TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hàm sinh số ngẫu nhiên có thể kiểm chứng và ứng dụng

NGUYỄN ĐĂNG DƯƠNG
duong.nd183902@sis.hust.edu.vn

Ngành Toán Tin
Chuyên sâu Toán ứng dụng

Giảng viên hướng dẫn: PGS. TS. Nguyễn Đình Hân _____

Chữ kí GVHD

Bộ môn: Toán Tin

Viện: Toán ứng dụng và Tin học

HÀ NỘI, 08/2022

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới PGS. TS. Nguyễn Đình Hân, Viện Toán ứng dụng và Tin học, Đại học Bách khoa Hà Nội đã dạy em rất nhiều học phần nền tảng và quan trọng, hướng dẫn em hoàn thành đồ án này và TS. Trần Vĩnh Đức, Trường Công nghệ Thông tin, Đại học Bách khoa Hà Nội đã tận tình giải đáp cho em các kiến thức cơ bản quan trọng, các ý tưởng ứng dụng thực tế trong đồ án này. Bên cạnh đó, em cũng muốn dành lời cảm ơn đến các thầy cô của Viện Toán ứng dụng và Tin học, trường Đại học Bách Khoa Hà Nội, những người đã chỉ dạy và truyền đạt kiến thức cho em trong suốt quá trình học tập.

Tiếp theo, em xin gửi lời cảm ơn đến tất cả thành viên trong gia đình mình, những người luôn quan tâm và tạo động lực để em có thể hoàn thành đồ án.

Cuối cùng, em muốn nói lời cảm ơn đến những người bạn của mình, những người đã góp phần làm cho năm tháng thời sinh viên của em trở lên đầy màu sắc và thú vị.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Cùng với thống kê, lý thuyết xác suất là một nhánh của toán học đã được phát triển để giải quyết với sự không chắc chắn. Lý thuyết toán học cổ điển đã thành công trong việc mô tả thế giới như một chuỗi các sự kiện cố định và thực tế có thể quan sát được, tuy nhiên đến thế kỷ XVII, nó phần lớn không đủ khả năng giải quyết các quá trình hoặc thí nghiệm liên quan đến các kết quả không chắc chắn hoặc ngẫu nhiên. Ban đầu được thúc đẩy bởi mong muốn phân tích các trò chơi cờ bạc của nhà toán học và sau đó là phân tích khoa học về các ca tử vong trong ngành y, lý thuyết xác suất đã được phát triển như một công cụ khoa học giải quyết các mô tả liên quan tới sự ngẫu nhiên.

Với Ethereum blockchain, việc tạo ra các số ngẫu nhiên hay giả ngẫu nhiên gặp nhiều khó khăn hơn máy tính và hệ thống thông thường bởi tính công khai, tất định và phân tán của máy ảo EVM. Hãy tưởng tượng, bạn tham gia một trò chơi có thưởng với mức phí tham gia là 1 ETH. Giải thưởng sẽ là 1000 ETH. Kết quả thắng cuộc được ban tổ chức nói rằng dựa trên một số ngẫu nhiên. Vậy làm sao để chắc rằng con số này không bị thao túng bởi một bên thứ ba hay con số này thực sự là biến ngẫu nhiên có phân phối đều. Hơn nữa nếu bạn là chủ một chương trình game deFi và các vật phẩm trong game đều yêu cầu một số ngẫu nhiên, vậy tốc độ sinh ra số ngẫu nhiên có đủ tốt hay không.

Cách tiếp cận đồ án: sử dụng hàm sinh số ngẫu nhiên có thể kiểm chứng. Trong mật mã học, một hàm ngẫu nhiên có thể kiểm chứng (VRF) là một hàm khóa công khai giả ngẫu nhiên mà cung cấp bằng chứng rằng kết quả được tính toán đúng. Chủ sở hữu khóa bí mật có thể tính toán kết quả trả về của hàm cũng như bằng chứng với bất kỳ đầu vào nào. Mọi người khác có thể kiểm chứng kết quả trả về thực sự được tính toán đúng thông qua bằng chứng và khóa công khai.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ VIẾT TẮT

Thuật ngữ	**Ý nghĩa**
Block	Một lô các transaction cùng với blockhash của block trước đó
Blockchain	Chuỗi khối - Cơ sở dữ liệu công khai, phân tán
Blockhash	Mã băm của block thỏa mãn điều kiện độ khó của thuật toán bằng chứng công việc (PoW)
dApps	Các ứng dụng phi tập trung
EVM	Máy ảo ethereum (Ethereum virtual machine)
Gas	Thước đo về số bước tính toán của EVM
Oracle	Thực thể kết nối blockchain và các hệ thống bên ngoài
Precompile	Đoạn code sẽ được biên dịch tính toán trước và được sử dụng như đầu vào cho chương trình
Transaction	Hành động được thực hiện bởi tài khoản của người dùng không phải của hợp đồng thông minh
VRF	Hàm sinh số ngẫu nhiên có thể kiểm chứng

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Mọi thứ bắt đầu từ năm 2008 với Bitcoin, mọi người có thể gửi tiền cho bất cứ ai trên thế giới không kể vùng lãnh thổ mà không cần người trung gian. Điều này làm Bitcoin khác biệt so với các hệ thống ngân hàng thế giới. Hơn nữa không ai có thể truy cập được vào tài khoản của bạn. Bước đột phá thứ hai là sự xuất hiện của Ethereum vào năm 2015 (phụ lục A), hệ thống máy tính ảo được xây dựng trên sự đột phá của Bitcoin. Từ đó đã dẫn đến sự phát triển của các ứng dụng và dịch vụ trên mạng lưới phi tập trung (dApps). Các ứng dụng này trước hết sẽ có tính phi tập trung do đặc tính trên mạng phi tập trung, không có cá nhân hay tổ chức nào điều khiển. Tính tất định của máy ảo Ethereum. Tính độc lập, nếu ứng dụng phi tập trung có lỗi logic ở các hợp đồng thông minh thì sẽ không ảnh hưởng tới các chức năng chính của Ethereum. Do đó dApps sẽ có một vài ưu điểm sau:

1. Liên tục: Một khi hợp đồng thông minh đã được triển khai trên blockchain thì bất kỳ node nào đều có thể giúp người dùng tương tác với hợp đồng thông minh này.
2. Riêng tư: Ta không cần phải xác thực danh tính để triển khai các hợp đồng thông minh với các ứng dụng phi tập trung.
3. Không kiểm duyệt: Không một thực thể đơn lẻ nào có thể chặn bạn gửi transaction, triển khai hợp đồng thông minh, tương tác với hợp đồng thông minh hay là đọc dữ liệu từ blockchain.
4. Toàn vẹn của dữ liệu: Dữ liệu được đưa lên mạng chuỗi khối là bất biến và không thể chối cãi.
5. Tính toán tin cậy: Hợp đồng thông minh sẽ tính toán đúng theo những gì có thể dự đoán mà không cần tin tưởng vào một bên thứ ba.

Tuy nhiên dApps sẽ có một vài mặt hạn chế về sự khó bảo trì, khó nâng cấp. Bởi vì code đã được công khai lên blockchain sẽ khó chỉnh sửa, kể cả là việc tung ra các bản cập nhật vá lỗi hay các rủi ro bảo mật. Chi phí về hiệu suất cũng là một trong những hạn chế. Do cơ chế của mạng phi tập trung, tất cả các transaction đều được xác thực và lan truyền sang xung quang đối với toàn node. Điều này làm cho các transaction sẽ tiêu tốn tài nguyên toàn mạng gấp nhiều lần dù chỉ phức tạp hơn một chút. Đây thực sự là cản trở cũng như là thử thách đối với việc mở rộng ứng dụng phi tập trung.

1.1 Hàm sinh số ngẫu nhiên có thể kiểm chứng

Từ sự phát triển của dApps, sự cần thiết của số ngẫu nhiên là tất yếu. Số ngẫu nhiên đóng vai trò quan trọng trong lý thuyết trò chơi, với số ngẫu nhiên ta có thể:

1. Tạo ra các hộp quà ngẫu nhiên.
2. Chọn ra người chiến thắng ngẫu nhiên trong một trò chơi xổ số.
3. Tạo ra các tình huống bất ngờ trong trò chơi.
4. Tạo lập và phân phối các sản phẩm NFT.

Tuy nhiên việc tạo ra số ngẫu nhiên trên ethereum không giống như trên máy tính thông thường bởi chính tính minh bạch, phân tán của nó. Nếu ta triển khai bất kỳ thuật toán giả ngẫu nhiên nào trên hợp đồng thông minh thì tại cùng một thời điểm bất cứ ai với bất cứ node nào đều cho ra cùng một kết quả.

Việc giải quyết được bài toán sinh số ngẫu nhiên trên mạng blockchain sẽ tăng tính bảo mật cho các dự án phi tập trung, góp phần làm tăng trải nghiệm người dùng, dẫn ra một hướng đi mới của cơ chế đồng thuận cho mạng blockchain - bằng chứng cổ phần (proof of stake) [1].

1.2 Mục tiêu và phạm vi đề tài

Hiện nay với đa số các ứng dụng phi tập trung nhỏ có sử dụng yếu tố ngẫu nhiên, đội ngũ phát triển sẽ lấy nguồn ngẫu nhiên từ blochhash do cơ sở dữ liệu chuỗi khối này cứ tiếp tục được nối dài và các giá trị blockhash (thêm chi tiết về blockhash phụ lục A) là khó bị thao túng. Tuy nhiên các thợ đào (miner) hoàn toàn có thể bỏ đi lợi ích của việc công khai một block đúng để lấy lợi ích từ việc thao túng nguồn ngẫu nhiên này.

