thực trong hệ thống cố gắng bao gồm một Input Block nhất định thì sau thời gian trôi qua tương ứng với u vòng, tất cả các Node báo cáo Input Block là ổn định.

Chúng tôi thiết lập các thuộc tính này là hệ quả của ba thuộc tính cơ bản hơn của Blockchain của Ranking Block, được xây dựng ban đầu trong [24] (chúng tôi sử dụng công thức được điều chỉnh một chút từ [18]):

- **Tiền tố chung (CP - Common Prefix); với tham số** $k \in \mathbb{N}$. Các chuỗi C_1, C_2 được chấp nhận bởi hai bên trung thực khi bắt đầu các vòng $r_1 \leq r_2$ sao cho $C^k_1 \prec C2$, trong đó C^k_1 biểu thị chuỗi có được bằng cách loại bỏ *k* Block cuối cùng khỏi C_1 và $\prec$ biểu thị quan hệ tiền tố.
- **Chất lượng chuỗi hiện tại (ECQ - Existential Chain Quality); với tham số** $s \in \mathbb{N}$. Hãy xem xét chuỗi *C* được chấp nhận bởi một bên trung thực khi bắt đầu một vòng và bất kỳ phần nào của *C* kéo dài *s* vòng trước liền kề; thì ít nhất một Block được tạo trung thực sẽ xuất hiện trong phần này.
- **Tăng trưởng chuỗi (CG - Chain Growth); với các tham số** $\tau \in (0, 1]$ **và** $s \in \mathbb{N}$. Hãy xem xét chuỗi *C* được sở hữu bởi một bên trung thực khi bắt đầu một vòng và bất kỳ phần nào của *C* kéo dài *s* vòng trước liền kề; khi đó số Block xuất hiện trong phần này của chuỗi ít nhất là τs. Chúng tôi gọi τ là *hệ số tốc độ*.

Một trong những kết luận quan trọng của nghiên cứu trước đó là các thuộc tính này (CP, CG và ECQ) thể hiện trực tiếp tính tồn tại và tính bền bỉ. Từ góc độ phân tích, có thể được đảm bảo chỉ dựa trên chuỗi đặc trưng liên quan đến một thực thi cụ thể. Thực tế này khá tức thì đối với CG và ECQ, trong khi việc xác định các thuộc tính của chuỗi đặc trưng đảm bảo CP thì tinh vi hơn.

Chúng tôi đưa ra bản tóm tắt lý thuyết này trong Phụ lục C, vừa để bài viết được bí mật vừa để chúng tôi có thể mô tả phần mở rộng để khởi động lại trong Phụ lục D. May mắn thay, có thể tóm tắt ngắn gọn các kết luận của lý thuyết này khi chúng liên quan đến nhu cầu của chúng tôi, là đơn hàng tiếp theo của doanh nghiệp.

Để tiếp tục, trước tiên chúng tôi giới thiệu hai giả định liên quan đến mức độ của độ cứng vừa phải của họ tính toán DAG cơ bản mà tôi sử dụng bởi Π và độ phức tạp của hệ thống SNARG được sử dụng.

Giả định 1. Đối với các tham số $t\hat{}$, $\epsilon\hat{}$, $k\hat{}$, chúng tôi giả định rằng họ tính toán DAG *I* mà tôi đã sử dụng trong Π là $(t\hat{}, \epsilon\hat{}, k\hat{})$ – MH (độ cứng vừa phải).

Giả định 2. Đối với các tham số c_P, c_V, c_S, chúng tôi giả định rằng tồn tại một hệ thống SNARG, SNARG đang chạy định lý (resp., verifier, setup) thực hiện

các bước c_P (resp. c_V, c_S).

Cho $w = w_1, \ldots w_L$ là một chuỗi đặc trưng, như trên. Chúng tôi cố định hằng số Γ, một khoảng thời gian với *đảm bảo tuần tự hóa* Γ như sau: *nếu Ranking Block B_2 được tạo bởi bên trung thực P ít nhất* Γ *vòng sau khi Ranking Block B_1 tạo ra sự trung thực được khuếch tán, thì tính toán đầy đủ hỗ trợ B_2 (bao gồm cả Pre-Hash) được thực hiện trong khi P nhận thức được B_1.* Trong cài đặt của chúng tôi, Γ có thể được đặt thành $2 + \Delta + c_P/c_H + t\hat{}/c_H$ (tương ứng với số vòng được thực hiện để tạo Pre-Hash (≤ 1), công việc hữu ích ($\leq t\hat{}/c_H$), Post-Hash (≤ 1) và SNARG (c_P/c_H) cho Block B_2 ngoài bất kỳ độ trệ của mạng lưới). Với điều này, chúng tôi nói rằng t là *một vòng thành công riêng biệt duy nhất (isolated uniquely successful round)* Γ nếu vùng $w_{t-\Gamma} \ldots w_t \ldots w_{t+\Gamma}$ thoả mãn $h_t = 1$ và hơn nữa, tổng $h_i = 1$ trên vùng này (hãy nhớ $w_i = (a_i, h_i)$). Lưu ý rằng một vòng không thể bị cô lập nếu nó không được theo sau bởi ít nhất Γ ký hiệu. Đối với mỗi t xác định I_t là một biến chỉ số cho trường hợp t là một vòng thành công riêng biệt duy nhất.

Các đại lượng cơ bản của lãi suất được đưa ra bởi hai quy ước để tính số dư của thành công đối nghịch và trung thực.

Định nghĩa 4: (Walk rào cản; Walk tự do). Giả sử $x = x_1, \ldots, x_n \in \mathbb{N}^*$. Xác định *Walk hàng rào* $B(x)$ bằng quy tắc đệ quy $B(\epsilon) = 0$ (đối với chuỗi trống €) và, với bất kỳ $x \in \mathbb{N}^*$ và $a \in \mathbb{N}$, $B(xa) = \max(B(x) + a, 0)$. Tương tự, xác định *Walk tự đo* $F(x) = \sum_i xi$.

Định nghĩa 5. Đối với chuỗi đặc trưng $w \in (\mathbb{N}^2)^L$ và $0 < t \leq L$, xác định *hiệu ứng ký quỹ (Margin Effect)* $w^*_t = a_t - I_t \in \mathbb{N}$ (và w^* là dãy các phần tử của $\mathbb{N}$ được cho bởi quy tắc này). Sau đó chúng ta xác định $B^*(w) = B(w^*)$ và $F(w) = F(w^*)$. Cuối cùng, đối với một chuỗi đặc trưng $w = xy$ với $|x| = l$, chúng ta xác định *lợi nhuận riêng biệt (Isolated Margin) l* của w là $\beta_l(w) = B(x^*) + F(y)$.

Vai trò của lợi nhuận riêng biệt l được làm rõ bằng cách sau, nó thiết lập một kết nối trực tiếp đến tiền tố chung.

Định lý 6. *Cho $w \in (\mathbb{N}^2)^L$ là chuỗi đặc trưng được liên kết với một thực thi thỏa mãn bảo đảm tuần tự hóa Γ. Giả sử, xa hơn nữa rằng (i.) Thực thi thỏa mãn (k / s, s) -CG, và (ii.) với bất kỳ tiền tố xy của w mà $|y| \geq s$, chúng ta có $\beta_{|x|}(xy) < 0$.*

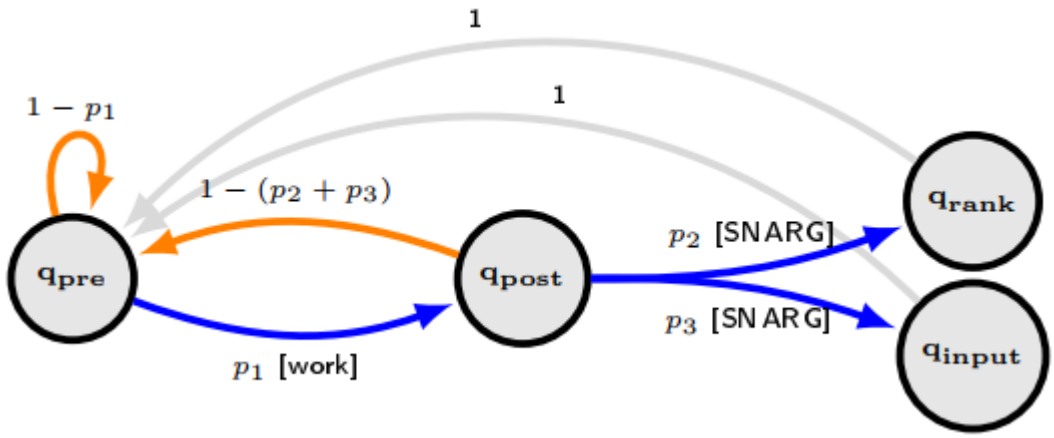
Khi đó việc thực hiện thỏa mãn k-CP.

Đây là thành phần chính trong định lý sau; như đã lưu ý, các chi tiết của lý thuyết hiện có này được thảo luận trong Phụ lục C.

Định lý 7. *Gọi D_{Π} là một phân phối trên các chuỗi đặc trưng có độ dài L (do giao thức Π gây ra), λ là tham số bảo mật, và α> β là hai hằng số tương ứng với tỷ lệ các Block riêng biệt duy nhất và tỷ lệ các Block gian lận. Giả sử với một hằng số δ <(α - β)/2, khi w được vẽ từ D_{Π}, mọi khoảng của w có độ dài poly(λ) có ít nhất α - δ Block riêng biệt duy nhất và không nhiều hơn β + δ Block gian lận, ngoại trừ với xác suất không đáng kể. Sau đó, ngoại trừ với xác suất không đáng kể, giao thức thỏa mãn (i.) CG với s = poly(λ) và hệ số tốc độ không đổi, (ii.) ECQ với s = poly(λ) và (iii.) CP với tham số k = poly(λ).*

5.1.1. Phân tích chuỗi Markov

Theo mô tả ở trên, chúng tôi đặc biệt quan tâm đến việc phân tích trình tự của (i.) thành công khai thác đối nghịch và (ii.) thành công trung thực riêng biệt duy nhất. Việc phân tích được đơn giản hóa bởi thực tế là diễn biến thời gian của các bên trung thực là độc lập. Chúng tôi tập trung vào chuỗi Markov trong hình bên dưới, hiển thị các Node cho “Pre-Hash”, "Post-Hash", và cả sản xuất "Ranking" và "Input" Block. Thật thuận tiện cho chúng tôi để trang trí thêm cho quá trình chuyển đổi với độ trễ: các cạnh màu cam được duyệt qua một vòng (hoặc thời gian c_H, tương ứng với các truy vấn Hash), các cạnh màu xám được duyệt ngay lập tức và các cạnh màu xanh có thời gian chuyển tiếp được cung cấp bởi phân phối của công việc hữu ích (giới hạn trên bởi t^) và thời gian SNARG (c_P). (Lưu ý rằng độ trễ thời gian được chỉ ra trong chuỗi này có thể được thực hiện với đường dẫn của các trạng thái riêng lẻ được kết nối bằng các cạnh có độ trễ đơn vị. Do đó, bản trình bày này có thể được kết thúc bằng chuỗi Markov tiêu chuẩn). Trong khi các thuộc tính bảo mật cơ bản của giao thức chỉ phụ thuộc vào sản xuất các Ranking Block, động lực của chuỗi Markov phụ thuộc vào cả Ranking Block và Input Block.



Chúng tôi bắt đầu bằng cách thiết lập điều đó. Mặc dù thực tế là các bên trung thực bắt đầu giao thức được đồng bộ hóa (trước). Họ nhanh chóng hội tụ đến các vị trí độc lập lẫn nhau trong chuỗi khai thác. Sắp tới, lập luận hỗn hợp này sẽ là công cụ để thiết lập các giới hạn đối với sản xuất Block riêng biệt duy nhất.

5.1.2. Thời gian hỗn hợp; hội tụ để độc lập lẫn nhau

Bằng một đối số ghép nối tiêu chuẩn, chúng tôi nhận được như sau:

Bổ đề 8. *Xét m hạt P_1 , ... , P_m phát triển độc lập trên chuỗi Markov với mọi trạng thái ban đầu cố định. Đặt (S_1, ..., S_m) biểu thị một biến ngẫu nhiên để mỗi tọa độ là độc lập và đứng yên trên chuỗi. Sau đó cho $T = (1 + t\hat{} + c_P/c_H)$,*

$$\|(P_1^T, \ldots, P_m^T) - (S_1, \ldots, S_m)\|_{t.v} \leq m(1 - p_{couple})^L ,$$

trong đó $||X - Y||_{t.v}$ biểu thị khoảng cách trong tổng biến thiên giữa các biến ngẫu nhiên X và Y. Ở đây $p_{couple} > 0$ là hằng số chỉ phụ thuộc vào $c_P / t\hat{}$.

Chứng minh. Chúng tôi tiến hành với một đối số khớp nối tiêu chuẩn. Xét *m* hạt (bên) P_1 , ... , P_m, ban đầu ở trạng thái q_{pre}, tiến hành đồng thời, độc lập theo động lực của chuỗi. Chúng tôi muốn chỉ ra rằng sự phân bố chung vị trí của tất cả các hạt nhanh chóng hội tụ về *m* bản sao độc lập của phân phối tĩnh. Với mục đích này, hãy xem xét thêm *m* hạt R_1 , ... , R_m trên chuỗi, ban đầu phân phối độc lập theo phân phối tĩnh. Chúng tôi đặt P^t_i và R^t_i là vị trí của các hạt tại thời điểm *t*. Chúng tôi đưa ra một cách ghép đơn giản *C* về sự phát triển của P^t_1, … , P^t_m với R^t_1, … , R^t_m, và áp dụng "bổ đề ghép" chuẩn thiết lập sự hội tụ cho phân phối tĩnh. Ghép nối *C* được mô tả, tại mỗi bước thời gian, bởi một họ các biến ngẫu nhiên U^t_i; với mỗi $i \in \{1, \ldots, m\}$, $U^t_i : Q \rightarrow Q$ là một hàm trong đó *Q* là tập hợp các trạng thái của chuỗi (thực tế là lớn hơn so với biểu đồ chỉ ra là kết quả của việc thực hiện các quá trình chuyển đổi "dài"). "Các hàm cập nhật" U_i được chọn để toàn bộ các mục

nhập ($U^t_i(q)$) (trên tất cả t, i và $q \in Q$) là độc lập và mỗi $U_i(q)$ được phân phối theo phân phối xác định cho trạng thái q. Sau đó, P_i và R_i được cập nhật theo cùng một bản cập nhật: $P_i^{t+1} = U_i(P^t_i)$ và $R_i^{t+1} = U_i(R^t_i)$. Quan sát các động lực của P^t_i, mỗi biến đổi độc lập theo chuỗi; điều này cũng đúng với R^t_i, tất nhiên là tiếp tục độc lập và cố định. Quan sát thấy nếu $R^t_i = P^t_i$ tại một thời điểm t nào đó, thuộc tính này sẽ được giữ lại bởi khớp nối trong tương lai (vì chúng có cùng hàm cập nhật). Bây giờ, hãy xem xét bất kỳ khoảng thời gian nào có độ dài $E = 1+ (t^ + c_P)/c_H$ vòng và bất kỳ cặp hạt P_i và Q_i nào. Quan sát thấy cả hai hạt phải đến trạng thái q_{pre} trong khoảng thời gian này (vì $t^$ và c_P là giới hạn trên của thời gian chuyển tiếp của quá trình chuyển đổi màu xanh lam); theo đó, nếu hạt đầu tiên trong hai hạt tới q_{pre} vẫn ở trạng thái đó trong phần còn lại của bước thời gian E thì hai hạt phải kết đôi (nghĩa là trùng hợp trong khoảng thời gian này và mãi mãi về sau). Hãy nhớ rằng chúng tôi lấy $T_1 \geq t^/c_H$, xác suất để hạt đầu tiên vẫn ở trong q_{pre} khi hạt thứ hai đến là ít nhất $p_{couple} := (1 - p_1)^{E/cH} = (1 - p_1)^{(t^+cP) / cH} = [(1 - p_1)^{t^ / cH}]^{(1+cP) / t^} \geq [(1-1/T_1)^{T1}]^{(1+cP/t^)} \geq (1/e - O(1/T_1))^{(1+cP/t^)}$. Do đó, p_{couple} là một hằng số lớn hơn 0 (và có thể được giới hạn dưới dạng hàm của hằng số $c_P/t^$). Lưu ý rằng các sự kiện mà P_i kết đôi với R_i (đối với i riêng biệt) trong một Epoch như vậy là độc lập, và điều đó xảy ra sau L Epoch đó, xác suất có một cặp (P_i, R_i) không ghép đôi là không quá $n(1\text{-}p_{couple})^L$. Theo bổ đề ghép nối chuẩn (xem, ví dụ, [4, 12]), sau khoảng cách L Epoch tổng biến thiên giữa (P_1, .., P_m) và phân bố đứng yên độc lập trong mỗi tọa độ là không quá $m(1\text{-}p_{couple})^L$, có xu hướng nhanh chóng về 0 theo cấp số nhân trong L. Điều này chứng minh bổ đề.

5.1.3. Giới hạn đối với các sự kiện quan tâm

Như ở trên, hãy xem xét mật độ của các hạt (người chơi) P_1, ... , P_n trên chuỗi Markov. Theo sự phát triển của các hạt này, được đưa ra bởi các biến ngẫu nhiên P^t_i, chúng tôi quan tâm đến việc thiết lập các giới hạn trên về tốc độ mà đối thủ tạo ra các Ranking Block và giới hạn thấp hơn về tốc độ mà những người chơi trung thực tạo ra các khối cô lập duy nhất.

Bổ đề 9. *Xét m bên, với các điều kiện ban đầu tùy ý nhưng phát triển độc lập trên chuỗi Markov. Cho $S = (t^ + c_P)/c_H$ +1 và xem xét bất kỳ khoảng thời gian nào của R vòng, khoảng thời gian đầu tiên bắt đầu ít nhất S bước sau khi quá trình tiến hóa bắt đầu. Sau đó, xác suất mà một người chơi cụ thể tạo ra ít nhất k Ranking Block trong khoảng thời gian này không quá*

$$\binom{R+S}{k}(p_1p_2)^k \leq (R+S)^k(p_1p_2)^k.$$

Chứng minh. Để một Ranking Block được tạo ra trong khoảng thời gian, Pre-Hash được liên kết với Ranking Block phải thành công không sớm hơn *S* vòng trước khi bắt đầu khoảng thời gian. Để phân tích sự kiện này, giả sử *I* biểu thị tập hợp các vòng bao gồm tất cả các vòng được đề cập trong phát biểu của định lý và *S* vòng trước đó. Gọi A^*_i và B^*_i biểu thị hai họ biến ngẫu nhiên chỉ báo độc lập mà $\Pr[A^*_i = 1] = p_1$ và $\Pr[B^*_i = 1] = p_2$. Sau đó xác định A_i là biến ngẫu nhiên của chỉ báo với định nghĩa sau: nếu người chơi hoàn thành truy vấn Pre-Hash trong vòng thứ *i*, thì A_i là biến chỉ báo cho sự thành công của Pre-Hash này; nếu không, không có Pre-Hash nào được hoàn thành và $A_i = A^*_i$. Tương tự, B_i được xác định các biên tương tự: nếu vòng *i* chứa một bước Pre-Hash thành công, B_i là biến ngẫu nhiên chỉ báo cho sự kiện mà bước Post-Hash tiếp theo thành công; nếu không, không có Pre-Hash thành công trong vòng *i* và $B_i = B^*_i$. Nếu có *k* Ranking Block được tạo thành công trong *R*, thì $A_iB_i = 1$ cho ít nhất *k* trong số *i*. Hơn nữa, do kết quả của các định nghĩa có điều kiện ở trên, họ A_i và B_i là các biến ngẫu nhiên Bernoulli độc lập. Như vậy xác suất của *k* thành công không quá $\binom{R+S}{k}(p_1p_2)^k$ như mong muốn.