Giải pháp mang tính đột phá hiện nay đó là Chain Link [2]. Dự án cho phép trả về số ngẫu nhiên kèm theo bằng chứng. Dự án được xây dựng từ thuật toán VRF. Tuy nhiên sẽ không có gì đảm bảo cho việc oracle của ChainLink không bị sập.

Cũng dựa trên thuật toán sinh số ngẫu nhiên có thể kiểm chứng, mạng blockchain Algorand đã xây dựng cơ chế đồng thuật bằng chứng cổ phần (PPOS - pure proof of stake) [1]. Cơ chế PPOS giúp cho giao thức đồng thuận tiêu tốn tối thiểu công tính toán ở các node, giải quyết vấn đề về môi trường trong việc vận hành mạng blockchain với bằng chứng công việc (POW- proof of work).

Trong phạm vi đồ án này em sẽ đứng trên vai của ChainLink và đưa ra giải pháp cho định hướng thiết kế mới giúp bất kỳ ai cũng có thể cung cấp dịch vụ sinh số ngẫu nhiên trên blockchain.

1.3 Định hướng giải pháp

Từ định hướng em đã trình bày ở trên em sẽ sử dụng thuật toán sinh số ngẫu nhiên có thể kiểm chứng được trình bày và chứng minh tính đúng đắn tại [3]. Về ý tưởng thuật toán khá tương đồng với ý tưởng chữ ký số. Tức là ai cũng có thể xác thực được chữ ký số hợp lệ nhưng chỉ có một người sở hữu khóa bí mật mới có thể ký được lên transaction hợp lệ.

Với thuật toán này em có thể giải quyết được vấn đề không ai có thể thao túng được nguồn ngẫu nhiên tới hợp đồng thông minh.

Em sẽ thiết kế kiến trúc mô hình gửi và nhận số ngẫu nhiên trên hợp đồng thông minh sử dụng mạng lưới Oracle. Xây dựng mạng oracle đơn giản để tương tác với hợp đồng thông minh.

1.4 Bố cục đồ án

Phần còn lại của đồ án em xin được tổ chức theo bố cục như sau:

Chương 2 trình bày về cơ sở lý thuyết và các kiến thức toán quan trọng, nền tảng em đã học được trên ghế đại học.

Chương 3 em trình bày về thuật toán sinh số ngẫu nhiên có thể kiểm chứng, thuật toán và các tính chất bảo mật của thuật toán.

Chương 4 trình bày về đề xuất mô hình, thử nghiệm và đánh giá kết quả mô hình trong thực tế.

Phần kết luận em kết lại những gì đạt được trong quá trình đồ án và nêu ra định hướng phát triển của đồ án trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này đồ án trình bày về các cơ sở lý thuyết toán quan trọng phục vụ cho việc xây dưng thuật toán ở chương 3. Bao gồm các kiến thức về cấu trúc lý thuyết số nhóm giao hoán, nhóm nhân modulo, trường và trường hữu hạn. Trong cơ sở lý thuyết đồ án trình bày hệ mật đường cong Elliptic - hệ mật được dùng trong thử nghiệm thực tế trong chương 4.

2.1 Lý thuyết số tính toán

2.1.1 Nhóm cyclic

Định nghĩa 2.1.1. *Một nhóm X được gọi là **cyclic** nếu X có hệ sinh là một phần tử: $\{g\} \subset X$. Phần tử g khi đó được gọi là phần tử sinh của X.*

Nhận xét 2.1.1. *Giả sử $(G, \cdot)$ là một nhóm với phần tử đơn vị e và $g \in G$. Ta xét nhóm con cyclic sinh bởi g. Có hai trường hợp xảy ra như sau:*

1. *Nếu $g^\alpha \neq g^\beta$ với mọi $\alpha \neq \beta$ thì nhóm cyclic $\langle g \rangle = \{g^k \mid k \in \mathbb{Z}\}$*
2. *Nếu tồn tại hai số nguyên $\alpha \neq \beta$ sao cho $g^\alpha = g^\beta$ thì vì $g^{\alpha-\beta} = g^{\beta-\alpha} = e$ nên luôn tồn tại số nguyên dương m để $g^m = e$. Gọi n là số nguyên dương nhỏ nhất sao cho $g^n = e$. Khi đó*

$$\langle g \rangle = \{e, g, g^2, ..., g^{n-1}\}$$

có cấp là n và $g^k = e$ khi và chỉ khi k chia hết cho n.

Định nghĩa 2.1.2. ***2.1.2 Nhóm nhân Modulo***

Nhóm nhân modulo p, ký hiệu là $\mathbb{Z}_p^*$. Là một tập gồm $\phi(p)$ phần tử. Với p là số nguyên tố thì $\mathbb{Z}_p^*$ gồm $p-1$ phần tử $1, 2, ..., p-1$ dưới toán tử nhóm nhân modulo p.

Nhận xét 2.1.2. *$\mathbb{Z}_p^*$ thỏa mãn các tính chất của nhóm Abel:*

1. *Hai phần tử bất kì trong tập nhân với nhau modulo p thì cho kết quả thuộc tập hợp.*
2. *Phần tử 1 là phần tử đơn vị, tức là $1 \cdot x = x = x \cdot 1$ với mọi phần tử x trong tập hợp.*
3. *Mọi phần tử x đều tồn tại phần tử nghịch đảo x^{-1} sao cho $xx^{-1} = 1 = x^{-1}x$ (mod p).*
4. *Với mọi phần tử a, b, c trong tập, $a(bc) = (ab)c$ (mod p).*
5. *Với hai phần tử a, b trong tập, $ab = ba$ (mod p).*

Tính chất 4 và 5 là kế thừa từ tính chất của phép nhân trong $\mathbb{Z}$ và cũng như phần tử đơn vị trong tính chất 2. Để chỉ ra tính chất 1 đúng, ta cần chỉ ra tích của hai phần tử trong tập hợp không bao giờ bằng không. Giả sử $xy \equiv 0 \pmod{p}$ với x, y trong tập hợp. Điều này ngụ ý rằng hoặc x hoặc y chia hết cho p. Điều này dẫn tới mâu thuẫn.

Để chứng minh rằng mọi phần tử đều tồn tại phần tử nghịch đảo. Ta để ý rằng $\gcd(x, p) = 1$. Điều này có nghĩa tồn tại hai số nguyên r, s sao cho $xr + ps = 1$. Ta có thể viết lại $xr = 1 - px$ hay $xr \equiv 1 \pmod{p}$. Ta kết luận rằng r (hoặc $r \pmod{p}$) là nghịch đảo của x.

Với phần tử $a \in (G, *)$ xác định a^n bởi $a = a * a * \cdots * a$ [n lần]. Với nhóm $\mathbb{Z}_p^*$ ký hiệu a^n tương ứng với phép toán mũ modulo, ở đó a^n là phần tử thu được từ phần tử a và nhân với bản thân nó n lần $\mod p$. Từ đó ta có thể xác định luật mũ.

1. $a^m b^m = (ab)^m$
2. $a^m a^n = a^{m+n}$
3. $(a^m)^n = a^{mn}$

Để ý rằng m, n là các số nguyên không phải các phần tử của nhóm. Do đó ta có thể thực hiện phép toán cộng và phép toán nhân một cách bình thường. Với số mũ bằng không, $a^0 = 1$.

Định lý 2.1.3. *(Định lý Fermat nhỏ) Với mọi phần tử $a \in \mathbb{Z}_p^*$ ta có $a^{p-1} = 1$.*

Để tìm phần tử sinh của nhóm cyclic $\mathbb{Z}_p^*$ ta có hai cách và cả hai đều có thể triển khai số nguyên tố lớn p.

Cách thứ nhất là chọn một phần tử $a \in \mathbb{Z}_p^*$ và chỉ ra rằng $a^i \neq 1$ với mọi $i = 1, 2, ..., p-2$. Điều này dẫn đến số lượng phép tính lớn khi p lớn.

Cách thứ hai là phân tách $p - 1$ và sử dụng tính chất rằng a là phần tử sinh khi và chỉ khi $a^{(p-1)/q} \not\equiv 1 \pmod{p}$ với mọi số nguyên tố q là ước của $p - 1$. Vì thế ta cần thử ứng viên đối với các ước nguyên tố của $p - 1$. Nhưng tuy vậy bài toán phân tích thừa số nguyên tố cũng là một bài toán khó.

Ví dụ 2.1.4. *Phần tử sinh với $p = 11$*

Với số nguyên tố $p = 11$, ta có $\mathbb{Z}_{11}^* = \{1, 2, ..., 10\}$ với bậc $10 = p - 1$. Xét phần tử sinh $g = 2$ và các phần tử được sinh bởi nó (các tính toán đều được lấy modulo p).

$$\langle 2 \rangle = \{2, 4, 8, 5, 10, 9, 7, 3, 6, 1\} = \mathbb{Z}_{11}^*$$

Với phần tử sinh $g = 3$ chỉ sinh ra nhóm con có bậc bằng 5.

$$\langle 3 \rangle = \{3, 9, 5, 4, 1\}$$

Và phần tử sinh $g = 10$ chỉ sinh ra nhóm con có hai phần tử.

$$\langle 10 \rangle = \{10, 1\}$$

Ta có thể tính toán tất cả các nhóm con được sinh ra từ mỗi phần tử trong $\mathbb{Z}_{11}^*$ như trong bảng 2.1.

g	$\langle g \rangle$	Bậc($\langle g \rangle$)
1	{1}	1
2	{2,4,8,5,10,9,7,6,1}	10
3	{3,9,5,4,1}	5
4	{4,5,9,3,1}	5
5	{5,3,4,9,1}	5
6	{6,3,7,9,10,5,8,4,2,1}	10
7	{7,5,2,3,10,4,6,9,8,1}	10
8	{8,9,6,4,10,3,2,5,7,1}	10
9	{9,4,3,5,1}	5
10	{10,1}	2

Bảng 2.1: Bảng các nhóm con cyclic của $\mathbb{Z}_{11}^*$

Nhận xét 2.1.5. *Với bất kỳ một số nguyên tố p ta đều có*

1. *Nhóm con sinh bởi phần tử 1 luôn luôn có bậc là một.*
2. *Nhóm con sinh bởi phần tử $p-1$ có bậc là 2, do $(p-1)^2 = p^2 - 2p + 1 \equiv 1 \pmod{p}$.*
3. *Có đúng $\phi(p-1)$ phần tử sinh ra toàn bộ nhóm, trong đó $\phi(n)$ là hàm phi Euler, số các số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n.*

Trong ví dụ $p = 11$. Ta có đúng $\phi(11-1)$ phần tử sinh ra toàn bộ nhóm $\mathbb{Z}_{11}^*$. Cụ thể là $\langle 2 \rangle = \langle 6 \rangle = \langle 7 \rangle = \langle 8 \rangle = \mathbb{Z}_{11}^*$.