Bổ đề 10. *Hãy xem xét m các bên độc lập di chuyển trên chuỗi Markov trong phân phối tĩnh. Gọi* p^*_{rank} *biểu thị xác suất đứng yên của* q_{rank}*, khi đó* $\Pr[t \text{ là một vòng cô lập duy nhất}] \geq m(1 - (3\Gamma)p_1p_2)^m p^*_{rank}$.

Bằng chứng. Chúng tôi xem xét sự kiện mà một người chơi trung thực cụ thể P_w tạo ra một Block riêng biệt duy nhất trong vòng *t*. Gọi p^*_{rank} biểu thị xác suất q_{rank} dưới phân phối tĩnh. Khi đó xác suất thành công trong vòng *t* chính xác là p^*_{rank}. Trong trường hợp P_w tạo ra một Ranking Block trong vòng *t*, hãy xem xét quỹ đạo của P_w qua chuỗi Markov trong đó vòng lặp giữa vòng *t* và lần truy cập trước đó tới q_{pre} bị loại bỏ. Quỹ đạo còn lại tuân theo động lực học của chuỗi mà không cần điều hòa, và theo đó xác suất P_w tạo ra Block thứ hai trong vùng có kích thước $2\Gamma + 1$ này không lớn hơn $3\Gamma(p_1p_2)$ (lưu ý rằng đại lượng *S* của bổ đề trên không quá $(t\hat{} + c_P)/c_H + 1 \leq \Gamma - 1$). Một lần nữa áp dụng bổ đề trước, xác suất để $m - 1$ người chơi trung thực còn lại không tạo ra Block nào trong vùng này là ít nhất $(1 - (3\Gamma)p_1p_2)^{m-1}$. Lưu ý rằng các sự kiện mà những người chơi khác biệt đóng vai trò của P_w ở trên là không giao nhau, chúng tôi kết luận rằng $\Pr[t \text{ là một vòng cô lập duy nhất}] \geq m(1 - (3\Gamma)p_1p_2)^m p^*_{rank}$.

Theo bổ đề 8, điều sau đây là ngay lập tức.

Bổ đề 11. *Hãy xem xét m người chơi phát triển theo chuỗi Markov, những người*

chơi ban đầu đứng yên và độc lập. Gọi p_{couple} biểu thị hằng số ghép nối của Bổ đề 8. Xét hai vòng $s^{\check{}} < s$ mà $|s^{\check{}} - s| \geq L\ (1 + (t^{\hat{}} + c_P)/c_H)$. Gọi I_s biểu thị biến ngẫu nhiên của chỉ báo cho sự kiện s là riêng biệt duy nhất. Gọi C biểu thị một sự kiện tùy ý chỉ phụ thuộc vào quỹ đạo của người chơi trước $s^{\check{}}$. Khi đó $|\Pr[I_s|C] - \Pr[I_s]| \leq (1 - p_{couple})^L$.

Bổ đề 12. *Xem xét m người chơi phát triển trên chuỗi Markov với bất kỳ trạng thái ban đầu cố định nào. Gọi p_{iso} biểu thị xác suất một vòng riêng biệt duy nhất dưới phân phối tĩnh, giới hạn bên dưới bởi Bổ đề 10. Cố định một tham số $\sigma > 0$ và xác định $L = \ln(p_{iso}\sigma/2) / \ln(1 - p_{couple})$ và $E = L\ (1 + (t^{\hat{}} + c_P)/c_H)$. Cho $\{R, \ldots, R + S - 1\}$ là một chuỗi các vòng trong đó $R \geq E$. Gọi I_s sự kiện người chơi tạo ra một Block riêng biệt duy nhất trong vòng s. Sau đó*

$$\Pr\left[\sum_s I_s \leq (1-\sigma)p_{iso}S\right] \leq E\exp\left(-\frac{(1-\sigma/2)\sigma^2 p_{iso}\cdot S}{8E}\right).$$

Chứng minh. Chúng tôi chứng minh một phiên bản tham số hóa nhiều hơn một chút. Xét một giá trị tùy ý của $L > 0$ và xác định $\delta = (1 - p_{couple})^L$ và $E = L\ (1 + (t^{\hat{}} + c_P)/c_H)$ (phù hợp với phát biểu của bổ đề). Sau đó, chúng tôi hiển thị điều đó cho bất kỳ $\gamma > 0$

$$\Pr\left[\sum_s I_s \leq (1-\gamma)(p_{\text{iso}} - \delta)S\right] \leq E\exp(-\gamma^2 S(p_{\text{iso}} - \delta)/2E).$$

Phát biểu bổ đề sau bằng cách chọn $\gamma = (1 - \sigma/2)$ và $L = \ln(p_{iso}\sigma / 2) / \ln(1 - p_{couple})$ sao cho $\delta = p_{iso}\sigma/2$.

Quan sát thấy các biến I_s không độc lập; tuy nhiên, đối với hai chỉ số khoảng cách *s* và *s'*, chúng gần như độc lập như công thức trong bổ đề trên và trên thực tế, biến I_s gần như không phụ thuộc vào bất kỳ điều kiện nào trên các biến $I_1, \ldots, I_{s^{\check{}}}$ với $s^{\check{}} < s - E$. Để khai thác điều này, chúng ta sắp xếp các biến ngẫu nhiên thành tập hợp *E* gồm các biến "khoảng cách": tập hợp thứ *i* gồm các biến I_i, I_{E+i}, I_{2E+i}, …. Quan sát thấy tất cả các cặp biến trong một tập hợp cụ thể là ít nhất *E*-khoảng cách và tập hợp phân vùng tập hợp đầy đủ các biến. Mặc dù các biến trong một tập hợp cụ thể không hoàn toàn độc lập, nhưng chúng thỏa mãn yêu cầu $E[X_i|X_j, j < i] \geq p_{iso} - \delta$. Nhắc lại ràng buộc Chernoff cơ bản: nếu $X_1, \ldots, X_n$ là các biến ngẫu nhiên chỉ báo độc lập với $\text{Exp}[X_i] = p$ thì $\Pr[\sum Xi < (1 - \gamma)\ np] \leq \exp(-\gamma^2 np/2)$. Chính ràng buộc Chernoff này áp dụng cho các biến trong một tập hợp duy nhất thông qua đối số thống trị ngẫu nhiên chuẩn so sánh chúng với i.i.d. các biến có cùng kỳ vọng này. Cuối cùng, hãy quan sát nếu $\sum_s I_s \geq (1 - €)\ (p_{iso} - \delta)S/E$

cho mỗi tập hợp, thì bất đẳng thức tương tự này áp dụng cho toàn bộ tập hợp các biến. Lấy sự liên kết ràng buộc về những sự kiện xấu *E* này sẽ kết thúc lập luận

Bảng 1: *Các thông số phân tích*.

	Tham số bảo mật
λ :	Số lượng các bên
n :	Ràng buộc sự gian lận của bên đối thủ
t :	Ràng buộc gian lận của bên đối thủ được khuếch đại
t' :	Các bước "khai thác" mà mỗi bên thực hiện mỗi vòng
c_H :	Định lý SNARG / chi phí xác minh
$c_{\mathsf{P}}, c_{\mathsf{V}}$:	Tham số MH DAG
$\hat{\epsilon}, \hat{t}, \hat{k}$:	Mục tiêu / xác suất thành công của Pre-Hash
$T_1, p_1 = T_1/2^\lambda$:	Mục tiêu / xác suất thành công của Ranking Block Post-Hash
$T_2, p_2 = T_2/2^\lambda$:	Mục tiêu / xác suất thành công của Input Block Post-Hash
$T_3, p_3 = \frac{T_3 - T_2}{2^\lambda}$:	Tham số giới hạn nồng độ
σ :	Mạng /uần tự hóa độ trễ trong trường hợp xấu nhất
Δ, Γ :	Giới hạn trên về tỷ lệ tính toán Ranking Block
β :	Lợi thế đối đầu trong tỷ lệ tính toán DAG
δ_{MH} :	Lợi thế trung thực về số bước mỗi vòng
δ_{Steps} :	Giới hạn trên trên tổng tỷ lệ tính toán Block
δ_{tot} :	

5.1.4. Giới hạn về số lần khai thác đối thủ thành công

Tiếp theo, chúng tôi tiến hành ràng buộc tỷ lệ khai thác đối thủ thành công. Phân tích của chúng tôi sẽ phụ thuộc vào ở mức độ độ cứng vừa phải của họ tính toán DAG cơ bản.

Theo Bổ đề 3, tốc độ tăng tốc đối thủ nhận được bằng mỗi truy vấn bổ sung cho Oracle $\mathcal{O}$ là tối đa $t\hat{}$ bước. Do đó, để bảo vệ giao thức khỏi các cuộc tấn công nghiền nát, chúng tôi *đặt tham số độ cứng Pre-Hash* p_1 đến $c_H/((1+\sigma)t\hat{}+4)$, trong đó $\sigma \in (0,1)$ là một tham số liên quan đến giới hạn nồng độ mà chúng tôi sử dụng sau trong phân tích. Đặt p_1 theo cách này ngụ ý rằng việc tính toán một chi phí Pre-Hash nhỏ trên kỳ vọng $c_H/p_1 = (1+\sigma)t\hat{}+4 > t\hat{}$ bước; các bước bổ sung được thêm vào có liên quan đến chi phí xảy ra trong việc giảm tải sau này. Để đơn giản, chúng tôi xác định

$$t' = t + (2n + 4(p_2 + p_3)(nc_P + tc_V)) \cdot p_1 / c_H$$

là mức độ gian lận gia tăng mà đối thủ có được, do thực tế là việc giảm từ MH của tính toán DAG không chặt chẽ, chủ yếu là do chi phí tạo và xác minh các bằng chứng SNARG cho các tính toán DAG. Với tầm nhìn xa, chúng tôi cho rằng β là một ước tính về tốc độ mà đối thủ tạo ra các Ranking Block

$$\beta := \frac{p_2}{(1-\hat{\epsilon}(1,2,2n/t'))\hat{t} + (\frac{1}{(1+\sigma)p_1}+1)(c_H - 4p_1)}.$$

Như mong đợi β^{-1}, số bước dự kiến để tìm một Block, về cơ bản là số lần thử cần thiết để tìm một Post-Hash nhỏ ($1/p_2$), nhân với số bước cần thiết để tìm một Pre-Hash nhỏ (c_H/p_1) cộng với thời gian cần thiết để thực hiện phép tính DAG ($(1-\epsilon\hat{})t\hat{}$). Các hằng số khác của công thức liên quan đến phân tích bảo mật, tức là việc giảm của chúng tôi từ kẻ tấn công chống lại Blockchain thành kẻ tấn công chống lại tài sản MH. Cuối cùng, các tham số 0, 2, $2n/t'$ của $\epsilon\hat{}$ liên quan đến tốc độ truy vấn đối thủ $q_{\mathcal{O}}, q_{\mathcal{V}}, q_{\mathcal{M}}$ như được giải thích trong Phần 3.2.

Gọi r.v. $Z(S)$ biểu thị số lượng Block riêng biệt tối đa được tính toán bởi kẻ tấn công trong S, trong đó truy vấn Pre-Hash cho mỗi Block này cũng được cấp cho

RO trong *S*. Chúng tôi chứng minh rằng kẻ tấn công không thể khai thác các Ranking Block mới với tốc độ và xác suất tốt hơn so với việc phá vỡ thí nghiệm độ cứng vừa phải. Ý tưởng chứng minh chính là sử dụng một đối thủ tạo nhanh các Block, để tạo ra một đối thủ phá vỡ độ cứng vừa phải của *I*. Tóm tắt về ký hiệu của chúng tôi được đưa ra trong Bảng 1.

Bổ đề 13. *Với bất kỳ tập các vòng liên tiếp S, trong đó* $|S| \geq k\hat{}\ (1 + \mathring{\sigma})p_2 / (\beta \cdot t')$. *Nó cho rằng* $Z(S) \geq (1 + \sigma)\ \beta \cdot t' c_H|S|$ *với xác suất negl(λ).*

Chứng minh. Vì lợi ích của sự mâu thuẫn, giả sử rằng bổ đề không đúng. Điều này thể hiện tồn tại một khoảng vòng $S = \{i' | i \leq i' < i + s\}$ sao cho sự kiện *E* sau đây xảy ra với xác suất không đủ điều kiện trong λ: kẻ tấn công đã tính ít nhất $(1 + \sigma)\beta t' c_H|S|$ Block mới cho đến vòng $i + s$. Sử dụng *A*, chúng ta sẽ tạo ra một kẻ tấn công $A' = (A'_1, A'_2)$ phá vỡ độ cứng vừa phải (Định nghĩa 2) của *I* với xác suất không đáng kể. Đối với phần còn lại của chứng minh, hãy cho $m := (1 + \sigma)\beta \cdot t' c_H|S|$.

A' sẽ chạy bên trong *A* và *Z*, đồng thời mô phỏng công việc mà các bên trung thực thực hiện bằng cách sử dụng các phép thử *M* và *V* được cung cấp bởi thí nghiệm độ cứng vừa phải (Định nghĩa 2). Nó cũng sẽ mô phỏng nhanh các truy vấn RO do A_2 thực hiện và đưa ra các thách thức do Oracle $\mathcal{O}$ tạo ra để A_2 giải quyết. Bằng một lập luận hỗn hợp, chúng ta sẽ chỉ ra rằng khung nhìn của *A*, *Z* là không thể phân biệt được cả trong thực tế và chạy mô phỏng, và do đó xác suất xảy ra *E* sẽ giống nhau trong cả hai trường hợp.

Tiếp theo, chúng tôi mô tả hành vi của *A'* chi tiết hơn một cách riêng biệt cho cả hai giai đoạn. Đầu tiên, A'_1 đặt Ω làm đầu vào chung cho *A* và *Z*, trong đó Ω đã được tạo ra bằng cách sử dụng bộ tạo SNARG CRS S. Sau đó, nó mô phỏng hoàn hảo các bên trung thực lên đến vòng i - 1 và đồng thời chạy *A* và *Z* theo cách hộp đen. Để làm được điều đó, bất cứ khi nào *A* truy vấn RO, nó sẽ trả lời với một giá trị được lấy mẫu ngẫu nhiên; w.l.o.g, chúng tôi giả định trong phân tích rằng *A* không lặp lại cùng một truy vấn Oracle hai lần. Cuối cùng, nó xuất ra nội dung của các thanh ghi *A* và *Z*. Nó có thể làm được tất cả điều này, vì trong thí nghiệm độ cứng vừa phải, nó có thời gian đa thức trên λ tùy ý sử dụng. Lưu ý

rằng cho đến thời điểm này, đối với A và Z, việc thực thi mô phỏng không thể phân biệt được so với thực tế.

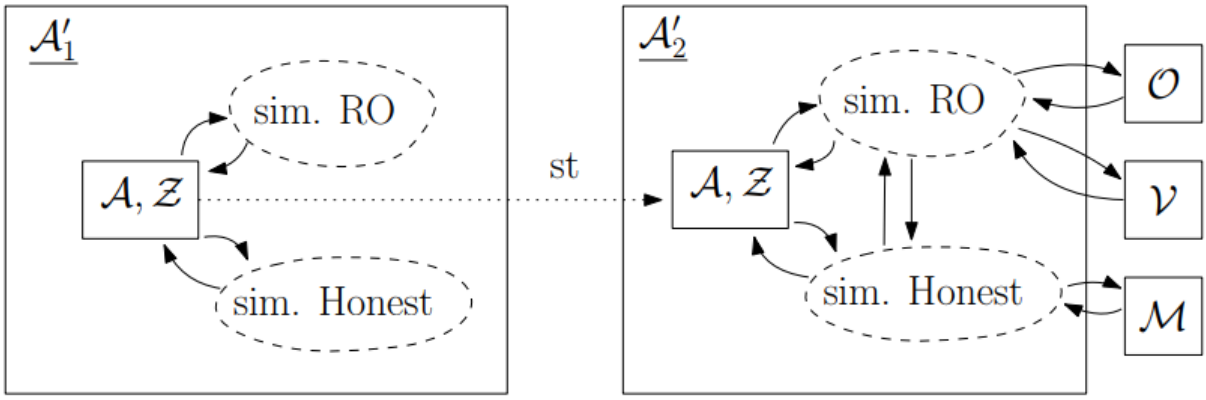
Đối với giai đoạn thứ hai, $A'_2(st)$, đầu tiên là sử dụng st để đặt lại A và Z về trạng thái trước đó của chúng. Chúng tôi giả định rằng điều này có thể được thực hiện một cách nhanh chóng, ví dụ, bằng cách để A và Z đọc từ các thanh ghi nơi st được lưu trữ bất cứ khi nào chúng thực hiện một số thao tác trên thanh ghi.

Tiếp theo, chúng tôi mô tả cách A'_2 mô phỏng các bên trung thực. Đối với mỗi bên trung thực P, đầu tiên nó lấy mẫu một giá trị l từ phân phối hình học với tham số p_2, số lượng truy vấn Post-Hash mà P phải thực hiện cho đến khi tìm thấy một Post-Hash nhỏ. Sau đó, A'_2 lấy mẫu l giá trị (t_i) với $i \in [l]$ từ hình học phân phối với tham số p_1. Các giá trị này là số lượng truy vấn Pre-Hash mà P phải thực hiện cho đến khi tìm thấy l Pre-Hash nhỏ. Tiếp theo, A'_2 truy vấn Oracle $\mathcal{M}$ l lần, với các đầu vào thích hợp (G_i, r_i). G_i được xác định bởi góc nhìn của P tại vòng mà nó được cho là sẽ thực hiện phép tính này, và r_i được lấy mẫu đồng nhất từ phạm vi của Hash. Cuối cùng, sau khi $\sum_{i \in [l]} ti + l \cdot (t/c_H + 1)$ vòng, A'_2 mô phỏng P bằng cách khuếch tán một Block mới bằng cách lập trình RO theo cách mà các thách thức r_i và các giá trị Pre-Hash và Post-Hash làm cho Block hợp lệ. Nó cũng phải tính toán bằng chứng SNARG cho phản hồi $\mathcal{M}$ giảm thiểu g và cho phản hồi $\mathcal{M}$ chiến thắng. Quá trình tương tự được lặp lại cho đến khi kết thúc khoảng thời gian S, đến vòng tiếp theo mà P khuếch tán một Block mới.