Hơn nữa, nếu d là ước của $p-1$ thì ta có $\phi(d)$ phần tử sinh ra nhóm con có bậc là d.

2.1.3 Trường và trường hữu hạn

Định nghĩa 2.1.3. *(Trường) Trường F là một tập các phần tử được trang bị hai phép toán $(+), (\cdot)$ với các tính chất sau*

1. *$(F, +)$ tạo thành một nhóm Abel với phần tử trung hòa là 0.*

2. *$(F/\{0\}, \cdot)$ tạo thành một nhóm Abel với phần tử trung hòa là 1.*

3. *Phép nhân trong F phân phối với phép cộng trong F*

$$z(x + y) = zx + zy$$

với mọi $x, y, z \in F$.

Ví dụ 2.1.6.

- Trường số thực $\mathbb{R}$ là một trường có phần tử trung hòa 0 với phép toán cộng và phần tử trung hòa 1 với phép toán nhân. Mọi số thực a đều có phần tử nghịch đảo phép cộng là $-a$, mọi phần tử khác không b có phần tử nghịch đảo phép nhân là $1/b$.
- Tập $\mathbb{Z}_p$ với hai toán tử cộng và nhân modulo p là một trường. Trường này là trường hữu hạn các phần tử.

Số các phần tử của trường được gọi là bậc hoặc lực lượng của trường.

Định lý 2.1.7. *Một trường bậc n chỉ tồn tại nếu n là lũy thừa của một số nguyên tố, i.e $n = p^m$ với số nguyên dương m và số nguyên tố p. Số p được gọi là đặc số của trường.*

Ví dụ 2.1.8.

- Trường hữu hạn có 11 phần tử $GF(11)$.
- Trường hữu hạn có 256 phẩn tử $GF(2^8)$.

Các phần tử của $GF(p^m)$ là các đa thức bậc nhỏ hơn n với hệ số trong $GF(p)$.

$$a_{m-1}x^{m-1} + \ldots a_1x + a_0 = A(x) \in GF(p^m)$$

Trong đó $a_i \in \mathbb{Z}_p$

Phép toán cộng và trừ trên $GF(p^m)$ được thực hiện trên các hệ số của đa thức theo $GF(p)$. Phép toán nhân hai phần tử là phần dư của phép chia Euclidean bởi đa thức bất khả quy P bậc m với tích hai đa thức.

Ví dụ 2.1.9.

Các phần tử của trường $GF(2^3)$ là các đa thức có dạng

$$A(x) = a_2x^2 + a_1x + a_0$$

$GF(8)$ có 8 phần tử

$$GF(2^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$$

$$A(x) = X^2 + x + 1$$

$$B(x) = X^2 + 1$$

$$A(x) + B(x) = x$$

Đa thức bất khả quy bậc 3 $P(x) = x^3 + x + 1$

$$A(x) \cdot B(x) = x^2 + x \mod P(x)$$

Phần tử nghịch đảo $A^{-1}(x)$ của phần tử khác 0 $A(x) \in GF(2^m)$ được xác định bởi

$$A(x) \cdot A^{-1}(x) = 1 \mod P(x)$$

2.2 Đường cong Elliptic

2.2.1 Đường cong Elliptic trên trường số thực

Định nghĩa 2.2.1. Cho $a, b \in \mathbb{R}$ là các hằng số thỏa mãn $4a^3 + 27b^2 \neq 0$. Một đường cong Elliptic không suy biến trên trường số thực là tập E các nghiệm $(x, y) \in \mathbb{R} \times \mathbb{R}$ của phương trình Weistrass

$$y^2 = x^3 + ax + b,$$

cùng với một điểm đặc biệt $\mathcal{O}$ được gọi là điểm ở vô cùng.

Chú ý 2.2.1. *Có thể chứng minh được rằng điều kiện* $4a^3 + 27b^2 \neq 0$ *là cần và đủ để đảm bảo phương trình* $x^3 + ax + b = 0$ *có 3 nghiệm phân biệt (thực và phức). Nếu* $4a^3 + 27b^2 = 0$ *thì đường cong Elliptic được gọi là đường cong Elliptic suy biến. Trong phạm vi đồ án này, chỉ xét các đường cong không suy biến.*

Để các điểm trên đường cong Elliptic tạo thành nhóm Abel, tạo ra một điểm đặc biệt, gọi là điểm ở vô cùng thêm vào đường cong, kí hiệu $\mathcal{O}$. Điểm này không tồn tại trên mặt phẳng xy, nhưng ta giả sử nó nằm ở trên cùng và dưới cùng của trục y. Đặt $P + P' = \mathcal{O}$ thì $\mathcal{O}$ là phần tử đơn vị đối với phép cộng $P + \mathcal{O} = \mathcal{O} + P = P$ với mọi $P, P' \in E$.

Định lý 2.2.2. *(Thuật toán cộng trên đường cong Elliptic) Cho*

$$E\colon y^2 = x^3 + ax + b$$

là đường cong Elliptic và xét các điểm P_1 và P_2 trên E.

- *Nếu $P_1 = \mathcal{O}$ thì $P_1 + P_2 = P_2$.*
- *Trái lại, nếu $P_2 = \mathcal{O}$ thì $P_1 + P_2 = P_1$.*
- *Trái lại, viết $P_1 = (x_1, y_1)$ và $P_2 = (x_2, y_2)$.*
- *Nếu $x_1 = x_2$ và $y_1 = -y_2$, thì $P_1 + P_2 = \mathcal{O}$.*
- *Trái lại, định nghĩa λ bởi:*

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{nếu } P_1 \neq P_2, \\ \dfrac{3x_1^2 + A}{2y_1} & \text{nếu } P_1 = P_2, \end{cases}$$

và đặt

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{và} \quad y_3 = \lambda(x_1 - x_3) - y_1.$$

thì $P_1 + P_2 = (x_3, y_3)$.

Phép nhân 2 điểm trên đường cong E không được định nghĩa, có nghĩa là không tồn tại $P_1 \times P_2$ với $P_1, P_2 \in E$.

Không tồn tại thuật toán chia một điểm cho một số nguyên dương $P_2 : n$. Bài toán tìm số n thỏa mãn $P_2 = nP_1$ là bài toán chưa có thuật toán giải trong thơi gian đa thức.

Định lý 2.2.3. *Cho E là một đường cong Elliptic. Khi đó luật cộng trên E có các tính chất sau:*

1. *Phần tử đơn vị: $P + \mathcal{O} = \mathcal{O} + P = P$ với mọi $P \in E$.*
2. *Phần tử nghịch đảo: $P + (-P) = \mathcal{O}$ với mọi $P \in E$.*
3. *Tính kết hợp: $(P + Q) + R = P + (Q + R)$ với mọi $P, Q, R \in E$.*
4. *Tính giao hoán: $P + Q = Q + P$ với mọi $P, Q \in E$.*

Nói cách khác, luật cộng làm cho các điểm của E tạo thành một nhóm Abel.

2.2.2 Đường cong Elliptic trên trên $\mathbb{Z}_p$

Với $p > 3$ là số nguyên tố. Các đường cong Elliptic trên $\mathbb{Z}_p$ được định nghĩa giống như trên trường số thực, thay các phép toán trên $\mathbb{R}$ bởi các phép toán tương ứng trên $\mathbb{Z}_p$.

Định nghĩa 2.2.2. Cho $p > 3$ là số nguyên tố. Đường cong Elliptic $y^2 = x^3 + ax + b$

trên trường $\mathbb{Z}_p$ là tập nghiệm $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ của phép đồng dư

$$y^2 \equiv x^3 + ax + b \mod p,$$

ở đó $a, b \in \mathbb{Z}_p$ là các hằng số thỏa mãn $4a^3 + 27b^2 \not\equiv 0 \mod p$, cùng với một điểm đặc biệt $\mathcal{O}$ được gọi là điểm ở vô cùng.

Phép toán cộng trên E được định nghĩa như sau: Giả sử $P_1 = (x_1, y_1)$ và $P_2 = (x_2, y_2)$ là các điểm trên E. Nếu $x_2 = x_1$ và $y_2 = -y_1$ thì $P_1 + P_2 = \mathcal{O}$; ngược lại $P_1 + P_2 = (x_3, y_3)$, ở đó

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{và} \quad y_3 = \lambda(x_1 - x_3) - y_1,$$

với

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{nếu } P_1 \neq P_2, \\ \dfrac{3x_1^2 + A}{2y_1} & \text{nếu } P_1 = P_2. \end{cases}$$

Cuối cùng, định nghĩa $P + \mathcal{O} = \mathcal{O} + P = P$, với mọi $P \in E$.