Trong khi mô phỏng các bên trung thực, A'_2 phải đối phó với lưu lượng mạng lưới đến từ A_2. Trong trường hợp tốt nhất, A'_2 phải xác minh một số bằng chứng SNARG bằng với số lượng Input và Ranking Block hợp lệ mà A_2 có thể tạo. Mặc dù A_2 có thể có khả năng lừa gạt các bên trung thực với các bằng chứng SNARG không chính xác, chúng tôi lưu ý rằng các cuộc tấn công như vậy có thể được xử lý bằng cách yêu cầu các bên tạo ra một PoW dựa trên Hash bổ sung dựa trên SNARG và Block liên quan. Độ cứng của PoW nên được thiết lập thích hợp, để một mặt, kẻ tấn công chỉ có thể tạo ra các cặp SNARG / PoW hợp lệ ở một tỷ lệ

giới hạn. Mặt khác, các bên trung thực không chi tiêu quá nhiều sức mạnh của chúng khi Hash so với chạy *M*. Để đơn giản trong phân tích, chúng tôi giả định rằng A_2 không khuếch tán các Block không hợp lệ cho mạng lưới.

A'_2 cũng mô phỏng các truy vấn tới RO được thực hiện bởi *A* theo một cách khác so với trước đây. Mỗi lần *A* truy vấn RO với chuỗi *w*, A'_2 đầu tiên sẽ kiểm tra xem $H(w)$ có phản hồi được mã hóa cứng hay không. Nếu có, nó trả về giá trị đã được mã hóa cứng trước đó. Ngược lại, (i) nếu *w* tương ứng với một chuỗi Pre-Hash, nó lấy mẫu và xuất ra một chuỗi ngẫu nhiên trong phạm vi của Hash và nếu giá trị nhỏ hơn T_1, nó sẽ truy vấn Oracle $\mathcal{O}$ và mã hóa phản hồi dưới dạng giá trị của[5] $H(w||H(w))$, (ii) nếu *w* là một chuỗi Post-Hash, nó sử dụng Oracle $\mathcal{V}$ để kiểm tra xem tính toán DAG được mô tả trong *w* có hợp lệ không và nếu có, nó sẽ lưu trữ. Sau đó, nó lấy mẫu một lần nữa và trả về một chuỗi ngẫu nhiên. Trong cả hai trường hợp, nếu chuỗi nhỏ hơn các mục tiêu tương ứng T_1, T_2, T_3, kết quả được lưu trữ và sử dụng trong mô phỏng của các bên trung thực để xác định xem một Block có hợp lệ hay không.



Hình 3: Một sơ đồ của quá trình giảm. *A'* mô phỏng cả các bên trung thực và các truy vấn tới RO. Trong giai đoạn thứ hai của quá trình giảm, *A'* sử dụng các Oracle mà nó có được trong thí nghiệm MH. Nó sử dụng: $\mathcal{O}$ để đưa ra các thử thách mới cho *A, V* để nhanh chóng xác minh tính hợp lệ của các tính toán DAG được thực hiện bởi *A, M* để nhanh chóng mô phỏng các tính toán DAG được thực hiện bởi các bên trung thực.

[5]Luôn có thể mã hóa cứng này, vì xác suất *A* truy vấn $H(w||H(w))$ trước khi truy vấn $H(w)$ là không đáng kể trong λ.

Tiếp theo, chúng tôi phân tích xác suất *A'* phá vỡ độ cứng vừa phải của *I*. Đầu tiên, hãy lưu ý *A* và *Z* không thể phân biệt giữa thực thi thực tế và thực thi mô phỏng mà chúng tôi đã mô tả ở trên, vì các truy vấn đến RO được mô phỏng hoàn hảo và các bên trung thực được mô phỏng hoàn hảo sử dụng Oracle $\mathcal{M}$. Do đó, *E* cũng sẽ xảy ra trong thực thi mô phỏng với xác suất không đáng kể, tức là *A* sẽ tính ít nhất *m* Block mới bắt đầu từ vòng *i* trở lên đến vòng *i* + *s*. Điều này thể hiện *A* sẽ đưa ra ít nhất *m* truy vấn Post-Hash hợp lệ nhỏ hơn T_2. Chúng tôi sẽ sử dụng thực tế này để giới hạn xác suất thắng của *A'*. Do quy luật xác suất toàn phần, chúng ta có:

$$\begin{aligned}&\Pr[\mathcal{A}' \text{ breaks MH}] = \\ &= \Pr[\mathcal{A}' \text{ breaks MH} \wedge E] + \Pr[\mathcal{A}' \text{ breaks MH} \wedge \neg E] \\ &\geq \Pr[\mathcal{A}' \text{ breaks MH} | E] \Pr[E] .\end{aligned}$$

Vì Pr[*E*] là không đáng kể, tiếp theo chúng tôi tập trung vào giới hạn dưới Pr[*A'* phá vỡ MH|*E*]. Gọi q_H là số truy vấn *A* được thực hiện đối với mô phỏng RO. Để đơn giản, chúng tôi giả định chi phí mô phỏng một truy vấn RO như được mô tả ở trên là không đáng kể so với c_H, chi phí của một truy vấn RO trong quá trình thực thi thực tế. Bây giờ, hãy để biến ngẫu nhiên *U* (tương ứng với *U'*) biểu thị số lượng Ranking và Input Block được tạo bởi các bên trung thực (tương ứng với kẻ tấn công) trong *S*. Cũng tính đến các lệnh gọi Oracle được thực hiện bởi A'_2, tổng số bước A'_2 thực hiện bất kỳ, ký hiệu là $\mathbf{Steps}_{A'2}$, tối đa là:

$$\mathbf{Steps}_{A'2} \leq s \cdot t c_H - q_H c_H + (q_{\mathcal{O}} + q_{\mathcal{V}} + q_{\mathcal{M}}) + U c_P + U' c_V. \quad (2)$$

Tiếp theo, chúng ta chứng minh rằng $q_{\mathcal{O}}$, $q_{\mathcal{V}}$, $q_{\mathcal{M}}$, tức là số lượng truy vấn được thực hiện đối với Oracles $\mathcal{O}$, $\mathcal{V}$, $\mathcal{M}$ cũng như *U*, *U'*, được giới hạn trên như sau:

Tuyên bố 1. Điều này cho rằng $q_{\mathcal{V}}$, $q_{\mathcal{O}} < 2q_H p_1$, $q_{\mathcal{M}} < 2p_1 ns$ và $U < 4p_1(p_2 + p_3)ns$, $U' < 4p_1(p_2 + p_3)ts$ với xác suất áp đảo trong λ.

Chứng minh. Đầu tiên chúng ta phân tích giới hạn $q_{\mathcal{M}} < 2p_1 ns$. Nếu một số P_j đã chạy xong một lệnh gọi *M*, nó sẽ là trường hợp một Pre-Hash nhỏ đã

được tính toán trước đó. Do đó, chúng ta có thể giới hạn trên q$\mathcal{M}$, bằng cách tính toán giới hạn trên về số lượng Pre-Hash mà P_j đã tính toán thành công. Vì trong mỗi vòng, P_j có thể thực hiện nhiều nhất một truy vấn Hash, nên nó thành công với xác suất p_1. Cho r.v. $X_{i,j}$ bằng 1 với xác suất p_1, và ngược lại thì bằng 0.

Chúng ta có thể giới hạn trên tổng số các Pre-Hash nhỏ được các bên trung thực tính toán bởi $X = \sum_{i \in S, j \in [n]} Xi,j$. Bởi một ứng dụng của Chernoff dễ dàng ràng buộc $\Pr[X \geq 2p_1ns] < negl(\lambda)$.

Theo cách tương tự, chúng ta có thể giới hạn trên số lượng Block được tạo bởi các bên trung thực, (và do đó số SNARG được tính toán). Cho biến ngẫu nhiên R'_i bằng 1, nếu lệnh gọi thứ i của M được thực hiện bởi một bên trung thực trong S dẫn đến việc tạo ra một Input hoặc Ranking Block mới. Theo giới hạn trước đó, chúng ta có thể xác định $R' := \sum_{i \in 2p1ns]} R'i$, trong đó $E[R'] \leq 2p_1ns(p_2 + p_3)$ và R' là giới hạn trên về số Block được tạo bởi các bên trung thực trong S với sự áp đảo xác suất. Một lần nữa sử dụng giới hạn Chernoff, ta có $U < 4p_1(p_2 + p_3)ns$ với xác suất áp đảo trong λ.

Để thiết lập ràng buộc q$\mathcal{V}$, q$\mathcal{O} < 2q_Hp_1$, lưu ý đầu tiên q$\mathcal{V}$, q$\mathcal{O} \leq q_{pre}$, bởi thực tế là Oracle $\mathcal{O}$ được truy vấn bất cứ khi nào một truy vấn Pre-Hash thành công hợp lệ mới được thực hiện và các truy vấn Post-Hash hợp lệ riêng biệt phải chứa Pre-Hash nhỏ hợp lệ riêng biệt. Đặt R^*_i bằng 1 nếu truy vấn băm phân biệt thứ i là một trước băm hợp lệ. Cho $R^* := \sum_{i \in [qH]} R^*_i$. Nó cho rằng $E[R^*] \leq q_Hp_1$. Hơn nữa, các biến ngẫu nhiên trong $\{R^*_i | i \in [q_H]\}$ là độc lập với nhau. Do đó, chúng ta có thể áp dụng giới hạn Chernoff (w.l.o.g. cũng giả sử rằng $q_H > \lambda$):

$$\Pr[R^* \geq (1+\sigma)q_Hp_1] \leq \Pr[R^* \geq (1+\sigma)q_Hp_1]$$
$$< e^{-q_Hp_1\sigma^2/3} < negl(\lambda)\,.$$

Tiếp theo là với xác suất áp đảo q$\mathcal{V}$, q$\mathcal{O} \leq q_{pre} < 2q_Hp_1$. Theo cách tương tự như trước, chúng ta có thể ràng buộc U' bởi $4q_{qpre}(p_2 + p_3) \leq 4q_Hp_1(p_2 + p_3) \leq 4p_1(p_2 + p_3)ts$, vì $q_H < ts$.

Gọi D_1 là sự kiện $q_{\mathcal{V}}, q_{\mathcal{O}} \leq q_{\text{pre}} < 2q_H p_1, q_{\mathcal{M}} < 2p_1 ns$ và $U < 4p_1(p_2 + p_3)ns$, $U' < 4p_1(p_2 + p_3)ts$. Nếu D_1 giữ nguyên, theo Bất đẳng thức 2, chúng ta có:

$$\begin{aligned}\mathsf{Steps}_{\mathcal{A}_2'} &\leq s \cdot tc_H - q_H c_H + 4q_H p_1 \\ &\quad + sp_1(2n + 4(p_2 + p_3)(nc_{\mathsf{P}} + tc_{\mathsf{V}})) \\ &= s \cdot tc_H - q_H(c_H - 4p_1) \\ &\quad + sp_1(2n + 4(p_2 + p_3)(nc_{\mathsf{P}} + tc_{\mathsf{V}}))\,.\end{aligned}$$

Theo định nghĩa của β, chúng ta có $1/\beta \leq \frac{CH}{p1p2}$. Cũng bằng cách sử dụng khái niệm của m, chúng ta có thể ràng buộc tốc độ mà tại đó A' truy vấn Oracle $\mathcal{V}$ và $\mathcal{M}$:

$$\frac{q_{\mathcal{V}}}{m/((1+\sigma)p_2)} \leq \frac{2p_1p_2q_H}{\beta t' c_H s} \leq \frac{2p_1p_2st}{p_1p_2/c_H \cdot t'c_H s} \leq 2$$

và

$$\frac{q_{\mathcal{M}}}{m/((1+\sigma)p_2)} \leq \frac{2p_1p_2ns}{\beta t' c_H s} \leq \frac{2p_1p_2ns}{p_1p_2/c_H \cdot t'c_H s} \leq 2n/t'\,.$$

Đối với phần còn lại của chứng minh, chúng ta sử dụng $\epsilon'(c)$ thay vì $\epsilon'(1 + c/(m/((1 + \sigma)p_2), 2, 2n/t')$, tức là c biểu thị số lượng truy vấn bổ sung được làm cho Oracle $\mathcal{O}$. Theo định nghĩa của m, chúng ta có:

$$\begin{aligned}&s \cdot tc_H \leq m/((1+\sigma)\beta) - sp_1(2n + 4(p_2 + p_3)(nc_{\mathsf{P}} + tc_{\mathsf{V}})) \\ &\leq \frac{m}{(1+\sigma)p_2}\left((1 - \hat{\epsilon}(0))\hat{t} + \left(\frac{1}{(1+\sigma)p_1} + 1\right)(c_H - 4p_1)\right) \\ &- sp_1(2n + 4(p_2 + p_3)(nc_{\mathsf{P}} + tc_{\mathsf{V}}))\,.\end{aligned}$$

Kết hợp mọi thứ lại với nhau, chúng ta nhận được:

$$\begin{aligned}\mathsf{Steps}_{\mathcal{A}_2'} &\leq \frac{m}{(1+\sigma)p_2}(1 - \hat{\epsilon}(0))\hat{t} \\ &+ \left(\frac{m}{(1+\sigma)p_2}\left(\frac{1}{(1+\sigma)p_1} + 1\right) - q_H\right)(c_H - 4p_1)\,.\end{aligned}$$

Hơn nữa, do Bổ đề 3 và bất đẳng thức trên, nó cho rằng với mọi $c \geq 0$:

$$\mathsf{Steps}_{\mathcal{A}'_2} \leq \frac{m}{(1+\sigma)p_2}(1-\hat{\epsilon}(c))\hat{t} \tag{3}$$
$$+\left(\left(\frac{\frac{m}{(1+\sigma)p_2}+c}{(1+\sigma)p_1}+\frac{m}{(1+\sigma)p_2}\right)-q_H\right)(c_H-4p_1)\,.$$

Nó vẫn để chứng tỏ rằng số hạng thứ hai trong vế phải của bất đẳng thức là 0. Bước đầu tiên, chúng ta thiết lập cácgiới hạn thấp hơn về số lượng truy vấn Pre-Hash và Post-Hash cho RO do *A*. Đặt $q_{\mathrm{H,pre}}$, $q_{\mathrm{H,post}}$ biểu thị số lượng nhỏ hợp lệ riêng biệt, tức là nhỏ hơn T_1 và T_2, các giá trị Pre-Hash và Post-Hash được tính bởi *A*, tương ứng (không nên nhầm lẫn với q_{pre}, q_{post} các trạng thái của chuỗi Markov được trình bày trước đó). Gọi $q_{\mathrm{H,pre}}$, $q_{\mathrm{H,post}}$ biểu thị số lượng truy vấn Pre-Hash và Post-Hash hợp lệ khác nhau được thực hiện cho RO, tương ứng.

Yêu cầu 2. Với σ ∈ (0, 1) và bất kỳ $x \geq \lambda$, ta có:

- $\Pr[q_{post} \geq x \wedge q_{H,post} \leq \frac{x}{p_2(1+\sigma)}] \leq e^{-\Omega(\lambda)}$;
- $\Pr[q_{pre} \geq x \wedge q_{H,pre} \geq \frac{x}{p_1(1+\sigma)}] \leq e^{-\Omega(\lambda)}$.

Chứng minh. Đầu tiên, chúng ta ràng buộc xác suất *A* tạo ra *x* (≥ λ) hoặc nhiều Block hơn bằng cách truy vấn RO nhỏ hơn $\frac{x}{\mathrm{p2}(1+\sigma)}$ lần với các truy vấn Post-Hash. W.l.o.g., giả sử $q_{\mathrm{H,post}} = \frac{x}{\mathrm{p2}\ (1+\sigma)}$. Đặt R_i bằng 1 nếu truy vấn Post-Hash thứ *i* thành công, tức là nhỏ hơn T_2, và bằng 0 trong trường hợp khác. Cho R := $\sum_{\mathrm{i}\in[\mathrm{qH,post}]} Ri$, trong đó $q_{\mathrm{H,post}}$ là số truy vấn Post-Hash riêng biệt. Ta có $q_{\mathrm{post}} \leq R$ và E[*R*] = $q_{\mathrm{H,post}} \cdot$ p2 = *x*/(1 - σ). Bởi một ứng dụng của Chernoff đã ràng buộc:

$$\Pr[R \geq x] = \Pr[R \geq (1+\sigma)\mathbb{E}[R]] \leq e^{-\frac{\mathbb{E}[R]\sigma^2}{3}} = e^{-\frac{x\sigma^2}{3(1+\sigma)}} \leq negl(\lambda)\,.$$

Do đó, trường hợp $q_{\mathrm{post}} \geq x$ và $q_{\mathrm{H,post}} \leq \frac{x}{\mathrm{p2}(1+\sigma)}$ xảy ra với xác suất *negl*(λ).

Tương tự, chúng ta ràng buộc xác suất mà *A* tính toán nhiều hơn *x* (≥ λ) số Pre-Hash trong ít hơn $\frac{x}{(1+\sigma)\mathrm{p1}}$ truy vấn RO Pre-Hash. Đặt R'_i bằng 1 nếu truy vấn Pre-Hash thứ *i* thành công, sau đó thực hiện theo một suy luận tương tự như trước đây chúng ta có:

$$\Pr[R' \geq x] \leq \Pr[R' \geq (1+\sigma)\mathbb{E}[R']] \leq e^{-\frac{\mathbb{E}[R']\sigma^2}{3}}$$

$$= e^{-\frac{x\sigma^2}{(1+\sigma)3}} = e^{-\Omega(\lambda)}.$$

Yêu cầu sau.

Gọi D_2 là biến cố mà $q_{\text{post}} \geq m$ => $q_{\text{H,post}} > \frac{m}{p2(1+\sigma)}$ và với mọi $c \leq q_{\text{H}}$, $q_{\text{pre}} \geq \frac{m}{p2(1+\sigma)} + c$ => $q_{\text{H,pre}} > \frac{\frac{m}{p_2(1+\sigma)}+c}{p_1(1+\sigma)}$.Theo yêu cầu trước đó và ứng dụng liên hợp ràng buộc, chúng ta nhận được $\Pr[D_2] \geq 1 - negl(\lambda)$.

Vì mỗi D_1 và D_2 xảy ra với xác suất áp đảo, và E xảy ra với xác suất không đáng kể, theo định nghĩa của xác suất có điều kiện, nên $D_1 \wedge D_2|E$ xảy ra với xác suất áp đảo:

$$\begin{aligned}
&\Pr[D_1 \wedge D_2|E] = 1 - \Pr[\neg(D_1 \wedge D_2)|E] \\
&= 1 - \frac{\Pr[\neg(D_1 \wedge D_2) \wedge E]}{\Pr[E]} \\
&\geq 1 - \frac{\Pr[(\neg D_1) \vee (\neg D_2)]}{\Pr[E]} \\
&\geq 1 - \frac{\Pr[\neg D_1] + \Pr[\neg D_2]}{\Pr[E]} = 1 - negl(\lambda)
\end{aligned}$$

trong đó bất đẳng thức cuối cùng xuất phát từ một ứng dụng liên hợp ràng buộc.

Tiếp theo, chúng tôi sẽ chỉ ra rằng điều kiện ở E, $D_1 \wedge D_2$ thể hiện *A'* thắng trong trò chơi MH. Thứ nhất, E thể hiện A đã đưa ra m truy vấn Post-Hash nhỏ hợp lệ. Theo đó $q_{\text{post}} \geq m$, do đó D_2 thể hiện $q_{H,post} > \frac{m}{p2(1+\sigma)}$. Vì mỗi cặp (G, r) xác định một đầu ra x duy nhất, nên mỗi truy vấn Pre-Hash xác định chính xác một truy vấn Post-Hash (ngoại trừ xác suất không đáng kể). Do đó, $q_{pre} \geq q_{H,post} > \frac{m}{p2(1+\sigma)}$.

Cho $q_{pre} := \frac{m}{p2(1+\sigma)} + c$, với $c \geq 0$. Đối với mỗi truy vấn Pre-Hash nhỏ, một truy vấn tới Oracle $\mathcal{O}$ phải được đưa ra, do đó $q\mathcal{O} := \frac{m}{p2(1+\sigma)} + c$. Cuối cùng, một lần nữa với D_2, chúng ta nhận được $q_{H,pre} \geq \frac{\frac{m}{p_2(1+\sigma)}+c}{p_1(1+\sigma)}$.

Hãy lưu ý các truy vấn Pre-Hash và Post-Hash là khác nhau, do đó $q_H \geq q_{H,pre} +$

$q_{H,post}$, và theo phân tích trước đây, A'_2 đã đưa ra ít nhất

$$\frac{\frac{m}{(1+\sigma)p_2}+c}{(1+\sigma)p_1}+\frac{m}{(1+\sigma)p_2}$$

truy vấn RO. Hơn nữa, đối với mỗi truy vấn Post-Hash nhỏ riêng biệt A'_2 đã trích xuất một tính toán DAG hợp lệ riêng biệt trên một thử thách do Oracle $\mathcal{O}$ đưa ra, tức là $\frac{m}{p2(1+\sigma)}$ ($\geq \beta t' c_H|S| / (1+\sigma)p_2 \geq k$^) tính toán DAG hợp lệ. Theo ràng buộc với q_H và Bất đẳng thức 3, ta có $\mathbf{Steps}_{A'2} \leq (1 - \epsilon$^$(c))\ t$^$\frac{m}{p2(1+\sigma)}$. Do đó, A' thắng trong trò chơi MH, vì nó đã thực hiện được $\frac{m}{p2(1+\sigma)}$ ($\geq k$^) các phép tính DAG hợp lệ trong tối đa (1 - ϵ^$(c))\ t$^$\frac{m}{p2(1+\sigma)}$ bước và truy vấn Ooracle $\mathcal{O}$ nhiều nhất là $\frac{m}{p2(1+\sigma)} + c$ lần. Như vậy:

$$\Pr[\mathcal{A}' \text{ breaks MH}|E] \geq \Pr[D_1 \wedge D_2|E] \geq 1 - negl(\lambda)$$

Điều đó thể hiện

$$\Pr[\mathcal{A}' \text{ breaks MH}] \geq \Pr[\mathcal{A}' \text{ breaks MH}|E]\Pr[E] > negl(\lambda)$$

vì cả hai phần của sản phẩm đều không đáng kể. Điều này mâu thuẫn với Giả định 1, và bổ đề sau.

5.1.5. Kết hợp mọi thứ lại với nhau

Tiếp theo, chúng ta thấy xác suất xảy ra một vòng thành công duy nhất lớn hơn tỷ lệ khai thác đối thủ dự kiến cho mỗi vòng. Hướng tới mục đích này, giả định tiếp theo đảm bảo lợi thế về sức mạnh tính toán của các bên trung thực tốt hơn lợi thế về độ cứng vừa phải trong tính toán DAG của kẻ tấn công, đồng thời tốc độ các Block được tạo ra có mức giới hạn trên.

Giả định 3. Tồn tại δ_{MH}, δ_{Steps} và $\delta_{tot} \in (0, 1)$, sao cho đủ lớn $\lambda \in \mathbb{N}$:

- $(n - t)\ (1 - \delta_{Steps}) \geq t'$ (khoảng cách giữa các bước)
- $p^*_{rank} \geq (1 - \delta_{MH})\beta c_H$ (khoảng cách MH)
- $\delta_{Steps} - \delta_{MH} \geq \delta_{tot}$ (khoảng cách giữa các bước so với khoảng cách MH)
- $\delta_{tot} > 3\Gamma \cdot \beta c_H(n - t)$ (tỷ lệ Block được giới hạn)

Như đã có, và dựa trên Giả định 3, chúng ta chứng minh bổ đề sau.

Bổ đề 14. Nó cho rằng $p_{iso} > (1 + \delta_{tot})\ \beta t' c_H$.

Chứng minh. Chúng ta có:

$$
\begin{aligned}
p_{\text{iso}} &\geq (n-t)(1-(3\Gamma)p_1p_2)^{(n-t)}p^*_{\text{rank}} && \text{(Bernoulli)}\\
&> (n-t)(1-(3\Gamma)p_1p_2\cdot(n-t))p^*_{\text{rank}} && \text{(Tỷ lệ Block)}\\
&\geq (n-t)(1-\delta_{tot})p^*_{\text{rank}} && \text{(Khoảng cách MH)}\\
&\geq (n-t)(1-\delta_{tot})(1-\delta_{\mathsf{MH}})\beta c_H && \text{(K/cách giữa các bước)}\\
&\geq \frac{(1-\delta_{tot})(1-\delta_{\mathsf{MH}})}{1-\delta_{\mathsf{Steps}}}t'\beta c_H && \text{(Tỷ lệ Block)}\\
&\geq (1+\delta_{tot})\beta t' c_H\ .
\end{aligned}
$$

trong đó bất đẳng thức đầu tiên cũng xuất phát từ thực tế là $p_1p_2 \leq \beta c_H$.

Cùng với các giới hạn nồng độ thích hợp được chứng minh trong Bổ đề 13 và Bổ đề 12, Bổ đề 14 phù hợp để áp dụng Định lý 7 cho Π, do đó Π thỏa mãn cả Tính bền bỉ và Tính tồn tại với xác suất áp đảo. Cuối cùng, trong Phụ lục B, chúng tôi lập luận rằng trong điều kiện lý tưởng, tức là MH tối ưu, chi phí SNARG nhỏ, v.v., Π có thể chịu đựng mọi thiểu số không trung thực.

Xử lý chi tiết hơn về thời gian hoàn thành công việc hữu ích. Phân tích ở trên hiệu chỉnh độ cứng Pre-Hash như một hàm của $t\hat{}$, thời gian hoàn thành trong trường hợp xấu nhất của công việc hữu ích. Trong một số cài đặt nhất định, độ phức tạp về thời gian của nhiệm vụ công việc hữu ích có thể thỏa mãn một giới hạn rõ ràng mạnh hơn đáng kể với xác suất rất cao. Trong trường hợp đó, giới hạn giảm này có thể thay thế cho $t\hat{}$ chỉ với những thay đổi nhỏ nhất đối với sự phát triển ở trên. Cụ thể, nếu độ phức tạp thời gian là $\bar{t} < t\hat{}$ trừ phi với xác suất không đáng kể, thì giá trị $\bar{t}$ có thể được thay thế đồng nhất cho $t\hat{}$ ở trên với việc cộng thêm sai số không đáng kể các thuật ngữ trong các định lý trên.

5.2. Bảo mật DPLS

Việc thực thi DPLS trong cài đặt PoUW không được phép có khả năng thể hiện sự tham gia đáng kể của kẻ tấn công, điều này có thể gây ảnh hưởng tiêu cực đến hiệu suất của thuật toán theo nhiều cách. Cụ thể, kẻ tấn công không

phải tuân theo Thuật toán 1, ví dụ: bằng cách công bố kết quả của việc thực hiện *M* tồi tệ nhất, thay vì kết quả tốt nhất.

Mặc dù việc trình bày DPLS không có khả năng tham gia của kẻ tấn công, chúng tôi cung cấp các biện pháp bảo vệ tương ứng trong giao thức nhúng PoUW của nó. Chúng tôi đưa ra hai đảm bảo chất lượng quan trọng của triển khai DPLS bằng giao thức PoUW miễn là kẻ tấn công chỉ kiểm soát một phần nhỏ sức mạnh tính toán: (i) trong bất kỳ khoảng thời gian vòng đủ lớn nào, các bên trung thực đóng góp các cập nhật mới tương ứng với sức mạnh khai thác tương đối, cụ thể là các bên trung thực đóng góp cập nhật nhiều hơn kẻ tấn công; (ii) kẻ tấn công không thể thao túng rộng rãi điểm số của các bản cập nhật, vì chúng tôi thực thi mỗi bản cập nhật bao gồm thêm kết quả của việc thực hiện "ngẫu nhiên" *M* từ lô, được tính đến nếu lần thực hiện tốt nhất có điểm kém hơn.

Tiếp theo, chúng tôi tiến hành chính thức hóa các thuộc tính nói trên. Chúng tôi bắt đầu với quan sát về tốc độ mà kẻ tấn công tạo ra các bản cập nhật trong ngữ cảnh của giao thức DPLS, tức là tổng số Input và Ranking Block của kẻ tấn công. Gọi $\bar{Z}(S)$ là phần mở rộng tự nhiên của $Z(S)$ đề cập đến cả Input và Ranking Block, và

$$\bar{\beta} := \frac{p_2 + p_3}{(1 - \hat{\epsilon}(1, 2, 2n/t'))\hat{t} + (\frac{1}{(1+\sigma)p_1} + 1)(c_H - 4p_1)}.$$

Thực hiện theo các bước tương tự như trong Bổ đề 13, chúng tôi nhận được:

Bổ đề 15. *Với bất kỳ tập hợp các vòng liên tiếp S, trong đó* $|S| \geq k\hat{}(1 + \sigma)(p_2 + p_3) / (\bar{\beta} \cdot t'c_H)$, *ta có* $\bar{Z}(S) \geq (1 + \sigma)\bar{\beta} \cdot t'c_H|S|$ *với xác suất negl(λ).*

Sử dụng bổ đề trước, chúng tôi có thể chỉ ra rằng số lượng bản cập nhật được tạo ra bởi các bên trung thực tốt hơn kẻ tấn công.

Bổ đề 16. *Với bất kỳ tập các vòng liên tiếp S, trong đó* $|S| \geq k\hat{}(1 + \sigma)(p_2 + p_3) / (\bar{\beta} \cdot t'c_H)$, *nó thể hiện các bên trung thực tạo ra nhiều thông tin cập nhật hơn kẻ*

tấn công với xác suất áp đảo.

Chứng minh. Chúng ta có thể giới hạn dưới tỷ lệ tạo ra các Block trung thực dựa trên xác suất cố định p^*, một bên trung thực đang ở trạng thái sản xuất Input Block hoặc Ranking Block. Chúng ta có thể tính p^* tương tự như cách tính p^*_{rank} trong Phụ lục B. Chúng ta nhận được:

$$p^* := \frac{p_2 + p_3}{1/p_1 + 1 + \hat{t}/c_H + c_{\mathsf{P}}/c_H(p_2 + p_3)} .$$

Dựa trên giả định 3, chúng ta có tỷ lệ kỳ vọng mà tại đó các bên trung thực tạo ra Block tốt hơn so với kẻ tấn công, tức là, $(n - t)p^* \geq (1 - \delta_{\text{tot}})t'\bar{\beta}c_H$. Bổ đề được chứng minh dễ dàng.

Tiếp theo, chúng ta chuyển sự chú ý đến điểm số các bản cập nhật Block của kẻ tấn công. Vì giao thức Blockchain quy định rằng nếu điểm của nỗ lực chiến thắng lớn hơn điểm của nỗ lực tốt nhất trong một Block, thì nỗ lực chiến thắng sẽ được xem xét trong DPLS, chúng ta dễ dàng nhận được hệ quả sau.

Hệ quả 17. *Đối với bất kỳ tập hợp các vòng liên tiếp S, điểm của lần cập nhật đối thủ thứ i tương ứng với Block thứ i trong $\bar{Z}(S)$, lớn hơn hoặc bằng điểm của lần chạy M tương ứng cho nỗ lực Post-Hash nhỏ có liên quan.*

Để ảnh hưởng tiêu cực đến chất lượng của các bản cập nhật, lựa chọn tốt nhất có thể của kẻ tấn công là không sử dụng một số bản cập nhật trong số đó, với chi phí phần thưởng giảm và ảnh hưởng đến giao thức. Nếu không, các bản cập nhật sẽ không tệ hơn nhiều so với các lần chạy đồng nhất của *M*. Cuối cùng, hãy lưu ý rằng do thiết kế Pipelining được trình bày tại Phần 4.2, kẻ tấn công chỉ có một cửa sổ giới hạn bằng một Epoch để chọn Block nào trong số các Block để phát hành.

5.3. Tính hữu ích của giao thức

Mục tiêu của bất kỳ giao thức Blockchain nào dựa trên PoUW là được sử dụng để giải quyết một số vấn đề bên ngoài của Blockchain, có độ khó vừa phải và vấn đề

tính toán. Chúng ta nói rằng một giao thức có *tỷ lệ hữu ích* cao nếu tổng công việc tính toán được dành cho việc vận hành Blockchain và giải quyết vấn đề bên ngoài không lớn hơn nhiều so với việc chỉ giải quyết vấn đề bằng thuật toán tốt nhất cho cài đặt mà chúng ta xem xét, ký hiệu là A_{best}.

Chúng tôi nghiên cứu tỷ lệ hữu ích của giao thức bằng cách sử dụng hai chỉ số. Chỉ số đầu tiên là U_{eng} để đo tỷ lệ tổng thể của các bước tính toán mà công cụ hướng đến việc chạy thuật toán DPLS. Chỉ số này nắm bắt được giao thức hoạt động như thế nào một cách trực quan với phương diện là một công cụ DPLS. Chúng tôi tính tổng U_{eng} như sau (giả sử rằng thời gian chạy của M là cố định):

$$\begin{aligned} \mathsf{U}_{\mathrm{eng}} &:= \mathbb{E}[\text{DPLS steps per block}]/\mathbb{E}[\text{total steps per block}] \\ &= \frac{\hat{t}/(p_2+p_3)}{\hat{t}/(p_2+p_3) + 2\cdot c_{\mathsf{P}} + c_H/(p_1(p_2+p_3))} \\ &= [1 + 2\cdot c_{\mathsf{P}}\cdot(p_2+p_3)/\hat{t} + c_H/p_1 \cdot 1/\hat{t}]^{-1} < 1/2 \end{aligned}$$

trong đó bất đẳng thức cuối cùng xuất phát từ thực tế là thời gian dự kiến để tạo ra một Pre-Hash nhỏ (c_H/p_1) gần với thời gian giải quyết của M là $t\hat{}$. Hơn nữa, chi phí chứng minh SNARG (c_P) càng nhỏ so với công việc DPLS dự kiến để tạo ra một Block ($t\hat{}/(p_2 + p_3)$), thì U_{eng} càng gần 1/2. Do đó, theo thiết kế, giao thức sẽ quản lý để hướng khoảng 1/2 công việc để tạo Block tới thuật toán DPLS. Hãy lưu ý rằng tỷ lệ này có thể được cải thiện bằng cách tính đến các giới hạn bảo mật chính xác liên quan đến các cuộc tấn công nghiền nát đối với tính toán DAG vừa phải (xem Chú thích 2).

Chỉ số thứ hai là U_{alg} dùng để so sánh độ phức tạp của DPLS với thuật toán A_{best}. Lưu ý rằng đối với U_{alg}, chúng tôi chỉ tính đến các bước tính toán DPLS và không tính đến các bước khác liên quan đến giao thức, ví dụ như Hash và các tính toán SNARG.

U_{alg} := *E* [tổng số bước của A_{best}] / E[tổng số bước của DPLS]

Không thể nghiên cứu U_{alg} một cách chung chung vì nó phụ thuộc vào vấn đề cụ

thể bên ngoài được giải quyết cũng như mô hình tính toán mà chúng tôi xem xét. Ví dụ: chúng tôi mong đợi U_{alg} sẽ lớn hơn nhiều khi xem xét thuật toán tốt nhất trong cài đặt phân tán so với thuật toán tốt nhất trong cài đặt máy đơn lẻ. Thay vào đó, trong Phần 6, chúng tôi thực hiện phân tích thử nghiệm U_{alg} cho một biến thể DPLS của WalkSAT.

Hai chỉ số mà chúng tôi đã giới thiệu ghi lại cả chi phí liên quan đến giao thức sổ cái (Hash và SNARGS) và chi phí do thuật toán cụ thể mà chúng tôi triển khai. Trên thực tế, trong trường hợp các Block được tính bằng thuật toán khai thác trung thực, tích của hai chỉ số là một giá trị xấp xỉ tốt của tỷ lệ hữu ích.

Nhận xét 2. (U_{eng} được cải thiện) Để có thể chứng minh tính bảo mật của giao thức với bất kỳ tính toán DAG nào, chúng tôi đặt độ cứng Pre-Hash (T_1) xấp xỉ bằng độ phức tạp thời gian trong trường hợp xấu nhất của thuật toán thăm dò *M*. Điều này có tác dụng là U_{eng} nhỏ hơn 1/2, hãy lưu ý rằng có thể cải thiện U_{eng} bằng cách điều chỉnh độ cứng Pre-Hash dựa trên MH của tính toán DAG cụ thể được xem xét. Đặc biệt, để các đối số bảo mật vẫn hợp lệ, nó đề nghị rằng:

$$c_H/p_1 := \max_{m,a} \{a \cdot (\hat{\epsilon}(1+a, 2, 2n/t') - \hat{\epsilon}(1, 2, 2n/t'))\hat{t}\} \ .$$

Hiện tại, một phép tính MH DAG trong đó $\hat{\epsilon}(1 + a, 2, 2n/t') - \hat{\epsilon}(1, 2, 2n/t') \leq \delta a$, với một số $\delta \in [0, 1]$, chúng ta nhận được

$$U_{eng} = [1 + 2 \cdot c_P \cdot (p_2 + p_3)/\hat{t} + \delta]^{-1}$$

đối với các tham số thuận lợi, tức là, $c_P << \hat{t}/(p_2 + p_3)$ và $\delta \approx 0$, thể hiện rằng U_{eng} gần bằng 1.