Chú ý 2.2.4. *Luật cộng các điểm trên đường cong Elliptic E trên $\mathbb{Z}_p$ cũng xác định một nhóm Abel $(E, +)$.*

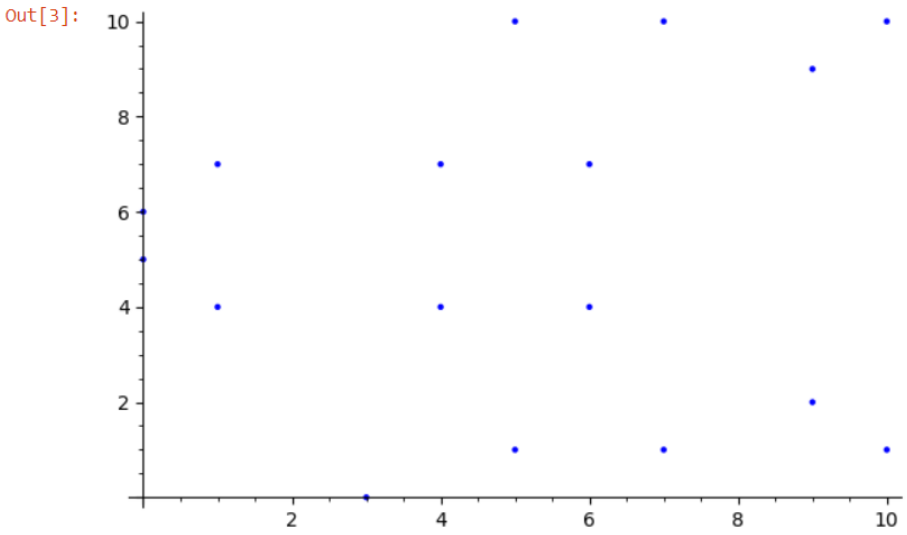
2.2.3 Đường cong Elliptic trên trường hữu hạn

Định nghĩa một đường cong Elliptic trên F_q bởi phương trình có dạng: $E\colon y^2 = x^2 + ax + b$, với $a, b \in F_q$ thỏa mãn $4a^3 + 27b^2 \neq 0$ và xét các điểm trên E với các tọa độ trong F_q, kí hiệu

$$E(F_q) = \big\{(x, y) : x, y \in F_q \text{ thỏa mãn } y^2 = x^2 + ax + b\big\} \cup \{\mathcal{O}\}.$$

Định lý 2.2.5. *Cho E là một đường cong Elliptic trên F_q và lấy P và Q là các điểm trong $E(F_q)$.*

1. *Thuật toán cộng trên đường cong Elliptic áp dụng với hai điểm P và Q tạo ra một điểm trong $E(F_q)$. Kí hiệu điểm này là $P + Q$.*
2. *Luật cộng trên $E(F_q)$ thỏa mãn các tính chất trong nhóm giao hoán. Hình 2.1 biểu diễn các điểm của một đường cong Elliptic trên trường hữu hạn F_{11}.*



Hình 2.1: Đường cong Elliptic trên trường hữu hạn F_{11}

CHƯƠNG 3. HÀM SINH SỐ NGẪU NHIÊN CÓ THỂ KIỂM CHỨNG

Trong phần 3 đồ án trình bày về hàm sinh số ngẫu nhiên có thể kiểm chứng được, đây là thuật toán cốt lõi cho việc triển khai mô hình giải quyết bài toán sinh số ngẫu nhiên trên mạng chuỗi khối có thể kiểm chứng. Đi theo đó. chương trình bày chứng minh tóm tắt các tính chất của VRF.

3.1 Định nghĩa hàm VRF

VRF là bộ ba thuật toán có độ phức tạp thời gian đa thức đó là:

1. Thuật toán sinh khóa G nhận đầu vào là một chuỗi bytes và trả về kết quả hai chuỗi bytes (khóa công khai pk và khóa bí mật sk).
2. $F = (F_1, F_2)$ nhận đầu vào là hai chuỗi bytes (khóa bí mật sk và x) trả về hai chuỗi bytes $v = F_1(sk, x)$ và bằng chứng tương ứng proof $= F_2(sk, x)$.
3. V - hàm xác thực với bốn tham số đầu vào là $pk, x, v,$ proof. Trả về kết quả hợp lệ hoặc không.

3.2 Thuật toán

Một hàm ngẫu nhiên có thể xác thực (VRF) là một hàm khóa công khai giả ngẫu nhiên mà cung cấp bằng chứng rằng kết quả được tính toán đúng. Chủ sở hữu khóa bí mật có thể tính toán kết quả trả về của hàm cũng như bằng chứng với bất kỳ đầu vào nào. Mọi người khác có thể kiểm chứng kết quả trả về thực sự được tính toán đúng thông qua bằng chứng và khóa công khai.

Trong phần này, em sẽ trình bày về bộ ba thuật toán chính của hàm sinh số ngẫu nhiên có thể kiểm chứng được. Chi tiết thuật toán, giao thức, các tính chất được trình bày trong bài báo [3] "Making NSEC5 practical for DNSSEC".

Các tham số công khai: q là số nguyên tố, G là nhóm cyclic cấp q với phần tử sinh g. Bởi vì tính kiểm tra các phần tử thuộc G sẽ tốn nhiều công tính toán, ta giả sử G là một nhóm con của E sao cho việc kiểm tra phần tử trong E là dễ hơn, hệ số $f = |E|/|G|$ không là chia hết q. Dĩ nhiên G có thể trùng với E, trong trường hợp này hệ số $f = 1$.

Ta giả thiết rằng q, g, f, G, E là các tham số công khai.

H_1 là hàm băm (random oracle) ánh xạ một chuỗi có độ dài tùy ý vào $G - \{1\}$. H_2 là một hàm lấy biểu diễn của một phần tử của E và rút gọn nó về độ dài thích hợp. H_3 là một hàm băm (random oracle) ánh xạ một đầu vào với độ dài tùy ý tới số nguyên l-bit.

Thuật toán sinh khóa (Keys) Khóa bí mật $x \in \{1, 2, ..., q-1\}$ được chọn ngẫu nhiên phân phối đều. Khóa công khai của VRF là $PK = g^x$.

Thuật toán sinh bằng chứng (Hashing) Cho khóa bí mật x và đầu vào α, tính toán bằng chứng π theo:

1. Tính phần tử $h = H_1(\alpha)$ và $\gamma = h^x$.
2. Chọn một số ngẫu nhiên dùng một lần $k \in \{1, 2, ..., q - `1\}$.
3. Tính $c = H_3(g, h, g^x, h^x, g^k, h^k)$.
4. $s = k - cx \pmod q$

Bằng chứng π là nhóm các phẩn tử gồm γ, c, s. Kết quả trả về của VRF $\beta = F_{SK}(\alpha)$. Do đó

$$\pi = (\gamma, c, s)$$

$$\beta = H_2(\gamma)$$

Thuật toán kiểm chứng (Verifying) Cho khóa công khai PK, xác thực bằng chứng $\pi = (\gamma, c, s)$ tương ứng với đầu vào α kết quả β

1. Tính $u = (PK)^c \cdot g^s$
   Nếu mọi thứ đúng thì $u = g^k$.
2. Cho đầu vào α, ta thu được $h = H_1(\alpha)$. Kiểm tra $\gamma \in G$. Tính $v = (\gamma)^c \cdot h^s$
   Nếu mọi thứ đúng thì $v = h^k$.
3. Kiểm tra $c = H_3(g, h, PK, \gamma, u, v)$

Cuối cùng, tính $\beta = H_2(\gamma)$.

3.3 Tính bảo mật của VRF

VRF có ba tính chất: tính duy nhất (uniqueness), tính giả ngẫu nhiên (Pseudorandomness), tính kháng xung đột (collistion - resistance). Để đơn giản ta giả sử $E = G$ do đó $f = 1$.

Tính duy nhất (Uniqueness) ([2]) Ta sẽ chứng minh phản chứng. Giả sử một ai đó biết được khóa bí mật x, cùng với đầu vào α, họ đưa ra kết quả gian lận $\beta_1 \neq H_2([H_1(\alpha)]^x)$. Một bằng chứng hợp lệ $\pi_1 = (\gamma_1, c_1, s_1)$ cho kết quả β_1. Các hàm xác thực sẽ tính toán $h = H_1(\alpha)$ và

$$u = (g^x)^{c_1} g^{s_1}$$

$$v = (\gamma_1)^{c_1} h^{s_1}$$

Lấy logarithm của phương trình thứ nhất cơ số g và phương trình thứ hai cơ số h. Trừ hai kết quả ta được biểu diễn của c_1

$$c_1 \equiv \frac{\log_g(u) - \log_h(v)}{x - \log_h(\gamma_1)} \mod q$$

Do $\gamma_1 \neq h^x$ (do β_1 không phải là kết quả đúng), do đó mẫu số khác không, có đúng một số c_1 modulo q sao cho thỏa mãn với $(g, h, g^x, \gamma, u, v)$ và s bất kỳ. Do H_3 là một hàm ngẫu nhiên, kết quả trả ra cũng là ngẫu nhiên. Vậy xác suất để nó bằng với một số được xác định trước trong đẳng thức trên là không đáng kể.

Tính giả ngẫu nhiên (Pseudorandomness) ([2]) Ta đã biết khóa bí mật VRF x sẽ rất khó bị tìm ra bởi bất cứ ai. Các tin tặc sẽ biết khóa công khai, g, g^x và dễ dàng tính được $h = H_1(\alpha)$ với mọi đầu vào α. Theo giả định Diffie – Hellman (DDH) quyết định h^x trông sẽ như là ngẫu nhiên khi biết (g, g^x, h). Do đó $H_2(h^x)$ là giả ngẫu nhiên trong phạm vi của H_2.