6. Ứng dụng

Trong phần này, chúng tôi mô tả các ứng dụng của giao thức đối với các vấn đề tối ưu hóa cụ thể, phù hợp với thực tế. Đầu tiên, chúng tôi đề xuất các thuật toán từ tài liệu có thể được sử dụng dưới dạng các bản khởi tạo của thuật toán DPLS chung. Tiếp theo là cuộc thảo luận về một số vấn đề trong thế giới thực có thể được giải

quyết bằng giao thức. Cuối cùng, chúng tôi đánh giá hiệu suất của một ví dụ cụ thể về biến thể DPLS của WalkSAT.

6.1. Các thuật toán phù hợp

Một ví dụ điển hình cho DPLS là thuật toán WalkSAT [55, 56], mà chúng tôi đã mô tả trong Phần 3. Trên thực tế, hầu hết các thuật toán tìm kiếm cục bộ Stochastic (Stochastic LocalSearch - SLS) nổi tiếng [34] có thể được ánh xạ tới DPLS như sau: Hàm Init cung cấp thông tin ban đầu cần thiết, ví dụ: một số vị trí bắt đầu khác nhau để thực hiện song song WalkSAT. Với vị trí hiện tại, M được thiết lập để khám phá một vị trí duy nhất trong vùng lân cận của nó và bất kỳ sự ngẫu nhiên cần thiết nào sẽ được cung cấp bởi hạt giống. Do đó, hàm UPDATE có thể được hiểu là khám phá một hoặc nhiều vị trí trong vùng lân cận tùy thuộc vào các tham số Post-Hash p_2 và p_3, sau đó trả về giá trị tối đa hóa hàm tính điểm g. Vị trí này sau đó có thể đóng vai trò là điểm tiếp theo trong Walk ngẫu nhiên. Lưu ý rằng DPLS cho phép cả hai lần Walk song song, có nghĩa là nhiều lần Walk có thể được thực hiện đồng thời bởi những thợ đào khác nhau phối hợp thông qua Blockchain và cho song song trong mỗi lần Walk, tức là trước khi một bước được thực hiện, nhiều vị trí trong khu vực lân cận được đã khám phá [58].

Như đã lập luận ở trên, DPLS biểu đạt khá tốt. Tuy nhiên, chúng tôi mong đợi hiệu suất tốt hơn khi đáp ứng hai điều kiện sau: (i) số lượng các vị trí lân cận được khám phá trong mỗi bước là lớn, sao cho công việc DPLS dự kiến tạo ra một Block lớn hơn nhiều so với công việc tạo bằng chứng SNARG tương ứng cho lợi ích của tính hữu ích, như được giải thích trong Phần 5.3, và (ii) tổng quy mô vùng lân cận là quá lớn, để các bên không khám phá cùng một vị trí do không đồng bộ hóa. Thực tế là các vị trí được tìm kiếm là xác định một cách ngẫu nhiên.

Hơn nữa, các thuật toán tìm kiếm các vùng lân cận lớn hơn cũng có thể là ứng cử viên tốt hơn liên quan đến độ cứng vừa phải, vì vùng lân cận nhỏ làm cho các cuộc tấn công tính toán trước có khả năng dễ dàng hơn.

Một lớp phụ của các thuật toán SLS có hai đặc điểm này là các thuật toán tìm kiếm Vùng lân cận Quy mô Rất lớn [3], trong đó thuật toán (một phần) tìm kiếm một vùng lân cận rất lớn trước khi thực hiện bước tiếp theo. Ví dụ, trong kinh nghiệm tìm kiếm vùng lân cận lớn [53], việc khám phá một vùng lân cận bao gồm việc phá hủy một phần của giải pháp và sau đó tái tạo lại nó theo cách tham lam, để đạt được mức tối đa mới. Tùy thuộc vào cách tái tạo xảy ra, thời gian cần thiết để có được một giải pháp mới có thể khác nhau, ví dụ: nếu trong quá trình tái tạo, một vấn đề Lập trình tuyến tính được giải quyết, toàn bộ quá trình có thể theo thứ tự vài giây. Một ví dụ khác là phương pháp tìm kiếm theo độ sâu thay đổi [40, 3], trong đó vùng lân cận trao đổi *k* được tìm kiếm một phần trong nỗ lực tìm kiếm các giải pháp gần với mức tối thiểu cục bộ của vùng lân cận. Như trước đây, tham số *k* có thể được sử dụng để điều chỉnh thời gian cần thiết cho một lần tìm kiếm vùng lân cận.

6.2. Các vấn đề trong thế giới thực

Bây giờ chúng tôi trình bày một số vấn đề tối ưu hóa trong thế giới thực phù hợp cho giao thức. Một trong những lợi thế nhỏ của hệ thống là nó có thể cung cấp quyền truy cập vào một lượng lớn tính toán sức mạnh. Mức tiêu thụ điện năng hiện tại của Bitcoin có thể so sánh với mức tiêu thụ điện của một quốc gia quy mô trung bình. Một lợi ích bổ sung là khả năng kiểm toán công khai: kết quả thực thi thuật toán được đồng ý công khai và đi kèm với bằng chứng về tính đúng đắn có thể xác minh công khai.

Các thuộc tính này đặc biệt hấp dẫn đối với các vấn đề quản trị. Ví dụ, vào năm 2016, Ủy ban Truyền thông Liên bang Hoa Kỳ (FCC) đã tổ chức một cuộc đấu giá quang phổ trên đài phát thanh [15] liên quan đến việc giải quyết các trường hợp lớn của vấn đề cải tạo lại nhà ga [22]. Cuộc đấu giá diễn ra theo từng vòng và các nhà tổ chức đặt ra giới hạn là một phút để giải quyết từng trường hợp, vì giải pháp của các trường hợp khác nhau trực tiếp cản trở các giai đoạn tiếp theo của cuộc

đấu giá. Quá trình giải quyết theo sau bởi FCC liên quan đến thuật toán SLS làm thành phần chính của nó. Tốc độ và khả năng kiểm toán công khai sẽ là những lập luận mạnh mẽ ủng hộ việc áp dụng khuôn khổ của chúng tôi trong trường hợp này.

Ví dụ thứ hai là vấn đề sắp xếp thứ tự cư dân / bệnh viện. Ban đầu, cả người nộp đơn và bệnh viện đều khai báo thứ tự mà họ thích hơn. Sau đó, cơ quan liên quan như NRMP ở Hoa Kỳ, cố gắng tạo ra sự phù hợp giữa người dân và bệnh viện để giảm thiểu các vị trí trống trong khi tôn trọng tối đa thứ tự ưu tiên do các bên liên quan đặt ra. Vấn đề là một ứng cử viên sáng giá vì một số thuật toán SLS [45, 54] đã được đề xuất để giải quyết, và cũng do thực tế là có lợi ích công cộng nên việc khớp lệnh thu được một cách công bằng.

Một vấn đề tối ưu hóa khó có liên quan khác là lập lịch sự kiện thể thao. Chẳng hạn, Liên đoàn bóng đá quốc gia Hoa Kỳ sử dụng thuật toán SLS [19] để tạo lịch thi đấu của mỗi mùa giải [47]. Vấn đề tối ưu hóa liên quan được tạo ra bằng cách cân nhắc xem mỗi đội phải di chuyển bao nhiêu, tính sẵn có của sân vận động và sở thích của các kênh phát sóng và những thứ khác. Mức độ khó khăn của vấn đề và thực tế là nhiều bên có lợi ích liên quan tham gia vào quá trình làm cho giao thức của chúng tôi trở nên tốt hơn.

6.3. Ví dụ cụ thể

Trong phần này, chúng tôi đánh giá hiệu suất biến thể DPLS của WalkSAT được mô tả trong Phần 3. Hãy lưu ý rằng mục tiêu ở đây là cung cấp bằng chứng về ý tưởng của chúng tôi, chứ không phải đưa ra giải pháp có thể cạnh tranh với công nghệ tiên tiến nhất.

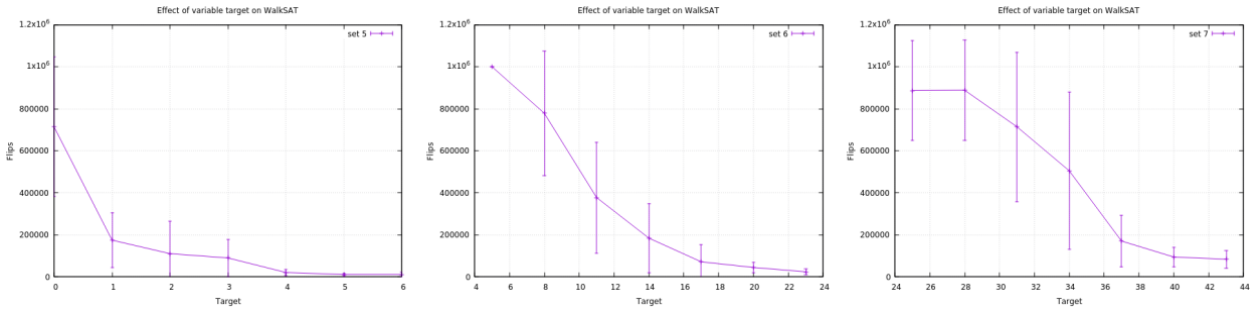
Ở mức cao, mỗi luồng của DPLS chạy quy trình WalkSAT (Phần 3.1, Thuật toán 2 và 3) cho một số lần lật (Flip) giới hạn. Quá trình cho điểm sử dụng bởi hàm UPADTE để chọn ra cái tốt nhất trong số các chủ đề này, xuất ra số lượng các mệnh đề phù hợp trong quy tắc cuối cùng của thuật toán Walk. Ngoài ra, nhiều Walk như vậy được thực hiện đồng thời, với các bên chọn ngẫu nhiên Walk nào để

thực hiện.

Chúng tôi đã thực nghiệm đánh giá hiệu suất của thuật toán so với thuật toán WalkSAT. Bộ thử nghiệm của chúng tôi bao gồm các trường hợp vấn đề về Lập kế hoạch tự động [28], cụ thể là Blocks World Planning; nhiều thử nghiệm hơn trên các tập dữ liệu khác nhau đã được thực hiện cho thấy hành vi tốt tương tự như được chứng minh ở đây. Chi tiết hơn, vấn đề bao gồm việc xác định một kế hoạch để di chuyển các Block trong bảng từ vị trí ban đầu sang vị trí mục tiêu trong một số lần di chuyển nhất định. Trong đó bước di chuyển duy nhất được phép là di chuyển một Block từ đỉnh của một nhóm Block đến đỉnh của một nhóm khác. Để biết thêm thông tin về sự cố và các trường hợp đã chọn (các trường hợp *bw_large.b, bw_large.c, bw_large.d* đã được sử dụng), chúng tôi giới thiệu người đọc đến [1, 33].

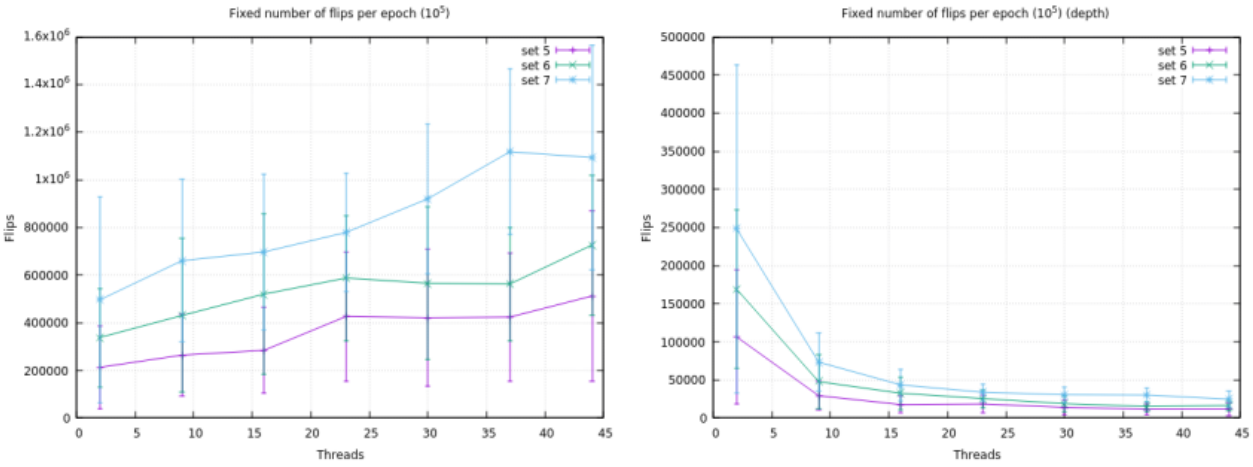
Để có thể so sánh, chúng tôi chọn giải quyết từng phần trong ba trường hợp đã cho bằng cách cho phép các giải pháp để lại nhiều nhất các mệnh đề T không thỏa mãn, đối với một số tham số T. Bằng cách này, chúng tôi có thể có được mức độ cứng tương tự cho ba trường hợp xấp xỉ 200k lên 300k lần lật khi chạy WalkSAT đơn giản. Giới hạn trên 10^6 lần lật được sử dụng trong tất cả các thử nghiệm của chúng tôi. Theo Hình 4, chúng tôi xác định T là 1, 13 và 33, cho các trường hợp khác nhau.

Trước tiên, chúng tôi kiểm tra hành vi của thuật toán, giả sử một điểm bắt đầu duy nhất và tất cả các bản cập nhật đều lần lượt được tạo ra một cách trung thực. Như đã mô tả trước đó, thuật toán của chúng tôi tiến hành trong các Epoch. Trong mỗi Epoch, một số luồng khác nhau chạy WalkSAT cho một số lần lật nhất định.



Hình 4: Độ khó của các trường hợp lập kế hoạch. Trên trục x, chúng tôi đặt mục tiêu *T*. Trên trục y, chúng tôi đếm tổng số lần lật mà WalkSAT phải giải quyết vấn đề. Độ lệch chuẩn của các thử nghiệm được hiển thị dưới dạng thanh dọc.

Ở đây, chúng tôi cố định tổng số lần lật trên mỗi Epoch thành 10^5 và nghiên cứu điều gì sẽ xảy ra khi số luồng (dự kiến) thay đổi. Các đại lượng cần quan tâm là: (i) tổng số lần lật mà thuật toán đã thực hiện trước khi đạt được giải pháp và (ii) độ sâu mà bước đi dài nhất trong tìm kiếm của chúng tôi đã đạt được. Trong Hình 5, chúng ta thấy trong khi việc tăng số lượng các luồng chỉ có tác động tiêu cực đến tổng số lần lật, nó cũng dẫn đến giảm độ sâu, do đó phản ánh một cách tích cực việc tìm ra giải pháp sớm như thế nào. Đối với phần còn lại của các thử nghiệm, chúng tôi đặt số luồng (dự kiến) là 20 và số lần lật trên mỗi luồng là 5000.



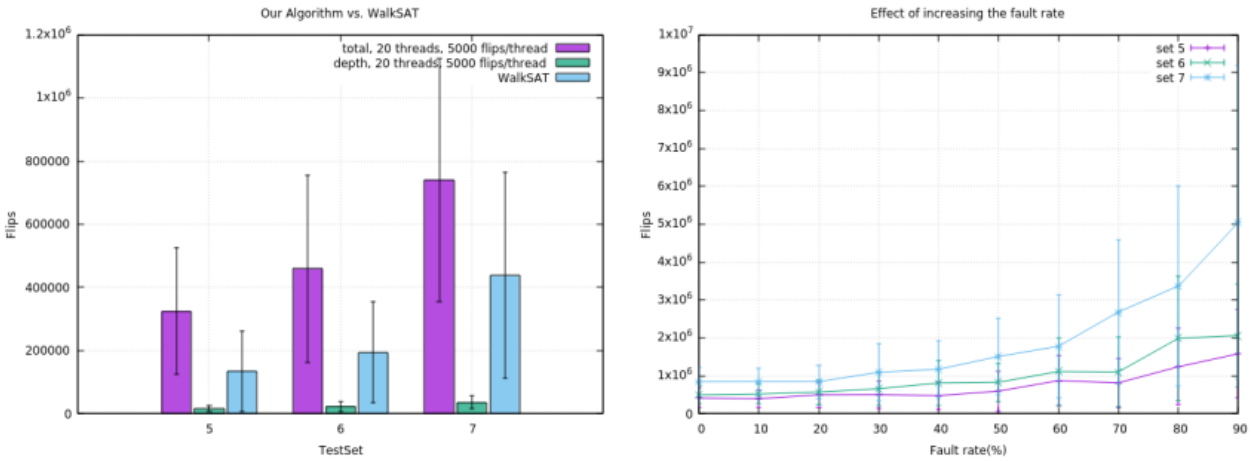
Hình 5: Thay đổi số luồng của DPLS cho một số lần lật cố định trên mỗi Epoch.

Tiếp theo, trong Hình 6, chúng tôi so sánh hiệu suất của thuật toán với WalkSAT và thấy rằng thuật toán của chúng tôi nhanh hơn khoảng một nửa. Lưu ý rằng hệ số này không gần bằng 1/20 (đối với 20 luồng), thể hiện các luồng phụ có ảnh hưởng

đáng chú ý đến tốc độ tìm ra giải pháp.