Tính kháng xung đột (Collision resistance) ([2]) Sự xung đột xảy ra khi $H_2(h_1^x) = H_2(h_2^x)$ trong đó $h_1 = H_1(\alpha_1)$ và $h_2 = H_1(\alpha_2)$ với $\alpha_1 \neq \alpha_2$. Giả sử rằng H_2 là một hàm τ-to-1. Tức là với mỗi giá trị h_1 sẽ có tối đa τ giá trị h_2 có thể gây ra xung đột. Từ việc h_1 và h_2 thu được từ giả thiết ngẫu nhiên mạnh (random oracle queries). Một cặp như vậy sẽ khó gây ra xung đột sau Q_H yêu cầu của H_1 miễn là G lớn hơn $\tau Q_h^2/2$.

3.4 VRF với đường cong elliptic Secp256k1

Trong đồ án này em sử dụng hệ mật đường cong elliptic (EC) thay vì hệ mật RSA bởi vì khi triển khai trên EC em sẽ thu được bằng chứng VRF có kích cỡ nhỏ hơn trong khi đạt được cùng mức bảo mật so với bằng chứng VRF được triển khai trên RSA (chi tiết [3]).

Trong đồ án này em sử dụng đường cong eliptic secp256k1 bởi vì hai nguyên nhân.

1. Đường cong secp256k1 là một nhóm cyclic. B.2
2. Đây là đường cong được dùng cho chữ ký số trên ethereum.

Với việc tận dụng được các hàm chữ ký số trên ethereum sẽ giảm được chi phí gas khi triển khai cũng như giảm thời gian xác thực bằng chứng ngẫu nhiên trên mạng blockchain.

Các tham số chi tiết về đường cong em xin được trình bày trong phần phụ lục B.2.

3.5 Các hàm băm

Trong thuật toán trong phần trước ta có sử dụng một hàm băm H_1 ánh xạ một chuỗi có độ dài tùy ý tới một điểm trong đường cong eliptic. Có một kỹ thuật mang ý tưởng thử và loại được nhắc đến trong [4]. Giả sử ta có phương trình đường cong $y^2 = x^3 + ax + b$ có bậc qf. Với một input là α. Ta sẽ ghép đầu vào với một biến đếm i, sau đó cho qua hàm băm SHA256 được gái trị lưu vào h. Bước tiếp theo là kiểm tra $h^3 + ax + b$ có là phần dư bậc hai hay không. Nếu thỏa mãn thì h là một tọa độ x hợp lệ trên đường cong. Nếu không, ta tăng biến đếm i lên một đơn vị và thử lại cho đến khi thu được kết quả. Sau cùng ta tăng kết quả mũ f để được phần tử trong G. Quá trình này có kỳ vọng với hai lần thử ta sẽ thu được kết quả thông qua bổ đề B.4.3.

Trong triển khai thực tế em vẫn sử dụng ý tưởng thử và loại tuy nhiên có một chút khác biệt thay vì em ghép input với một biến chạy đến khi thu được kết quả thì em băm lại kết quả vừa loại để được một điểm thử mới.

Bắt đầu bằng thuật toán băm từ một bytes tới một điểm trên trên trường hữu hạn F_p.

Thuật toán fieldHash Với đầu vào là bytes b.

1. x = Hash(b).
2. Kiểm tra $x < p$, nếu không $x =$ Hash(x). Quay lại bước 2.
3. Trả về x.

Tiếp theo là thuật toán tính một điểm ứng cử viên thuộc đường cong Secp256k1

Thuật toán newCandidateSecp256k1Point Với đầu vào là bytes b

1. Tính $px =$ fieldHash(b)
2. Tính ySquare $= px^3 + 7 \mod p$
3. Tính $py =$ ySquare$^{\text{SQRT_POWER}}$

   Với SQRT_POWER $= \frac{(p+1)}{4}$
4. Kiểm tra nếu $py\%2 = 1$ thì $py = p - py$.
5. Trả về $[px, py]$

Khi ấy ta chỉ cần kiểm tra điểm ứng cử viên, nếu thỏa mãn phương trình đường cong Elliptic Secp256k1. Kết thúc thuật toán, nếu chưa thỏa mãn tiếp tục lấy kết quả đầu ra trước làm đầu vào cho thuật toán điểm ứng cử viên và kiểm tra điều kiện.

Thuật toán hashToCurve Với đầu vào là một bytes b

1. $rv =$ newCandidateSecp256k1Point(b)
2. Kiểm tra rv là điểm trên đường cong. Nếu không $b =$ bytes$([rv[0])$ quay lại bước 1.

CHƯƠNG 4. ĐỀ XUẤT, THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương trình bày về mục đích và các giải pháp tồn tại cho hệ thống, phân tích ưu nhược điểm các phương pháp từ đó đề xuất mô hình mới giải quyết hạn chế phương pháp trước đó; thiết kế, thử nghiệm mô hình và đánh giá kết quả.

4.1 Đề xuất

4.1.1 Mục đích hệ thống

Hệ thống được xây dựng với mục đích:

1. Cung cấp dịch vụ sinh số ngẫu nhiên có thể kiểm chứng được cho khách hàng đảm bảo an toàn, không ai có thể thao túng.
2. Hệ thống cung cấp dịch vụ liên tục.
3. Thu phí sử dụng dịch vụ sinh số ngẫu nhiên và trả phần thưởng cho các bên tham gia cung cấp.

Ngoài ra hệ thống được thiết kế phải đảm bảo thời gian phản hồi không quá lâu. Thử nghiệm thành công trên mạng thử nghiệm Ethereum.

4.1.2 Các giải pháp đã tồn tại

Trong phần này em xin trình bày về các giải pháp hiện nay đang áp dụng, từ đó phân tích ưu và nhược điểm. Đa số các dApps nhỏ đều sử dụng blockhash như là nguồn ngẫu nhiên cho dịch vụ của mình. Với cơ chế đồng thuận bằng chứng công việc hiện nay, blockhash mang tính không dự đoán trước. Tuy nhiên blockhash bị các thợ đào (miner) kiểm soát. Các thợ đào có thể đảo thứ tự các transaction rồi tìm mã băm cho tới khi được kết quả như ý mới công bố lên mạng blockchain.

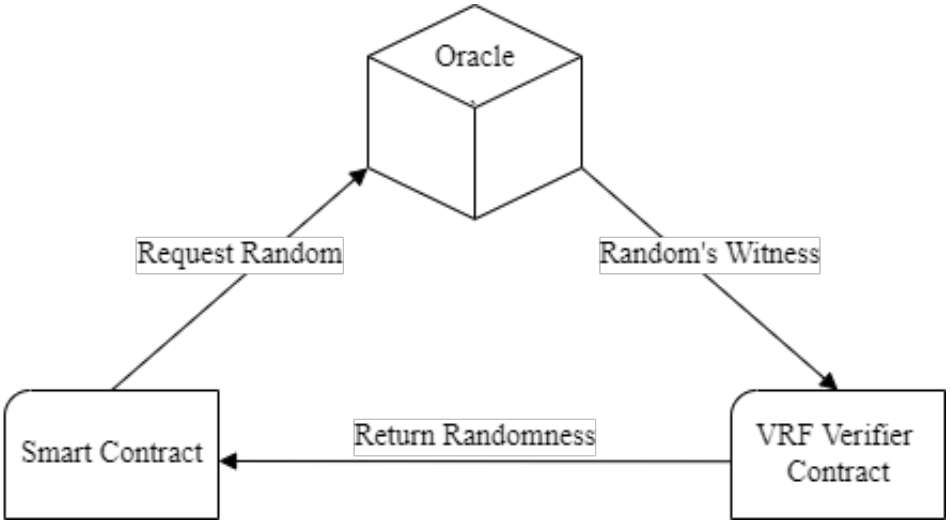
Với việc tiềm ẩn rủi ro về hệ thống nhưng các ứng dụng phi tập trung này vẫn sử dụng blockhash như là nguồn cấp ngẫu nhiên là do:

1. Chi phí lấy nguồn ngẫu nhiên rẻ.
2. Độ khó khi thợ đào được một block mới tăng cao trong bối cảnh thị trường tiền ảo tăng cao, thu hút nhiều thợ đào. Khiến thợ giảm khả năng đào được block mới và tăng tiền thưởng khuyến khích trung thực.

Do vậy ...

Chain Link là gã khổng lồ tiên phong trong việc cung cấp nguồn ngẫu nhiên có thể kiểm chứng có trả phí.

Mô hình hoạt động của ChainLink được mô tả tổng quát trong hình 4.1. Mô hình hoạt động theo luồng như sau. Mỗi khi hợp đồng thông minh yêu cầu một số ngẫu



Hình 4.1: Cách hoạt động của ChainLink

nhiên thì sẽ phát ra một nhật ký (logs) trên blockchain. Các oracle sẽ lắng nghe các sự kiện này và sử dụng thuật toán sinh số ngẫu nhiên có thể kiểm chứng để trả về kết quả cùng bằng chứng tới một hợp đồng thông minh khác gọi là hợp đồng xác thực (VRF Verifier Contract). Nếu hợp đồng xác thực bằng chứng cùng kết quả là khớp sẽ trả số ngẫu nhiên về cho hợp đồng yêu cầu ban đầu.

Cho đến hiện tại Chainlink đã phát hành hai phiên bản dịch vụ sinh số ngẫu nhiên có thể kiểm chứng trên mạng blockchain. Với phiên bản thứ nhất, Chainlink yêu cầu người dùng cung cấp cho Oracle một số ngẫu nhiên (seed) để làm dữ liệu đầu vào. Sau đó thực hiện theo luồng được mô tả trên hình 4.1. Cách thiết kế này sẽ đảm bảo được:

1. Người dùng không thể biết trước được kết quả ngẫu nhiên.
2. Oracle không thể biết trước kết quả cho đến khi người dùng cung cấp seed.
3. Oracle phải đưa ra bằng chứng hợp lệ không gian lận.