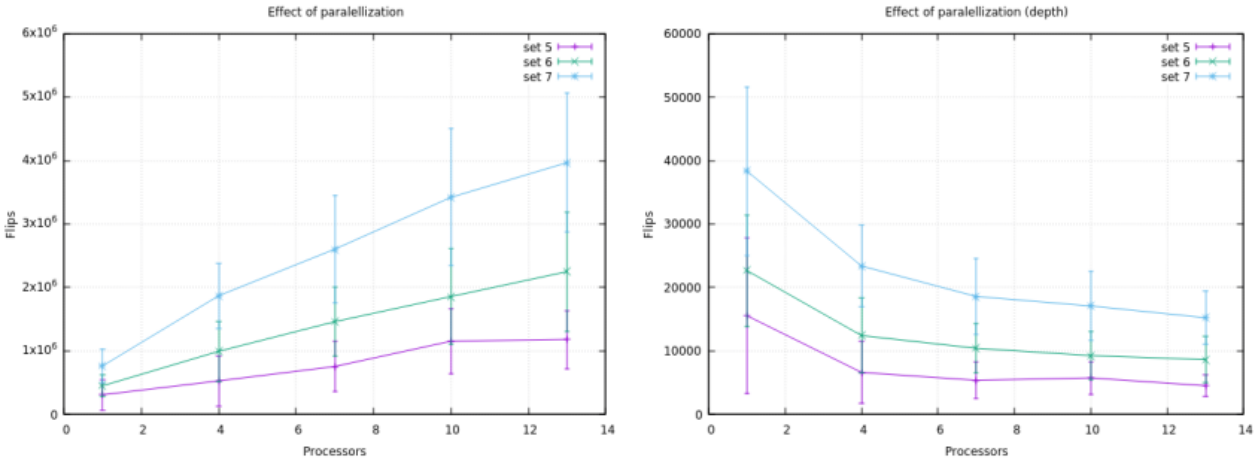
Tập hợp các thử nghiệm tiếp theo liên quan đến rất nhiều lỗi trong thuật toán của chúng tôi. Hành vi bị lỗi mà chúng tôi coi là kẻ tấn công chọn một lần chạy ngẫu nhiên của chức năng *M* thay vì lần chạy tốt nhất khi tạo ra một bản cập nhật mới. Hành vi này khá thực tế, vì có thể dễ dàng phát hiện ra nếu kẻ tấn công công bố một giải pháp tồi tệ hơn nhiều so với một giải pháp được lấy mẫu ngẫu nhiên, vì mỗi Block đều chứa thêm hành động của *M* dẫn đến một Post-Hash nhỏ. Điều chính mà chúng ta quan sát được trong Hình 6 là hiệu suất của thuật toán kém đi nhanh hơn sau khi đánh dấu lỗi 50%, cho thấy một số mức độ chắc chắn đối với các lỗi. Hơn nữa, với lỗi ở mức 50%, hiệu suất của thuật toán sẽ có hệ số nhỏ hơn 2.

Tiếp theo, chúng tôi kiểm tra hiệu ứng của nhiều bản cập nhật được tạo song song trong thuật toán.



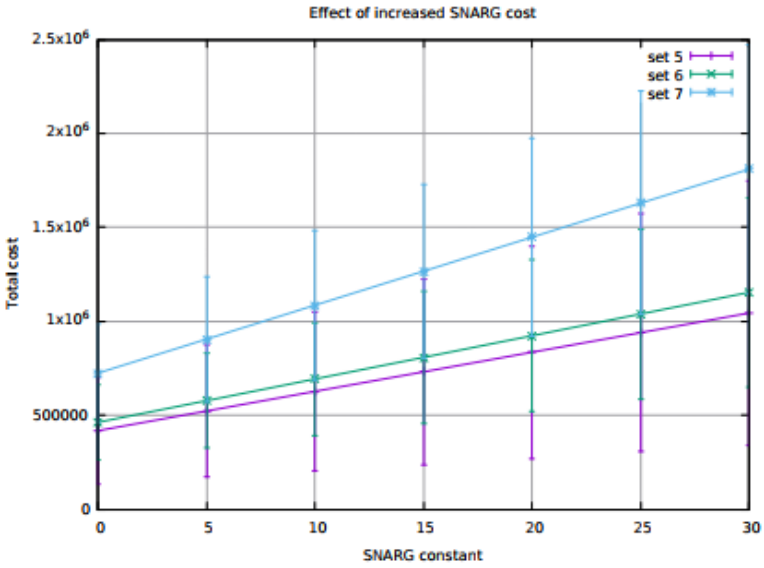
Hình 6: DPLS so với WalkSAT và cách tăng tỷ lệ lỗi trên DPLS.

Hình 7 mô tả hiệu suất của DPLS khi xem xét nhiều điểm xuất phát. Một lần nữa, chúng tôi đang ở trong cài đặt nơi tất cả các bản cập nhật lần lượt được tính toán trung thực. Chúng tôi quan sát thấy trong khi tăng số điểm bắt đầu (trục x) dẫn đến tổng số lần lật tăng lên, sự gia tăng này không tỷ lệ thuận với số điểm bắt đầu; tăng gấp đôi số điểm bắt đầu không tăng gấp đôi số bước. Nó thể hiện quá trình song song hóa tăng tốc độ tìm kiếm của chúng tôi, có nghĩa là độ sâu được giảm xuống.



Hình 7: Thay đổi số lượng điểm bắt đầu của DPLS.

Cuối cùng, trong Hình 8, chúng tôi xem xét hiệu ứng của việc tăng chi phí SNARG trên DPLS. Chúng tôi vẽ biểu đồ tổng chi phí của việc chạy DPLS với một điểm xuất phát duy nhất giả định rằng chi phí SNARG phụ thuộc tuyến tính[6] vào chi phí của một luồng WalkSAT đã được chứng minh là đúng. Hằng số nhân xuất hiện trên trục hoành của đồ thị. Đối với các trường hợp thử nghiệm, tổng chi phí tăng gấp đôi xấp xỉ khi SNARG tốn gấp 20 lần so với tính toán được chứng minh là đúng, tức là gần với số luồng trong quá trình thực thi DPLS. Điều này phản ánh phân tích lý thuyết, chúng tôi đã chỉ ra rằng số lượng luồng càng lớn thì chi phí của SNARG càng ít đáng kể so với tổng chi phí của thuật toán.



Hình 8: Chi phí SNARG tăng trên tổng thời gian chạy của DPLS.

[6]Spartan [57] là một ví dụ về hệ thống SNARG với thuộc tính này.

Sự ghi nhận. Chúng tôi cảm ơn Laurent Michel đã cung cấp thông tin có giá trị về các thuật toán tìm kiếm cục bộ ngẫu nhiên tiên tiến và ứng dụng của chúng vào các vấn đề trong thế giới thực.

Tài liệu tham khảo

[1] Mã hóa vệ tinh ngăn chặn các vấn đề về quy hoạch thế giới. *https://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/PLANNING/BlocksWorld/descr.html*.

[2] A. Aggarwal, A.K. Chandra và M. Snir. Sự phức tạp trong giao tiếp của prams. *Theor. Comput. Sci.*, 71(1): 3-28, 1990.

[3] R. K. Ahuja, Ö. Ergun, J. B. Orlin và A.P. Punnen. Một cuộc khảo sát về các kỹ thuật tìm kiếm khu vực lân cận quy mô rất lớn. *Discrete Applied Mathematics*, 123(1-3): 75 102, 2002.

[4] D. Aldous và J.A. Fill. Chuỗi Markov có thể đảo ngược và bước đi ngẫu nhiên trên đồ thị, 2002. Chuyên khảo chưa phân tích, được biên dịch lại năm 2014, có tại *http://www.stat.berkeley.edu/~aldous/RWG/book.html*.

[5] M. Andrychowicz và S. Dziembowski. Mật mã phân tán dựa trên các bằng chứng công việc. Cryptology ePrint Archive, Report 2014/796, 2014. *http://eprint.iacr.org/*.

[6] C. Badertscher, P. Gazi, A. Kiayias, A. Russell và V. Zikas. Consensus Redux: Sổ cái phân tán trong đối mặt với quyền lực tối cao của đối thủ. *IACR Cryptol. ePrint Arch.*, 2020:1021, 2020.

[7] C. Badertscher, U. Maurer, D. Tschudi và V. Zikas. Bitcoin như một sổ cái giao dịch: Một phương pháp nghiên cứu có thể kết hợp được. Trong J. Katz và H. Shacham, người biên tập, *Những tiến bộ trong Mật mã học CRYPTO 2017*, trang 324-356, Cham, 2017. Nhà xuất bản Quốc tế Springer.

[8] A. Baldominos và Y. Saez. Coin. ai: Một chương trình bằng chứng công

việc hữu ích cho học sâu phân tán dựa trên Blockchain. *Entropy*, 21(8): 723, 2019.

[9] M. Ball, A. Rosen, M. Sabin, và P. N. Vasudevan. Bằng chứng công việc từ các giả định trong trường hợp xấu nhất. Trong *Hội nghị Mật mã Quốc tế Thường niên*, trang 789-819. Springer, 2018.

[10] M. Bellare và P. Rogaway. Một mô hình để thiết kế các giao thức hiệu quả. Trong *CCS '93, Kỷ yếu của Hội nghị ACM lần thứ nhất về Bảo mật Máy tính và Truyền thông, Fairfax, Virginia, Hoa Kỳ, ngày 3-5 tháng 11 năm 1993.*, trang 62-73, 1993.

[11] D. Boneh, J. Bonneau, B. Bünz và B. Fisch. Các chức năng trì hoãn có thể xác minh được. *Những tiến bộ trong mật mã học - CRYPTO 2018.*

[12] K. Chatterjee, A. K. Goharshady và A. Pourdamghani. Khai thác hỗn hợp: khai thác sức mạnh tính toán của Blockchain để giải quyết vấn đề phân tán. Trong *Kỷ yếu của Hội nghị chuyên đề ACM / SIGAPP lần thứ 34 về Máy tính Ứng dụng*, trang 374-381, 2019.

[13] A. Chiesa và E. Tromer. Dữ liệu mang theo bằng chứng và đối số tin đồn từ thẻ chữ ký. Trong *những đổi mới trong Khoa học Máy tính - ICS 2010, Đại học Thanh Hoa, Bắc Kinh, Trung Quốc, ngày 5-7 tháng 1 năm 2010. Kỷ yếu.*

[14] F. Coelho. Một (gần như) giao thức POW xác minh giải pháp nỗ lực liên tục dựa trên Merkle Tree. Cryptology ePrint Archive, Report 2007/433, 2007. *https://eprint.iacr.org/2007/433.*

[15] F.C. Commission. Mở rộng các cơ hội kinh tế và đổi mới của phổ thông qua các cuộc đấu giá khuyến khích.
https:// docs.fcc.gov/public/attachments/FCC-14-50A1.pdf, 2014.

[16] A. Coventry. Nooshare: Một sổ cái phi tập trung của các tài nguyên tính toán được chia sẻ.

https://www.semanticscholar.org/paper/NooShare-%3A-A-decentralized-ledger-of-shared-Coventry/ 4616e9784009f1274a7f4bf6087a6870cd62f122, 2012.

[17] P. Daian, R. Pass, và E. Shi. Snow White: Sự đồng thuận có thể xác nhận lại một cách mạnh mẽ và các ứng dụng để cung cấp bằng chứng cổ phần an toàn tuyệt đối. Trong *Hội nghị Quốc tế về Mật mã Tài chính và Bảo mật Dữ liệu*, trang 23 41. Springer, 2019.

[18] B. David, P. Gazi, A. Kiayias và A. Russell. Ouroboros Praos: Một Blockchain bằng chứng cổ phần bán đồng bộ, an toàn thích ứng. *Những tiến bộ trong mật mã - EUROCRYPT 2018.*

[19] B. N. Dilkina và W. S. Havens. Vấn đề về lịch trình giải bóng đá quốc gia của chúng tôi. Trong *AAAI*, trang 814-819, năm 2004.

[20] M. Dotan và S. Tochner. Bằng chứng về công việc vô ích kết quả tích cực và tiêu cực cho các hệ thống khai thác không có đất. *arXiv preprint arXiv: 2007.01046*, 2020.

[21] S. Dziembowski, S. Faust, V. Kolmogorov và K. Pietrzak. Bằng chứng về không gian. Trong *Hội nghị mã hóa hàng năm*, 2015.

[22] A. Fréchette, N. Newman và K. Leyton-Brown. Giải quyết vấn đề đóng gói lại nhà ga. Trong *Kỷ yếu của Hội nghị AAAI về Trí tuệ nghệ sĩ*, tập 30, 2016.

[23] Gapcoin. Gapcoin, 2014. *https://gapcoin.org/*.

[24] J. A. Garay, A. Kiayias và N. Leonardos. Giao thức xương sống của Bitcoin: Phân tích và ứng dụng. *Những tiến bộ trong mật mã - EUROCRYPT 2015*.

[25] J. A. Garay, A. Kiayias và G. Panagiotakos. Cơ chế đồng thuận từ các chữ ký của công việc. Trong *Chủ đề về Mật mã học CT-RSA* 2020.

[26] J. A. Garay, A. Kiayias và G. Panagiotakos. Các Blockchains từ các hàm

Hash không được lý tưởng hóa. Trong *Lý thuyết Mật mã*, 2020.

[27] P. Gazi, A. Kiayias và A. Russell. Giới hạn nhất quán chặt chẽ cho Bitcoin. Trong J. Ligatti, X. Ou, J. Katz và G. Vigna, người biên tập, *Hội nghị ACM SIGSAC CCS '20: 2020 về Bảo mật Máy tính và Truyền thông, Sự kiện Ảo, Hoa Kỳ, ngày 9 - 13 tháng 11 năm 2020*, trang 819-838. ACM, năm 2020.

[28] M. Ghallab, D. Nau, và P. Traverso. *Lập kế hoạch tự động: lý thuyết và thực hành*. Elsevier, 2004.

[29] Y. Gilad, R. Hemo, S. Micali, G. Vlachos và N. Zeldovich. Algorand: Nhân rộng các thỏa thuận Byzantine cho Crypto. Trong *Kỷ yếu của Hội nghị Chuyên đề lần thứ 26 về Nguyên tắc Hệ điều hành*, trang 51-68, 2017.

[30] J. Groth. Về kích thước của các đối số không tương tác dựa trên ghép nối. *Những tiến bộ trong mật mã học EURO CRYPT 2016*.

[31] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn và I. Miers. Các chuỗi tham chiếu chung có thể cập nhật và phổ biến với các ứng dụng cho ZK-Snarks. Trong *Hội nghị Mật mã Quốc tế Thường niên*.

[32] N. Gupta và D. S. Nau. Về sự phức tạp của quy hoạch thế giới Block. *Artif. Intell.*, 56 (2-3): 223-254, năm 1992.

[33] H. H. Hoos và T. Stützle. Satlib: Một nguồn tài nguyên trực tuyến để nghiên cứu về Sat. *Sat*, 2000: 283-292, 2000.

[34] H. H. Hoos và T. Stützle. *Tìm kiếm cục bộ ngẫu nhiên: Cơ sở và ứng dụng*. Elsevier, 2004.

[35] H. Kautz, B. Selman và D. McAllester. Walksat cuộc thi Sat năm 2004. Trong *Kỷ yếu của Hội nghị Quốc tế về Lý thuyết và Ứng dụng của Kiểm tra Năng lực Đáp ứng*, 2004.

[36] T. Kerber, A. Kiayias và M. Kohlweiss. Khai thác vì quyền riêng tư: Cách khởi động một Blockchain linh hoạt. Cryptology ePrint Archive, Report

2020/401, 2020. *https://eprint.iacr.org/2020/401*.

[37] A. Kiayias, A. Russell, B. David và R. Oliynykov. Ouroboros: Một giao thức Blockchain bằng chứng cổ phần được chứng minh an toàn. Trong *Hội nghị Mật mã Quốc tế Thường niên*, trang 357-388. Springer, 2017.

[38] S. King. Primecoin: Crypto với số nguyên tố bằng chứng công việc, 2013.

[39] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics và M. Harvilla. Một bằng chứng về công việc hữu ích cho AI trên Blockchain. *arXiv preprint arXiv: 2001.09244*, 2020.

[40] S. Lin và B. W. Kernighan. Một thuật toán Heuristic hiệu quả cho bài toán nhân viên bán hàng đi du lịch. *Nghiên cứu hoạt động*, 21(2): 498-516, 1973.

[41] A. F. Loe và E. A. Quaglia. Chinh phục các vị tướng: một np-hard về bằng chứng công việc hữu ích. Trong *Kỷ yếu của Hội thảo đầu tiên về Tiền điện tử và Blockchains dành cho Hệ thống Phân tán*, trang 54-59, 2018.

[42] M. Maller, S. Bowe, M. Kohlweiss và S. Meiklejohn. Sonic: Zero-Knowledge Snarks từ các chuỗi tham chiếu có cấu trúc phổ quát và có thể cập nhật kích thước tuyến tính. Trong *Kỷ yếu của Hội nghị ACM SIGSAC 2019 về Bảo mật Máy tính và Truyền thông*, trang 2111-2128, 2019.

[43] A. Miller, A. Juels, E. Shi, B. Parno, và J. Katz. Permacoin: Thay thế công việc của Bitcoin để bảo toàn dữ liệu. Trong *Hội nghị chuyên đề IEEE về Bảo mật và Quyền riêng tư năm 2014*, trang 475-490. IEEE, 2014.

[44] T. Moran và I. Orlov. Bằng chứng về không gian-thời gian đơn giản và bằng chứng hợp lý về việc lưu trữ. *Những tiến bộ trong lĩnh vực mật mã CRYPTO 2019.*

[45] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, và P. Codognet. Giải quyết các vấn đề khó đối sánh ổn định thông qua tìm kiếm cục bộ và song song hợp tác. Trong *Kỷ yếu của Hội nghị AAAI về Trí tuệ Nghệ sĩ*, tập 29, 2015.

[46] S. Nakamoto. Bitcoin: Một hệ thống Crypto ngang hàng. http://bitcoin.org/bitcoin.pdf, 2008.

[47] NFL. Tạo lịch trình Nfl. https://operations.nfl.com/gameday/nfl-schedule/creating-the-nfl-schedule/.

[48] C. G. Oliver, A. Ricottone và P. Philippopoulos. Đề xuất cho một Blockchain hoàn toàn phi tập trung và thuật toán bằng chứng công việc để giải quyết các vấn đề np-complete. *arXiv preprint Xiv: 1708.09419*, năm 2017.

[49] C. H. Papadimitriou và J. D. Ullman. Đánh đổi thời gian giao tiếp. *SIAM J. Comput.*, 16 (4): 639-646 năm 1987.

[50] S. Park, A. Kwon, G. Fuchsbauer, P. Gazi, J. Alwen, và K. Pietrzak. Spacemint: Một loại Crypto dựa trên các bằng chứng về không gian. Trong *Hội nghị Quốc tế về Mật mã Tài chính và Bảo mật Dữ liệu*, 2018.

[51] R. Pass, L. Seeman, và Abhi Shelat. Phân tích giao thức Blockchain trong mạng không đồng bộ. Cryptology ePrint Archive, Report 2016/454.

[52] R. Pass và E. Shi. Fruitchains: Một Blockchain công bằng. Trong *Kỷ yếu của Hội nghị chuyên đề ACM về các Nguyên tắc của Máy tính Phân tán, PODC 2017, Washington, DC, Hoa Kỳ, ngày 25-27 tháng 7 năm 2017*, trang 315-324, 2017.

[53] D. Pisinger và S. Ropke. Tìm kiếm khu vực lân cận lớn. Trong *Handbook of Metaheuristics*, trang 99-127. Springer, 2019.

[54] M. Sartori. Một thuật toán tìm kiếm cục bộ để đối sánh bệnh viện với cư dân. 2013.

[55] B. Selman, H. A. Kautz và B. Cohen. Các chiến lược tiếng ồn để cải thiện tìm kiếm cục bộ. Trong *Kỷ yếu của Hội nghị Quốc gia lần thứ mười hai về Trí tuệ Nghệ sĩ (Tập 1)*, AAAI '94, trang 337â 343, Hoa Kỳ, 1994.

[56] B. Selman, H. A. Kautz và B. Cohen. Các chiến lược tìm kiếm cục bộ để

kiểm tra khả năng đạt yêu cầu. Trong *DIMACS Loạt bài Toán rời rạc và Khoa học Máy tính Lý thuyết*, trang 521-532, 1996.

[57] S. Setty. Spartan: Các ZK-Snarks đa năng và hiệu quả mà không cần thiết lập đáng tin cậy. Trong *Hội nghị mật mã quốc tế hàng năm*, 2020.

[58] M. G. A Verhoeven và E. H. Aarts. Tìm kiếm cục bộ song song. Tạp chí heuristics, 1 (1): 43 65, 1995.

[59] F. Zhang, I. Eyal, R. Escriva, A. Juels, và R. Van Renesse. {REM}: Khai thác tài nguyên hiệu quả cho các Blockchain. Trong *Hội nghị chuyên đề về bảo mật {USENIX} lần thứ 26 ({USENIX} Security 17)*, trang 1427 1444, năm 2017.

[60] W. Zheng, X. Chen, Z. Zheng, X. Luo, và J. Cui. Axechain: Một Blockchain an toàn và phi tập trung để giải quyết các vấn đề có thể dễ dàng xác minh. *arXiv Preprint arXiv: 2003,13999,* 2020.

A. Một giao thức đầy đủ

Các giao thức của chúng tôi tuân theo cùng một cấu trúc cấp cao (Thuật toán 6) như Bitcoin, được chính thức hóa trong [24]. Trong mỗi vòng, thợ đào tìm nạp bất kỳ chuỗi mới nào được khuếch tán trong mạng lưới và chọn chuỗi hợp lệ dài nhất giữa những chuỗi họ đã nhận được (Thuật toán 7 và 8). Sau đó, dựa trên nội dung của chuỗi đã chọn và các thông báo khác nhận được từ mạng lưới, họ chọn Input Block *m*, cũng như thông tin liên quan đến tính toán DAG mà họ sẽ thực hiện (Λ, *G*, *z*) và chạy chức năng tạo PoUW (Thuật toán 5). Nếu một Block mới được tạo ra, nó sẽ được sử dụng trong mạng lưới.

Thuật toán 6 Chức năng chính của giao thức được tham số hóa bởi *hàm đóng góp đầu vào* I(·) và *hàm đọc chuỗi* R(·).

1: $C := B_{\text{Gen}}$ -Khởi tạo Block Genesis B_{Gen}

2: st := $\in$, *round* := 0

3: **while** TRUE **do**

4: $C\sim \leftarrow$ maxvalid(*C*; mọi chuỗi *C'* được tìm thấy trong RECIEVE())

5: (st, m, Λ, G) ← I(st, C~, *round*, INPUT(), RECIEVE())

6: C_{new} ← PoUW(C~, m, Λ, G)

7: **if** $C \neq C_{new}$ **then**

8: $C \leftarrow C_{new}$

9: DIFFUSE(C)

10: *round* ← *round* + 1

11: **if** INPUT() contains READ **then**

12: **write** R(m_C) to OUTPUT()

Thuật toán 7: Hàm tìm chuỗi "tốt nhất", được tham số hóa bởi hàm max(·). Đầu vào là {C_1, ... , C_k}.

1: **function** maxvalid($\mathcal{C}_1, \ldots, \mathcal{C}_k$)
2: $temp := \varepsilon$
3: **for** $i = 1$ to k **do**
4: **if** validate($\mathcal{C}_i$) **then**
5: $temp := \max(\mathcal{C}, temp)$
6: **return** $temp$

Giao thức được tham số hóa bởi các hàm $V(\cdot)$, $R(\cdot)$, $I(\cdot)$. Vị từ xác thực nội dung $V(<x_1, \ldots, x_m>)$ là đúng nếu đầu vào của nó là sổ cái hợp lệ, tức là nó ở L. Các sổ cái hợp lệ cho giao thức của chúng tôi chứa các giao dịch di chuyển tiền xung quanh, chính xác như trong Bitcoin và các Input Block hợp lệ hoặc không hợp lệ như được mô tả trong Phần 4.1. Với tầm nhìn xa, chúng tôi cho phép sự tồn tại của các Input Block không hợp lệ trong các sổ cái hợp lệ, để có thể tách mã xác minh "tiêu đề" của Ranking Block khỏi nội dung của nó; xác minh tiêu đề bao gồm xác minh Pre-Hash, Post-Hash và SNARG, và không bao gồm xác minh rằng G đã được tính đúng hay chưa. Lưu ý rằng chỉ cần xác minh tiêu đề của các Ranking Block là đủ để đạt được sự đồng thuận, theo phân tích của chúng tôi trong Phần 5.1. Tiếp theo, hàm đọc chuỗi $R(C)$ trả về nội dung của chuỗi nếu chúng tạo thành một sổ cái hợp lệ. Cuối cùng, hàm đóng góp đầu vào

$I(st, C, round,$ INPUT(), RECIVE()) trả về (st, m, Λ, G), trong đó st là trạng thái mới và m là dãy con lớn nhất của các giao dịch và các Input Block hợp lệ trong đầu vào và nhận các băng (Tape) tạo thành một sổ cái hợp lệ, liên quan đến nội dung của chuỗi mà bên đó đã có, cùng với một giao dịch trung lập được tạo ngẫu nhiên để tránh xung đột.

$I(\cdot)$ cũng mã hóa một phần của Logic thuật toán DPLS, vì nó được sử dụng để chọn Λ và G. Chi tiết hơn, Λ được chọn theo quy trình lập lịch trình, như đã thảo luận trong Phần 4.2. G được chọn làm bản ghi của Λ được trích xuất từ C cộng với bất kỳ điểm / Input Block hợp lệ nào mà thợ đào đã nhận được và xác minh. Mặc dù chúng tôi cho phép sổ cái hợp lệ chứa các Input Block không hợp lệ, nhưng hàm I chỉ tính đến các Input Block hợp lệ trong đó các Pre-Hash và Post-Hash đủ nhỏ và phép tính DAG cũng như bản ghi được sử dụng được tính toán chính xác dựa trên con trỏ (Pointer) chuỗi được tham chiếu. Cuối cùng, chúng tôi giả định rằng I chỉ thay đổi Λ và G sau khi thợ đào tính toán thành công một Block mới hoặc một Epoch mới bắt đầu theo Mục 4.2. Mặc dù có thể dễ dàng mở rộng giao thức của chúng tôi để xử lý các bản ghi khác nhau ở mọi vòng, nhưng điều này sẽ dẫn đến hàm UPDATE phức tạp hơn.

Thuật toán 8: Quá trình xác thực được tham số hóa bởi hàm Hash $H(\cdot)$, vị từ xác nhận chuỗi $V(\cdot)$ và thuật toán xác minh **V** của SNARG. Đầu vào là C.

function validate($\mathcal{C}$)

$b \leftarrow \mathrm{V}(\mathbf{m}_{\mathcal{C}}) \wedge (\mathrm{tail}(\mathcal{C}) = B_{\mathtt{Gen}})$ -m_C mô tả nội dung chuỗi C.

if $b = \mathrm{False}$ **then**
  return b
$s' := H(B_{\mathtt{Gen}})$
$\mathcal{C} \leftarrow \mathcal{C}^{1]}$ -Xóa Genesis khỏi C
while $(\mathcal{C} \neq \epsilon \wedge b = \mathrm{True})$ **do**
  $\langle (s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b), \ldots$
    $\ldots (s, m, com, \Lambda, z, r, x'), \pi \rangle \leftarrow \mathrm{tail}(\mathcal{C})$
  $s'' := H(\mathrm{tail}(\mathcal{C}))$
  $h := H(s, m, com, \Lambda, G, z, r)$
  $r' := H(s, m, com, \Lambda, G, z, r, h)$
  $h' := H(s, m, com, \Lambda, G, z, r, h, x')$
  $b_0 := (s = s') \wedge (h < T_1) \wedge (h' < T_2)$
  $b_1 := (H(s_b, m_b, com_b, \Lambda_b, G_b, z_b, r_b, x'_b) = com)$
  $b_2 := \mathrm{SNARG.V}(\Sigma, ((\Lambda, G, z, r', x'), \ldots$
    $\ldots (\Lambda_b, G_b, z_b, r'_b, x'_b)), \pi)$
  if $(\bigwedge_{i \in \{0,\ldots,2\}} b_i)$ **then**
    $s' \leftarrow s''$ -Giữ lại giá trị Hash
    $\mathcal{C} \leftarrow \mathcal{C}^{1]}$ -Bỏ phần đuôi khỏi C
  else
    $b \leftarrow \mathrm{False}$
return b

B. Bảo mật trong điều kiện lý tưởng

Trong phần này, chúng tôi lập luận rằng trong điều kiện lý tưởng, Π có thể dung hoà bất kỳ thiểu số không trung thực nào, tức là, kẻ tấn công có thể gian lận gần 1/2 số bên. Theo điều kiện lý tưởng, chúng tôi muốn nói rằng σ, p_3, c_P, c_V, $\epsilon\hat{}$, p_1/c_H đều gần bằng 0.

Đầu tiên, chúng tôi có được một giới hạn dưới về xác suất đứng yên của q_{rank}. Nói chung, xác suất đứng yên của một trạng thái trong chuỗi Markov (hữu hạn, không thể giải thích được, tích cực lặp lại) là nghịch đảo của thời gian lặp lại trung bình.

Trong trường hợp q_{rank}, điều này có nghĩa là thời gian lặp lại dễ dàng xác định. Gọi R_{rank} biểu thị thời gian lặp lại trung bình cho q_{rank}. Lưu ý rằng kể từ khi chuyển đổi từ q_{pre} sang q_{rank} là chi phí bằng không (chúng tôi chỉ tính chi phí Hash, SNARK và M), R_{rank} bằng với thời gian trung bình để chuyển đổi từ q_{pre} sang q_{rank} (vì chỉ có một chuyển đổi đi từ q_{rank}). Vì vậy, chúng tôi tập trung vào thời gian chuyển tiếp trung bình này mà chúng tôi gọi đơn giản là R. Xét rằng việc trả về q_{pre} có bốn loại, chúng ta có thể mở rộng R như sau:

$$R = (1-p_1)(1+R) + p_1(1-p_2-p_3)[\bar{t}/c_H + 2 + R] \\ + p_1p_2((\bar{t}+c_{\text{P}})/c_H + 2) + p_1p_3((\bar{t}+c_P)/c_H + R + 2)$$

(trong đó $\bar{t}$ là thời gian làm việc hữu ích trung bình và c_P được coi là một hằng số). Như vậy

$$R = \frac{(1-p_1) + p_1[(1-p_2-p_3)[\bar{t}/c_H + 2]}{p_1p_2} \\ + \frac{(p_2+p_3)((\bar{t}+c_{\text{P}})/c_H + 2)]}{p_1p_2}$$

và $R_{\text{rank}} = R$.

Tiếp theo, chúng ta tiến hành chứng minh bổ đề. Ta có $p^*_{\text{rank}} = 1/R_{\text{rank}} = \frac{p2}{1/\text{p1} + t\text{^}/cH + 1 + (p2+p3)cP/cH}$. Đối với σ, np_1/c_H, c_P/c_H, c_V/c_H, €^ << 1, ta có $p^*_{\text{rank}} \approx \frac{p2}{\text{t^}/cH + 1/\text{p1} + 1}$. Tương tự, $\beta c_H \approx \frac{p2}{\text{t^}/cH + 1/\text{p1} + 1}$. Do đó, $p^*_{\text{rank}} \approx \beta c_H$, và do đó $\delta_{\text{MH}} \approx 0$. Bằng cách đặt δ_{tot} gần bằng 0, chúng ta nhận được $\delta_{\text{Steps}}(= \delta_{\text{tot}} + \delta_{\text{MH}})$ có thể gần bằng 0. Ngoài ra, theo giả định của chúng ta là $t' \approx t$, nó thể hiện $n - t \approx t$, hoặc $t \approx n/2$, tức là, bất kỳ đa số không trung thực nào đều có thể được dung thứ.

C. Phân tích chuỗi đặc trưng: Forks, Margin và Common Prefix

Trong phần phụ lục này, chúng tôi khảo sát lý thuyết chung về Forks và Margin được sử dụng để chứng minh Định lý 6 và 7. Một tài khoản đầy đủ có thể được tìm thấy trong [37, 6].

Công cụ ghi sổ kế toán cơ bản của lý thuyết là Δ-fork. Một Δ-fork là khái niệm trừu tượng theo lý thuyết đồ thị duy trì cấu trúc liên kết của Block được xây dựng

để thực thi giao thức đồng thuận kiểu Nakamato. Đặc biệt, nó xác định tiền thân của mỗi Block được tạo, trong mỗi vóng đó mỗi Block được sản xuất, và liệu nhà sản xuất Block là trung thực hay gian lận. Những chi tiết này là đủ để nắm bắt các thuộc tính tồn tại và bền bỉ cơ bản của việc thực hiện. Tham số Δ xác định khoảng thời gian mà tại đó quy tắc chuỗi dài nhất được đảm bảo hoạt động cho những người chơi trung thực: về mặt lịch sử, điều này được xác định bởi độ trễ của mạng lưới, nghĩa là bất kỳ Block nào được tạo trung thực phải có độ sâu vượt quá chiều sâu của bất kỳ Block nào được tạo thành trung thực Δ vòng trước đó. Trong cài đặt của chúng tôi, tham số Δ (được gọi là Γ trong phần nội dung chính) đã thực sự được sử dụng để phản ánh một số sự chậm trễ hơn nữa phát sinh do chuỗi Markov quy định động lực ý nghĩa. Trong mọi trường hợp, vai trò cơ bản mà tham số đảm nhận trong cách tiếp cận là giống hệt nhau.

Định nghĩa 18: (PoW Δ-fork). Gọi Δ là một số nguyên dương và $L \in \mathbb{N}$. Một *PoW Δ-fork* đối với chuỗi $w \in (\mathbb{N}^2)^L$ là một cây gốc được định hướng $F = (V, E)$ với một cặp hàm

$$l_{\#} : V \rightarrow \mathbb{N} \text{ và } l_{type} : V \rightarrow \{h,a\}$$

thỏa mãn các tiên đề dưới đây. Các cạnh được định hướng "cách xa" gốc để có một đường dẫn duy nhất từ gốc đến bất kỳ đỉnh nào. Giá trị $l_{\#}(v)$ được coi là *nhãn* của v. Giá trị $l_{type}(v)$ được coi là *kiểu* của đỉnh: khi $l_{type}(v) = h$, chúng ta nói rằng đỉnh là *trung thực*; nếu không thì đó là *gian lận*.

(i) gốc $r \in V$ là trung thực và có nhãn $l_{\#}(r) = 0$;

(ii) chuỗi các nhãn $l_{\#}()$ dọc theo bất kỳ đường dẫn có hướng nào là không giảm;

(iii) nếu $w_i = (h_i, a_i)$, có chính xác đỉnh trung thực h_i với nhãn i và không nhiều hơn a_i đỉnh gian lận của F với nhãn i;

(iv) cho bất kỳ cặp đỉnh trung thực v, w mà $l_{\#}(v) + \Delta \leq l_{\#}(w)$, $\text{len}(v) < \text{len}(w)$, trong đó len() biểu thị độ sâu của đỉnh.

Một ưu điểm của hình thức này là nó có thể dễ dàng gây ra một lỗi liên tục: hai chuỗi, mỗi chuỗi có độ dài bằng nhau, không đồng ý trong một vòng l cụ thể.

C.1. Ký hiệu Fork, closure

Chúng ta viết $F \vdash_{\Delta} w$ để chỉ ra rằng F là một Δ-fork của chuỗi w. Nếu $F' \vdash_{\Delta} w'$ cho tiền tố w' của w, chúng ta nói rằng F' là một *Fork phụ* của F, được ký hiệu là $F' \underline{\subset} F$, nếu F chứa F' là một đồ thị con có nhãn nhất quán. Một Fork $F \vdash_{\Delta}$ được *đóng lại* nếu tất cả các lá đều trung thực. Theo quy ước thì Fork bình thường, chỉ bao gồm một đỉnh gốc, được đóng lại. *Sự đóng lại (closure)* của một Fork F, được ký hiệu là $F^{\bar{}}$, là Fork phụ đóng tối đa của F.

C.2. Các nhánh (Tines)

Một đường trong Fork F bắt nguồn từ gốc được gọi là một *nhánh* (lưu ý rằng các nhánh không nhất thiết phải kết thúc ở một lá). Đối với đỉnh v trong F, $F(v)$ biểu thị nhánh trong F kết thúc ở v. Với sự tương ứng một-một này giữa các đỉnh và nhánh của một Fork, chúng tôi thường xuyên đặt thêm ký hiệu để nó áp dụng cho cả nhánh và đỉnh. Ví dụ, chúng tôi đặt len(T) biểu thị *độ dài* của nhánh T, bằng số cạnh trên đường đi; hãy nhớ lại rằng len(v) cũng chỉ ra độ sâu của đỉnh v. Để nhấn mạnh Fork mà từ đó v được vẽ, đôi khi chúng tôi viết $\text{len}_F(v)$. Chúng tôi tiếp tục đặt thêm ký hiệu len () để áp dụng cho các Fork: len(F) biểu thị độ dài của nhánh dài nhất trong một Fork F. Một nhánh được gọi là *trung thực* nếu nó kết thúc ở một số đỉnh v với $l_{\text{type}}(v) = h$.

Đối với hai nhánh T, T' của một Fork F, ta viết $T \sim T'$ nếu hai nhánh chia sẻ một đỉnh có nhãn lớn hơn hoặc bằng ℓ. Một cách trực quan, $T \sim_{\ell} T'$ đảm bảo rằng các Blockchain tương ứng đồng ý về trạng thái của sổ cái đến thời điểm ℓ. Nhìn về phía trước, kẻ tấn công chỉ có thể khiến hai bên trung thực không đồng ý về trạng thái của sổ cái tại thời điểm ℓ nếu cô ấy bắt họ nắm giữ hai chuỗi tương ứng với nhánh mà $T \not\sim_{\ell} T'$.