Tuy nhiên trong phiên bản thứ nhất, đứng trên trải nghiệm người dùng, phải cung cấp cho Oracle một số để làm yếu tố đầu vào sẽ gây phức tạp tới quy trình sử dụng dịch vụ.

Trong phiên bản thứ hai của mình, ChainLink đã cải tiến giúp người dùng không cần phải gửi kèm một số (seed) khi yêu cầu số ngẫu nhiên mà vẫn giữ được các ưu điểm của hàm sinh số ngẫu nhiên có kiểm chứng. Bởi vì yếu tố đầu vào sẽ được sinh ra từ địa chỉ khách hàng và blockhash của block chứa transaction yêu cầu. Điều này làm cho phía người dùng thuận tiện đơn giản hơn khi sử dụng dịch vụ. Tuy nhiên việc này lại dẫn ra một vài điểm yếu.

1. Về lý thuyết, nếu ChainLink thao túng được cả blockhash thì ChainLink hoàn

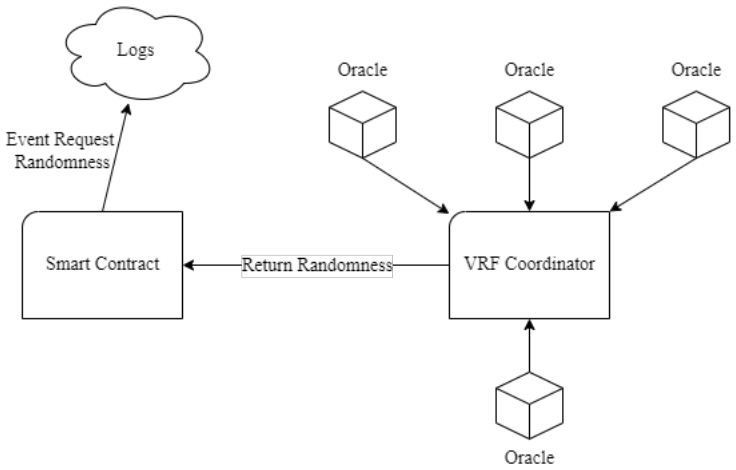
toàn có thể thao túng được cả kết quả ngẫu nhiên.

2. Trong luồng hoạt động của mô hình này, chỉ có một Oracle đảm nhận việc sinh số ngẫu nhiên và sẽ không có gì đảm bảo việc Oracle không hoạt động trong khoảng thời gian, làm gián đoạn dịch vụ.

Hơn nữa trong mạng Ethereum các node chỉ lưu lại blockhash của 256 block gần nhất. Chỉ cần Oracle được chỉ định sinh số ngẫu nhiên bị gián đoạn trong khoảng thời gian này thì yêu cầu khách hàng sẽ không được đáp ứng.

4.1.3 Đề xuất mô hình

Từ những hạn chế của các giải pháp đang tồn tại giải quyết bài toán sinh số ngẫu nhiên trên mạng chuỗi khối. Em đề xuất mô hình như hình 4.2



Hình 4.2: Luồng mô hình sinh số ngẫu nhiên có thể kiểm chứng trên blockchain

Với mỗi hợp đồng thông minh cần sử dụng số ngẫu nhiên sẽ gửi yêu cầu sinh số ngẫu nhiên. Yêu cầu này được ghi lại trên nhật ký (logs) của transaction. Mỗi Oracle nghe được yêu cầu sẽ triển khai tính toán số ngẫu nhiên kèm theo bằng chứng và trả kết quả về hợp đồng kiểm chứng VRF. Hợp đồng kiểm chứng sẽ tính toán xem liệu bằng chứng có hợp lệ hay không. Hợp đồng kiểm chứng sẽ tổng hợp các kết quả hợp lệ và gửi kết quả cuối dùng về tới cho hợp đồng khách ban đầu.

Với ý tưởng ai cũng có thể làm Oracle chỉ cần đăng ký với hợp đồng xác thực. Nếu các Oracle trung thực sẽ nhận được các phần thưởng khuyến khích. Và mô hình sẽ khắc phục được các hạn chế của phiên bản thứ hai Chainlink mà vẫn giữ được ưu điểm của hàm sinh số ngẫu nhiên có thể kiểm chứng.

1. Người có thể biết trước kết quả sinh số ngẫu nhiên thì phải sở hữu toàn bộ các Oracle, thao túng được blockhash. Rủi ro bảo mật sẽ giảm đi rất nhiều nữa nếu các bên Oracle là độc lập và tăng số lượng.

2. Có nhiều hơn một Oracle đảm nhận việc sinh số ngẫu nhiên, do đó nếu vẫn còn đủ Oracle đạt ngưỡng đặt trước thì dịch vụ vẫn đảm bảo.

4.2 Thiết kế kiến trúc và thử nghiệm

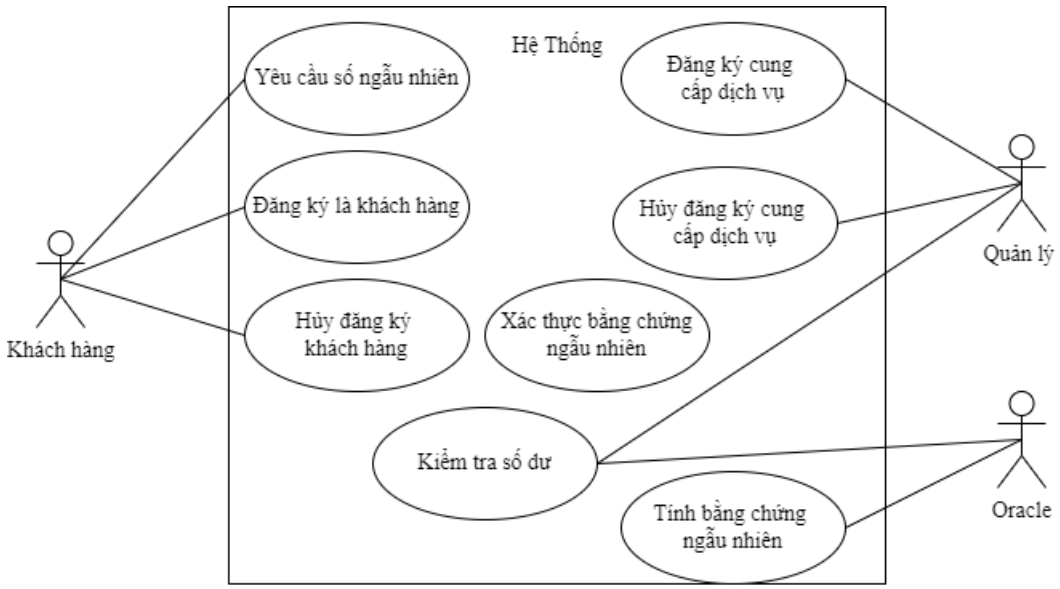
4.2.1 Tác nhân

Trong hệ thống này sẽ có các tác nhân như:

1. Khách hàng - người sử dụng dịch vụ nhận số ngẫu nhiên có thể kiểm chứng.
2. Người sở hữu Oracle - người đăng ký với quản lý để được cung cấp dịch vụ sinh số ngẫu nhiên.
3. Quản lý - chủ sở hữu của dịch vụ, nhận yêu cầu từ khách hàng và phân phối tới nhà cung cấp là các Oracle.

4.2.2 Phân tích yêu cầu chức năng và phi chức năng

Hình 4.3 thể hiện sơ đồ trường hợp sử dụng của hệ thống (use case) của ba tác nhân: khách hàng, người sở hữu Oracle và người quản lý hợp đồng điều phối với các chức năng cơ bản.



Hình 4.3: Sơ đồ trường hợp sử dụng của hệ thống

Chức năng yêu cầu số nhẫu nhiên:	
Mô tả:	Hợp đồng thông minh của khách hàng muốn gửi yêu cầu nhận số ngẫu nhiên.
Tác nhân:	Khách hàng
Luồng cơ bản:	1. Hợp đồng khách yêu cầu số ngẫu nhiên. 2. Một sự kiện yêu cầu số ngẫu nhiên được phát lên nhật ký của Ethereum. 3. Các Oracle nghe được sự kiện, sau đó tính toán bằng chứng nhẫu nhiên gửi tới hợp đồng xác thực. 4. Hợp đồng xác thực các bằng chứng hợp lệ, tổng hợp rồi gửi về cho hợp đồng khách hàng. 5. Hợp đồng khách hàng nhận kết quả ngẫu nhiên.
Luồng ngoại lệ:	1. Người dùng chưa đăng ký sử dụng dịch vụ sinh số ngẫu nhiên. 2. Oracle chưa đăng ký làm bên cung cấp dịch vụ.
Yêu cầu phi chức năng:	1. Thời gian trả về kết quả ngẫu nhiên không quá 10 block. 2. Kết quả ngẫu nhiên phải được xác thực bằng chứng. 3. Đảm bảo hợp đồng điều phối và các Oracle khó có cơ hội gian lận.

Bảng 4.1: Đặc tả chức năng yêu cầu số ngẫu nhiên

Bảng 4.1 đặc tả yêu cầu chức năng sinh số ngẫu nhiên. Khách hàng phải đăng ký cho hợp đồng của mình được sử dụng dịch vụ thông qua hợp đồng điều phối (VRFCoordinator). Sau đó, transaction yêu cầu số ngẫu nhiên được công bố lên mạng chuỗi khối, các Oracle lập tức tính toán bằng chứng và gửi về cho hợp đồng điều phối. Hợp đồng điều phối xác thực bằng chứng ngẫu nhiên và gửi lại trả về cho hợp đồng khách.