C.3. Chia nhỏ và chi phối Fork

Đối với một chuỗi đặc tính $w = w_1 \ldots w_n \in \Sigma^n$ và một số nguyên dương k, ta gọi $w_{\text{Lk}} = w_1 \ldots w_{n-k+1}$ biểu thị chuỗi có được bằng cách loại bỏ k - 1 ký hiệu cuối cùng. Đối với một Fork $F \vdash_{\Delta} w_1 \ldots w_n$ chúng ta đặt $F_{\text{Lk}} \vdash_{\Delta} w_{\text{Lk}}$ biểu thị Fork thu được bằng cách chỉ giữ lại những đỉnh có nhãn từ tập $\{1,\ldots, n - k + 1\}$. Quan sát

thấy các điểm trung thực xuất hiện trong $F_{L\Delta}$ là những điểm nhất thiết phải hiển thị cho những người chơi trung thực ở một vòng ngay ngoài vòng cuối cùng được mô tả bởi chuỗi đặc trưng. Ta nói rằng nhánh T trong F là Δ-*dominant* nếu len (T) ≥ len($F_{\ell\Delta}$) và đơn giản gọi nó là *có ưu thế* nếu Δ rõ ràng từ ngữ cảnh.

C.4. Lợi thế và lợi nhuận

Chúng tôi phát triển một số công cụ để lý luận về trò chơi dàn xếp. Đối với Δ-fork $F \vdash_\Delta w$, chúng tôi xác định *lợi thế* Δ của nhánh $T \in F$ là $\alpha^\Delta_F(T)$ = len(T) - len($F_{L\Delta}$). Quan sát thấy $\alpha^\Delta_F(T) \geq 0$ khi và chỉ khi T là Δ-dominant trong F. Đối với $\ell \geq 1$, chúng tôi xác định số lượng quan tâm

$$\beta^\Delta_\ell(F) = \max_{\substack{T \not\sim_\ell T^* \\ T^* \text{ is } \Delta\text{-dominant}}} \alpha^\Delta_F(T)\,,$$

Số lượng tối đa này mở rộng trên tất cả các cặp nhánh (T, T^*) trong đó T^* là Δ-dominant và $T \not\sim_\ell T'$. Lưu ý rằng có thể tồn tại nhiều cặp như vậy trong F, nhưng với điều kiện $\ell \geq 1$ sẽ luôn tồn tại ít nhất một cặp như vậy, vì nhánh nhỏ T_0 chỉ chứa đỉnh gốc thỏa mãn $T_0 \not\sim_\ell T$ với bất kỳ T và $\ell \geq 1$, cụ thể là $T_0 \not\sim_\ell T_0$. Vì lý do này, chúng tôi sẽ luôn coi β^Δ_ℓ chỉ cho $\ell \geq 1$. Chúng tôi đặt thêm ký hiệu và gọi

$$\beta^\Delta_\ell(w) = \max_{F \vdash_\Delta w} \beta^\Delta_\ell(F)\,.$$

Một cách trực quan, $\alpha^\Delta_F(T)$ nắm bắt lợi thế (hoặc thiếu hụt) về độ dài của nhánh T so với nhánh trung thực dài nhất được tạo ra ít nhất Δ vòng trước vòng sắp tới và tất cả các bên trung thực đều biết đến. Do đó, $\beta^\Delta_\ell(F)$ ghi lại lợi thế tối đa của bất kỳ nhánh T_a nào trong F có khả năng không đồng ý với một số Δ-dominant nhánh T_h về trạng thái chuỗi lên đến vòng ℓ.

Đặc tính quan trọng thúc đẩy các định nghĩa này là $\beta^\Delta_\ell()$ cung cấp khả năng kiểm soát rõ ràng đối với các sự kiện lỗi liên tục. Điều này được phản ánh trong bổ đề dưới đây.

Bổ đề 19. *Cố định một tham số k và xem xét chuỗi các Fork $F_1 \vdash w_1$, $F_1 \vdash w_1w_2$, . . . giả định với mỗi bước của trò chơi giải quyết cho chuỗi đặc trưng $w = w_1w_2$... Hãy xem xét nhánh T do một bên trung thực nắm giữ ở vòng r_1, đó là Δ-dominant trong*

F_{r1}; giả sử ℓ là một liên kết tròn có đỉnh (Block) B được cố định bởi k đỉnh (Block) trong T. Nếu $\beta_\ell (w_1 ... w_r) < 0$ với mọi $r_1 \leq r \leq r_2$, thì nhánh Δ-dominant bất kỳ chi phối T' của F_{r2} chứa đỉnh B. Đặc biệt, tính bền bỉ được đảm bảo cho Block này.

Chứng minh của bổ đề là một quy nạp đơn giản trên *r*; để biết đầy đủ chi tiết xem [27]. Do đó, để loại trừ các vi phạm liên tục, cần thiết lập $\beta^{\Delta}{}_{\ell}(w) < 0$ cho phù hợp với ℓ và $w_1 ... w_{\ell+t}$ (lưu ý rằng giới hạn này nằm trên $\beta^{\Delta}{}_{\ell}(F)$ đối với bất kỳ Fork có liên quan). Các kết nối tương tự ban đầu được thiết lập cho các Fork PoS [37] và gần đây hơn nữa cho các Fork PoW [27, 6]. Điều này dẫn đến Định lý 6.

Mặt khác, tỷ suất lợi nhuận được đánh giá tốt về mặt phân tích. Trong trường hợp đồng bộ Lockstep, về cơ bản nó tăng đối với mỗi phát hiện PoW gian lận và giảm đối với mọi phát hiện PoW trung thực, với sự hiểu biết rằng nó giảm xuống dưới 0 trước ℓ (do đó thúc đẩy Walk rào cản trong phần nội dung chính). Tình huống với độ trễ Δ phức tạp hơn, nhưng có một thực tế khá đơn giản là tỷ suất lợi nhuận bị giảm đi một cho mỗi khám phá PoW trung thực riêng biệt duy nhất và có thể tăng không quá một cho một phát hiện PoW gian lận; một lần nữa có thêm ràng buộc nó không giảm xuống dưới 0 trước ℓ. Đây là nội dung của Walk "rào cản" và "tự do" trong nội dung chính. Các chứng minh quy nạp đầy đủ xuất hiện trong [6].

Quay trở lại các tuyên bố chính cần thiết cho nội dung của nghiên cứu, để thuận tiện cho người đọc chúng ta nhớ lại cách các thuộc tính chuỗi cơ bản được suy ra từ các giả định của Định lý 7:

- CG trên một khoảng đủ lớn xuất phát từ thực tế là khoảng đó chứa nhiều Block trung thực riêng biệt duy nhất. Quan sát thấy trên (vùng Γ xung quanh) bất kỳ thành công riêng biệt nào, chiều cao của mỗi chuỗi trung thực phải tăng lên.
- ECQ tuân theo một đối số đếm đơn giản miễn là (i.) Biên ở đầu khoảng được giới hạn, để không có chuỗi gian lận có độ dài vượt quá đáng kể so với những người chơi trung thực nắm giữ và (ii.) số lượng Block riêng biệt duy nhất trong khu vực vượt quá số Block gian lận.
- CP tuân theo trực tiếp từ kiểm soát về ký quỹ và quản trị CG.

D. Giao thức khởi động lại trung thực

Chúng tôi đưa ra một bản phác thảo chứng minh rằng việc *khởi động lại* không thực sự thay đổi phân tích bảo mật và như đã chỉ ra trong nghiên cứu, trên thực tế, cải thiện các thuộc tính bảo mật của giao thức. Trong một vòng cụ thể, có ba loại sự kiện mà chúng tôi muốn ghi lại: (i.) Phát hiện ra chiến thắng khai thác trung thực, (ii.) Phát hiện ra chiến thắng khai thác gian lận và (iii.) Giao Block gian lận cho một bên trung thực làm tăng độ dài của chuỗi do bên (trung thực) nắm giữ. Chúng tôi nhận xét rằng các sự kiện cuối cùng này không bị loại bỏ bởi các phương pháp xử lý trước đó nhưng rất cần thiết trong bối cảnh của chúng tôi bởi vì các sự kiện phân phối Block này có thể khởi động lại quá trình khai thác của các bên trung thực (và may mắn thay, lợi nhuận tương đối phù hợp với Block trung thực mới). Để trình bày rõ điều này, chúng tôi sẽ tập trung vào một sự kiện mới: một "nâng cao" (Advance). Đây là sự kiện mà một bên trung thực áp dụng một Blockchain mới thông qua quy tắc chuỗi dài nhất. Lưu ý rằng đây chính xác là những sự kiện khiến các bên trong phân tích hiện tại của chúng tôi "khởi động lại" chuỗi Markov cơ bản. Chúng tôi có thể liên kết một cách tự nhiên một Block cụ thể với một sự kiện Advance; cụ thể là Block sâu nhất trên Blockchain được bên trung thực thông qua trong sự kiện Advance. Xem xét độ trễ của mạng lưới, một Block nhất định trên thực tế có thể chịu trách nhiệm cho nhiều sự kiện Advance, vì nó có thể được phân phối cho các bên trung thực khác nhau trong các vòng khác nhau. Tuy nhiên, hai sự kiện Advance được liên kết với cùng một Block có thể cách nhau không quá Δ vòng (vì những người chơi trung thực được cho là sẽ truyền phát bất kỳ chuỗi mới nào tại thời điểm họ áp dụng chúng và các sự kiện Advance nhất thiết phải tăng độ sâu của chuỗi được nắm giữ bởi bên nâng cao). Trong khi các Block trung thực nhất thiết phải được liên kết với ít nhất một sự kiện Advance phát sinh từ người chơi trung thực đã tạo ra các Block. Block gian lận có thể tạo ra 0 sự kiện Advance ngay cả khi cuối cùng chúng xuất hiện trên một chuỗi được chấp nhận bởi một người chơi trung thực (trong trường hợp đó, chúng không nhất thiết phải là Block cuối cùng trên một chuỗi được phân phối như vậy).

Để phản ánh điều này, chúng tôi làm việc với một khái niệm phong phú hơn về Δ-fork nhằm duy trì đối với mỗi Block, tập hợp các vòng mà Block tạo ra một sự kiện Advance. Các nhánh "chú thích nâng cao" (Advance-annotated) mới này sẽ được liên kết với các chuỗi đặc trưng cũng phản ánh những sự kiện này. Cụ thể, chúng tôi làm việc với các chuỗi đặc trưng "chú thích nâng cao": các chuỗi đặc trưng có cấu trúc $w = w_1, \ldots, w_L$ trong đó mỗi $w_i = (h_i, a_i, d_i, D_i) \in \mathbb{N}^4$: một cách trực quan, hai tọa độ đầu tiên có cùng cách diễn giải với các đối tác của chúng trong các chuỗi đặc tính chuẩn, trong khi hai tọa độ cuối là các sự kiện Advance. Đặc biệt,

- h_i biểu thị số lượng khám phá bằng chứng công việc trung thực,
- a_i biểu thị số lượng khám phá bằng chứng công việc gian lận,
- D_i biểu thị số lượng Block gian lận hoặc trung thực, được liên kết với một sự kiện Advance trong vòng này, có nghĩa là một bên trung thực thông qua Blockchain kết thúc tại Block thông qua quy tắc chuỗi dài nhất; và
- d_i biểu thị số Block gian lận hoặc trung thực, liên quan đến một sự kiện Advnace *lần đầu tiên* trong vòng này.

Chúng ta thấy $hi \leq d_i \leq D_i$.

Định nghĩa 20 (Tăng cường phân phối PoW Δ-fork). Gọi Δ là một số nguyên dương và $L \in \mathbb{N}$. Một sự tăng cường phân phối PoW Δ-fork chuỗi $w \in)\mathbb{N}^4)^L$ là cây có gốc định hướng $F = (V, E)$ với một bộ ba hàm

$$l_{\#} : V \rightarrow \mathbb{N}, \qquad l_A : V \rightarrow 2^L, \qquad \text{và} \quad l_{type} : V \rightarrow \{h,a\}$$

thỏa mãn các tiên đề dưới đây. Các cạnh được định hướng "ra khỏi" gốc để có một đường dẫn có hướng duy nhất từ gốc đến bất kỳ đỉnh nào. Giá trị $l_{type}(v)$ được coi là *kiểu* của đỉnh: khi $l_{type}(v) = h$, chúng ta nói rằng đỉnh *được tạo ra một cách trung thực*; nếu không thì $l_{type}(v) = a$ và nó *được tạo ra theo cách gian lận*. Giá trị $l_{\#}(v)$ được coi là *thời gian tạo* của đỉnh v. Khi $l_A(v) \neq \emptyset$, đỉnh được cho là *nâng cao* và, nếu $s \in l_A(v)$, chúng ta nói rằng đỉnh v có một sự kiện Advance được liên kết với vòng s. Đối với đỉnh nân cao v, số *nâng cao ban đầu* là vòng min $l_A(v)$.

(i) gốc $r \in V$ có kiểu $l_{type}(r) = h$, có thời gian tạo $l_{\#}(r) = 0$ và $l_A(r) = \{0\}$;

(ii) với mọi $v \in V$ và mọi $s \in l_A(v)$, $l_{\#}(v) \leq s$; nếu $l_{type}(v) = h$ thì $l_{\#}(v) \in l_A(v)$;

(iii) chuỗi thời gian tạo $l_{\#}()$ dọc theo bất kỳ đường dẫn được định hướng nào là không giảm;

(iv) nếu $w_i = (h_i, a_i, d_i, D_i)$, thì ta có

(a) chính xác h_i đỉnh trung thực được tạo với thời gian tạo i,

(b) không quá a_i đỉnh gian lận được với thời gian tạo i,

(c) chính xác D_i đỉnh mà $i \in l_A(v)$, và

(d) chính xác d_i đỉnh mà $i = \min l_A(v)$

(v) nếu hai đỉnh u và v thỏa mãn tính chất có một cặp vòng $s \in l_A(v)$ và $t \in l_A(v)$ mà $s + \Delta \leq t$ thì độ sâu của v vượt quá độ sâu của u.

Chúng tôi nhận xét rằng tiên đề cuối cùng phản ánh sự kết hợp của quy tắc chuỗi dài nhất với độ trễ của mạng lưới, có thể được định dạng lại theo cách tự nhiên sau đây. Đối với Fork F, giả sử $D_t(F)$ biểu thị đồ thị con quy nạp được cho bởi hợp của tất cả các đường kết thúc tại các đỉnh v mà $\min l_A(v) \leq t$ - tất cả các đỉnh "được giao cho các bên trung thực" theo thời gian t. Khi đó, nếu tồn tại một phần tử s thuộc $l_A(v)$ mà $s \geq t + \Delta$, thì độ sâu của v vượt quá độ sâu của $D_t(F)$. Về quy tắc chuỗi dài nhất, điều này cho thấy rằng bất kỳ bên nào thực hiện quy tắc chuỗi dài nhất tại thời điểm t đều quan sát tất cả các chuỗi do các bên trung thực nắm giữ tại thời điểm $t - \Delta$.

Như với Δ-fork, chúng tôi sử dụng ký hiệu $F \vdash^{A}_{\Delta} w$ để chỉ ra rằng F là một Δ-fork được chú thích nâng cao cho chuỗi đặc trưng w (được chú thích nâng cao).

Phân tích xoay quanh việc đánh giá lại hành vi của tỷ suất lợi nhuận $\beta_\ell()$ với các ký hiệu mới này. Trước hết, để phản ánh những kiểu mới xuất hiện trong khái niệm Fork ở trên, khái niệm "trung thực" trong khái niệm tiêu chuẩn về Fork phải được thay thế bằng khái niệm "nâng cao". Để được chính xác:

- Khái niệm *đóng* (*Closed*) được thay đổi để chỉ một thời điểm kết thúc bằng một đỉnh nâng cao (chứ không phải là một đỉnh trung thực). Việc

đóng một Fork sau đó chứa tất cả các đỉnh trên chuỗi được những người chơi trung thực áp dụng bằng cách sử dụng quy tắc chuỗi dài nhất. (Như đã đề cập ở trên, với quy ước rằng những người chơi trung thực bỏ qua các Block được giao không nằm trên các chuỗi kích hoạt quy tắc chuỗi dài nhất, thì Fork đóng sẽ trực quan hóa tất cả các Block được giao.)

- Việc chia nhỏ và chi phối Fork (như định nghĩa trong Phần C.3) phải được điều chỉnh thích hợp để xử lý đúng cách $l_A()$. Cụ thể, F_{Lk} (vẫn) được định nghĩa để bao gồm tất cả các đỉnh mà $l_{\#}(v) \leq n - k + 1$. Tất nhiên, định nghĩa của $l_A()$ được cập nhật một cách hợp lý để giải thích cho tập hợp các vòng được chia nhỏ; đó là $l_A(v) \cap \{1, \ldots, n - k + 1\}$. Quan sát thấy việc hạn chế $l_A()$ theo cách này có nghĩa là các đỉnh đang nâng cao trong F có thể không còn nâng cao trong F_{Lk} (vì các vị trí mà chúng sẽ được phân phối không còn tồn tại).

Với những quy ước này, điều quan trọng nhất trong số đó thay thế các đỉnh "trung thực" của lý thuyết thông thường bằng các đỉnh "nâng cao", mối liên hệ trực tiếp giữa $\beta_\ell()$ và các lỗi liên tục (Bổ đề 19) được giữ lại. Đầu tiên chúng ta thảo luận về trường hợp đồng bộ Lockstep (không có độ trễ mạng lưới). Trong trường hợp này, $l_A(v)$ là rỗng hoặc là một đơn vị và $D_i = d_i$. Nhận thấy khi $\beta_\ell(w) \leq 0$, không có Block gian lận nào có độ sâu vượt quá chuỗi trung thực dài nhất, vì vậy không có phân phối Block gian lận nào có thể buộc một bên trung thực từ bỏ chuỗi của họ. Mặt khác, khi $\beta_\ell(w) > 0$, kẻ tấn công có thể thực sự cung cấp các Block có độ sâu vượt quá Block do người chơi trung thực nắm giữ. Tuy nhiên, trong trường hợp này, độ sâu của chuỗi do người chơi trung thực nắm giữ được tăng lên ít nhất một: cụ thể là, điều này làm giảm β_ℓ. Đối với trường hợp có độ trễ của mạng lưới (hoặc độ trễ kỳ lạ hơn như những trường hợp được xem xét trong nghiên cứu này). Hãy lưu ý rằng chiều cao của chuỗi dài nhất mà bất kỳ người chơi trung thực nào quan sát được phải tăng trên bất kỳ khu vực nào có thành công PoW trung thực hoặc phân phối Block gian lận (mà nhất thiết phải tăng chiều cao của chuỗi người nhận) với ít nhất Δ ký hiệu ở hai bên. Điều này có thể làm giảm lợi nhuận trên một khu vực như vậy, như mong muốn. Sau đó phân tích đầy đủ [6].

Người dịch: Nguyễn Văn Tú

Telegram: https://t.me/Tulibra

Link gốc: https://iohk.io/en/research/library/papers/ofelimos-combinatorial-optimization-via-proof-of-useful-work-a-provably-secure-blockchain-protocol/