Chức năng đăng ký là khách hàng:	
Mô tả:	Hợp đồng thông minh của khách hàng muốn đăng ký sử dụng dịch vụ sinh số ngẫu nhiên.
Tác nhân:	Khách hàng
Luồng cơ bản:	1. Tài khoản khách hàng đăng ký cho hợp đồng sử dụng dịch vụ sinh số ngẫu nhiên. 2. Hợp đồng điều phối (VRFCoordinator) sẽ kiểm tra địa chỉ hợp đồng khách chưa tồn tại. 3. Tài khoản khách gửi lượng tiền làm phí dịch vụ vào hợp đồng điều phối. 4. Số dư khả dụng để sử dụng dịch vụ của địa chỉ hợp đồng khách tăng thêm. 5. Địa chỉ hợp đồng khách trở thành khách hàng của hợp đồng điều phối. 6. Đăng ký thành công
Luồng ngoại lệ:	1. Địa chỉ hợp đồng khách đã là địa chỉ đăng ký sử dụng dịch vụ. 2. Địa chỉ tài khoản khách chưa chuyển thành công tiền vào hợp đồng điều phối.
Yêu cầu phi chức năng:	1. Loại tiền (token) tài khoản khách chuyển vào phải là loại hợp đồng điều phối chấp nhận.

Bảng 4.2: Đặc tả chức năng đăng ký là khách hàng

Bảng 4.2 đặc tả yêu cầu chức năng đăng ký là khách hàng của tác nhân khách hàng. Khách hàng khi đăng ký sử dụng dịch vụ sinh số ngẫu nhiên cho hợp đồng thông minh phải kiểm tra liệu địa chỉ hợp đồng đã đăng ký hay chưa. Khi đăng ký sử dụng dịch vụ, khách hàng cần gửi một lượng tiền quy định trước vào địa chỉ của hợp đồng điều phối. Lượng tiền này sẽ trừ dần mỗi khi hợp đồng khách sử dụng dịch vụ.

Chức năng hủy đăng ký là khách hàng:	
Mô tả:	Hợp đồng thông minh của khách hàng muốn hủy đăng ký sử dụng dịch vụ sinh số ngẫu nhiên.
Tác nhân:	Khách hàng
Luồng cơ bản:	1. Tài khoản khách hàng yêu cầu hủy đăng ký cho hợp đồng sử dụng dịch vụ sinh số ngẫu nhiên. 2. Hợp đồng điều phối (VRFCoordinator) sẽ kiểm tra địa chỉ hợp đồng khách chưa tồn tại. 3. Hủy đăng ký thành công.
Luồng ngoại lệ:	1. Người gửi yêu cầu hủy không phải là tài khoản khách hàng đã đăng ký. 2. Hợp đồng khách chưa đăng ký sử dụng dịch vụ sinh số ngẫu nhiên.
Yêu cầu phi chức năng:	1. Chỉ tài khoản người đăng ký, tài khoản được người đăng ký ủy quyền mới có khả năng hủy đăng ký sử dụng dịch vụ của hợp đồng khách,

Bảng 4.3: Đặc tả chức năng hủy đăng ký là khách hàng

Bảng 4.3 đặc tả chức năng hủy đăng ký sử dụng dịch vụ của khách hàng. Chức năng yêu cầu chỉ người đăng ký sử dụng dịch vụ hoặc người được ủy quyền từ người đăng ký mới có thể hủy dịch vụ. Hợp đồng khách phải là hợp đồng đã đăng ký thành công trước đó.

Chức năng đăng ký Oracle:	
Mô tả:	Địa chỉ tài khoản khách muốn đăng ký cung cấp dịch vụ sinh số ngẫu nhiên.
Tác nhân:	Chủ sở hữu Oracle
Luồng cơ bản:	1. Tài khoản Oracle yêu cầu đăng ký cung cấp dịch vụ sinh số ngẫu nhiên. 2. Hợp đồng điều phối (VRFCoordinator) sẽ kiểm tra Oracle chưa tồn tại 3. Đăng ký thành công.
Luồng ngoại lệ:	1. Hợp đồng điều phối (VRFCoordinator) phát hiện Oracle đã tồn tại
Yêu cầu phi chức năng:	1. Chỉ yêu cầu khóa công khai của chủ sở hữu Oracle như là khóa công khai để sinh số ngẫu nhiên đối với Oracle này. 2. Không yêu cầu khóa bí mật. 3. Chỉ người quản lý hợp đồng điều phối mới có thể thêm Oracle.

Bảng 4.4: Đặc tả chức năng đăng ký là Oracle

Bảng 4.4 đặc tả yêu cầu chức năng đăng ký là Oracle của tác nhân chủ quản lý. Hệ thống thiết kế ra các khuôn về Oracle, bất kỳ cá nhân nào đều có thể lấy về và sử dụng, tuy nhiên phải đăng ký làm người cung cấp dịch vụ với hợp đồng điều phối. Khi đăng ký phải đăng ký cả khóa công khai dùng để sinh số ngẫu nhiên, điều này phục vụ cho chức năng xác thực bằng chứng của hợp đồng điều phối. Quá trình này yêu cầu địa chỉ Oracle chưa từng được đăng ký và không yêu cầu gửi khóa bí mật.

Chức năng hủy đăng ký Oracle:	
Mô tả:	Địa chỉ tài khoản khách muốn hủy đăng ký cung cấp dịch vụ sinh số ngẫu nhiên.
Tác nhân:	Chủ sở hữu Oracle
Luồng cơ bản:	1. Tài khoản Oracle yêu cầu hủy đăng ký cung cấp dịch vụ sinh số ngẫu nhiên. 2. Hợp đồng điều phối (VRFCoordinator) sẽ kiểm tra Oracle đã tồn tại 3. Hủy Đăng ký thành công.
Luồng ngoại lệ:	1. Hợp đồng điều phối (VRFCoordinator) phát hiện Oracle chưa tồn tại
Yêu cầu phi chức năng:	1. Chỉ người quản lý hợp đồng điều phối mới có thể thêm Oracle.

Bảng 4.5: Đặc tả chức năng hủy đăng ký là Oracle

Bảng 4.5 đặc tả chức năng hủy đăng ký làm Oracle của người quản lý hợp đồng điều phối. Quá trình hủy đăng ký yêu cầu phải là người quản lý hoạt động mới có quyền hủy đăng ký của Oracle, và địa chỉ Oracle phải được đăng ký trước đó.

4.2.3 Thiết kế lớp hợp đồng thông minh

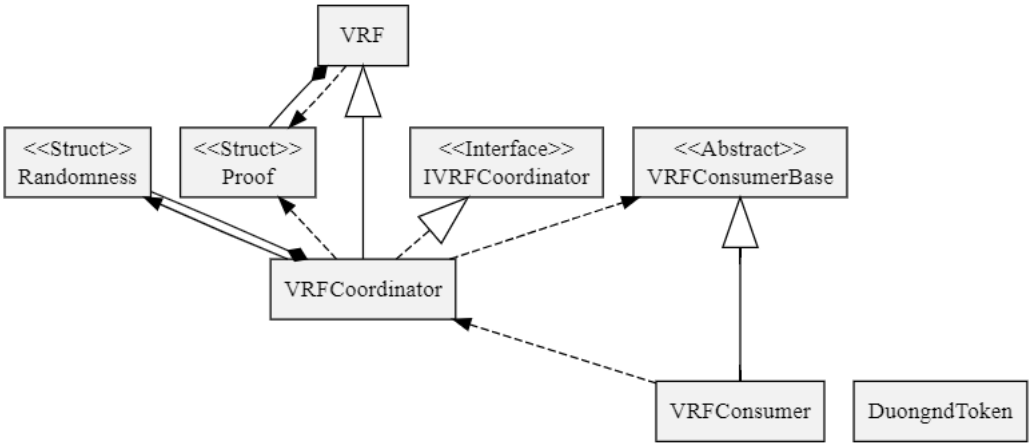
Với luồng hoạt động chính như sau:

1. Hợp đồng khách yêu cầu số ngẫu nhiên và gọi tới hợp đồng điều phối.
2. Hợp đồng điều phối phát lên sự kiện bao gồm thông tin người yêu cầu, số định danh cho yêu cầu.
3. Oracle nghe được và tính toán bằng chứng ngẫu nhiên.
4. Oracle thông qua một địa chỉ ví gửi bằng chứng ngẫu nhiên tới hợp đồng điều phối.
5. Hợp đồng điều phối xác thực ngẫu nhiên.
6. Hợp đồng điều phối gửi kết quả trả lại cho hợp đồng khách.

Dựa trên luồng hoạt động chính, từ đó đồ án thực hiện thiết kế lớp đối với hợp đồng thông minh.

Hình 4.4 mô tả mối quan hệ giữa các thực thể. VRFCoordinator kế thừa từ VRF

và IVRFCoordinator. VRF đảm bảo việc xác thực bằng chứng là hợp lệ và tính toán kết quả ngẫu nhiên trả về cho hợp không khách. IVRFCoordinator định ra các chức năng quản lý Oracle và hợp đồng khách. VRFConsumerBase định ra các hàm bắt buộc VRFConsumer phải triển khai để tương tác với VRFCoordinator và nhận kết quả ngẫu nhiên. DuongndToken, được kế thừa từ chuẩn ERC20 Openzeppelin, là hợp đồng thông minh của loại tiền phần thưởng cho các Oracle cũng như là chi phí cho các dịch vụ ngẫu nhiên.



Hình 4.4: Thiết lớp hợp đồng thông minh

Trong bảng 4.6 thể hiện các thuộc tính trong Proof. Ngoài $pk, gamma, c, s, seed$ như trong thuật toán sinh số ngẫu nhiên có thể kiểm chứng trình bày trong chương 3 mỗi Proof sẽ chứa các bằng chứng của phép nhân vô hướng trên đường cong elliptic cũng như tổ hợp tuyến tính các điểm trên đường cong. Việc thêm các trường thông tin này giúp tránh phép nhân vô hướng thực hiện trên hợp đồng thông minh bằng cách sử dụng hàm ecrecover giả chữ ký được trình bày ở phụ lục B.

Proof	
pk	uint256[2]
gamma	uint256[2]
c	uint256
s	uint256
seed	uint256
uWitness	address
cGammaWitness	uint256[2]
sHashWitness	uint256[2]
zInv	uint256

Bảng 4.6: Struct Proof

Bảng 4.7 thể hiện các trường thông tin của một kết quả ngẫu nhiên. Count thể hiện đã có bao nhiêu Oracle trả về bằng chứng hợp lệ đối với mỗi requestId của

khách hàng. Khi đủ ngưỡng sẽ tự động trả về kết quả cho hợp đồng khách.

Randomness	
count	uint256
randomness	uint256

Bảng 4.7: Struct Randomness

Bảng 4.8 thể hiện chi tiết hợp đồng xác thực. trong đó các trường thông tin sẽ được lưu trên storage EVM bao gồm:

1. Owner: Người làm chủ hợp đồng, có quyền onlyOwner.
2. DuongndToken: địa chỉ hợp đồng của đơn vị tiền.
3. fee: Xác định chi phí mỗi lần hợp đồng khách yêu cầu dịch vụ.
4. thresshold: ngưỡng số lượng các Oracle trung thực sẽ gửi kết quả về hợp đồng khách.
5. Oracles: ánh xạ lưu lại các Oracle đã đăng ký thành công.
6. isConsumer: ánh xạ lưu lại những hợp đồng khách đăng ký sử dụng dịch vụ.
7. balances: số dư của hợp đồng khách hàng còn lại.
8. nonces: ánh xạ lưu lại số các yêu cầu của mỗi khách hàng.
9. requestIds: ánh xạ lưu lại kết quả ngẫu nhiên đối với mỗi requestId.

Đối với các phần logic của hợp đồng xác thực được lưu trong EVM code với các chức năng:

1. computeRequestId: trả về requestId từ địa chỉ người gửi và số nonce.
2. registerOracle: hàm cho phép đăng ký là một Oracle cung cấp dịch vụ.
3. RandomRequested: sự kiện sẽ phát lên logs, cho biết đã khách hàng yêu cầu dịch vụ.
4. onlyOwner: modifier hạn chế quyền chỉ cho owner.
5. hashOfKey: hàm băm từ điểm trên đường cong về bytes32.
6. fulfillRandomness: hàm sẽ được gọi từ Oracle trả về bằng chứng ngẫu nhiên.
7. requestRandomness: hàm phát ra sự kiện yêu cầu số ngẫu nhiên.
8. addConsumer: đăng ký sử dụng dịch vụ ngẫu nhiên cho khách hàng.
9. removeConsumer: Xóa khách hàng khỏi danh sách được sử dụng dịch vụ.

Một hợp đồng khách hàng đơn giản sử dụng dịch vụ sinh số ngẫu nhiên được

VRFCoordinator
Public: DuongndToken: ERC20 owner: address fee: uint256 threshold: uint256 oracles: mapping(bytes32=>address) isConsumer: mapping(address=>bool) balances: mapping(address=>uint256) nonces: mapping(address=>uint256) requestIds: mapping(uint256=>Randomness)
Private: computeRequestId(uint256,sender): (uint256,uint256) External: registerOracle(address,uint256[2]) Public: «event» RandomRequested(address, uint256, uint256) «event» Transfer(address,address,uint256) «modifier» onlyOwner() constructor(address,uint256,uint256) hashOfKey(uint256[2]): bytes32 fulfillRandomness(Proof,address) requestRandomness(): uint256 addConsumer(address,uint256) removeConsumer(address)

Bảng 4.8: VRFCoordinator

thể hiện dưới bảng 4.9. Trong đó ta chỉ cần chỉ định hợp đồng VRFCoordinator thông qua địa chỉ. Cài đè hàm fulfillRandomness kế thừa từ hợp đồng trừu tương VRFConsumerBase (bảng 4.10) để lưu lại kết quả ngẫu nhiên. Cài đặt hàm requestRandomness để gửi yêu cầu số ngẫu nhiên.

VRFConsumer
Public: vrfCoordinator: address randomNumber: uint256
Internal: fulfillRandomness(uint256,uint256) Public: constructor() requestRandomness()

Bảng 4.9: VRFConsumer

VRFConsumerBase
Private: vrfCoordinator: address
Internal: «abstract»fulfillRandomness(uint256,uint256) External: rawFulfillRandomness(uint256,uint256) Public: constructor()

Bảng 4.10: VRFConsumerBase

Bảng 4.11 mô tả hợp đồng VRF. Đây là hợp đồng có nhiệm vụ xác thực bằng chứng ngẫu nhiên và tính toán kết quả từ bằng chứng ngẫu nhiên này. VRF có các trường thuộc tính

1. GROUP_ORDER: bậc của đường cong secp256k1.
2. FIELD_SIZE: đặc số trường hữu hạn F_p.
3. WORD_LENGTH_BYTES: độ dài của kiểu dữ liệu uint256.
4. SQRT_POWER: số mũ để tính căn bậc hai.
   Với SQRT_POWER $= \frac{(\text{FIELD\_SIZE}+1)}{4}$
   $q =$ FIELD_SIZE, $q \equiv 3 \mod 4, x \equiv r^2 \mod q$, suy ra $x^{\text{SQRT\_POWER}} = \pm r \mod q$
5. HASH_TO_CURVE_HASH_PREFIX: tiền tố thực hiện phép băm từ một số nguyên lớn tới một điểm trên đường cong.
6. SCALAR_FROM_CURVE_POINTS_HASH_PREFIX: tiền tố thực hiện phép băm từ một điểm trên đường cong tới một số nguyên lớn.
7. VRF_RANDOM_OUTPUT_HASH_PREFIX: tiền tố thực hiện phép băm từ bằng chứng hợp lệ ra kết quả ngẫu nhiên.

Các hàm của hợp đồng VRF thể hiện trên bảng 4.11 dùng để tính toán các phép tính trên đường cong Secp256k1 và xác thực bằng chứng ngẫu nhiên được gửi từ Oracle.

1. bigModExp: hàm tính base$^{\text{e}} \mod p$.
2. squareRoot: Hàm tính thử nghiệm căn bậc hai.
3. isOnCurve: kiểm tra một cặp giá trị có là một điểm trên Secp256k1.
4. fieldHash: hàm băm từ một bytes tới một số lớn thuộc F_p.
5. newCandidateSecp256k1Point: Tính toán một cặp giá trị là ứng cử viên cho

VRF
Private: GROUP_ORDER FIELD_SIZE WORD_LENGTH_BYTES SQRT_POWER Internal: HASH_TO_CURVE_HASH_PREFIX SCALAR_FROM_CURVE_POINTS_HASH_PREFIX VRF_RANDOM_OUTPUT_HASH_PREFIX
Internal: bigModExp squareRoot ySquared isOnCurve fieldHash newCandidateSecp256k1Point hashToCurve ecmulVerify projectiveSub projectiveMul projectiveECAdd affineECAdd verifyLinearCombinationWithGenerator linearCombination scalarFromCurvePoints verifyVRFProof randomValueFromVRFProof

Bảng 4.11: VRF

điểm trên đường cong.

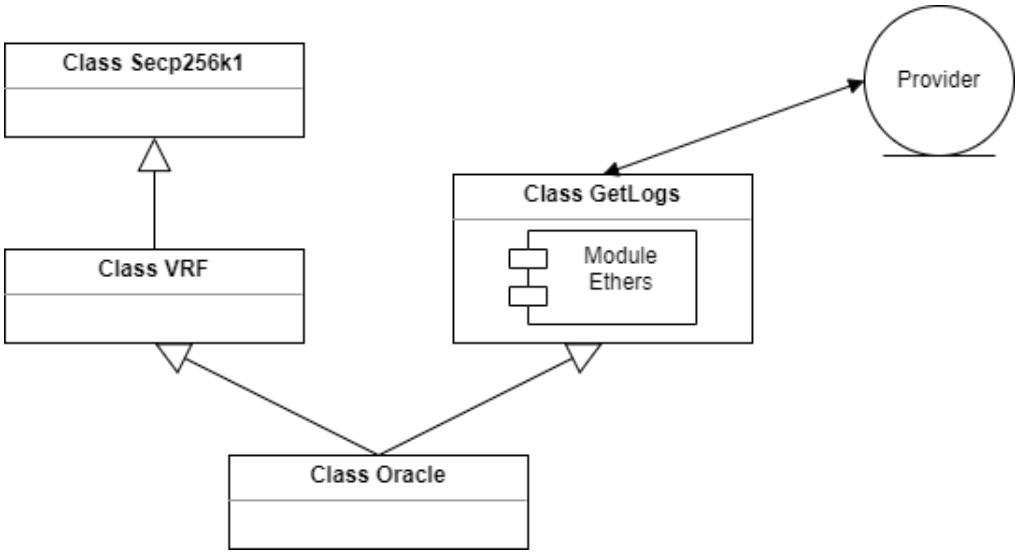
6. hashToCurve: hàm băm từ đầu vào tới một điểm trên đường cong.
7. ecmulVerify: hàm xác thực phép nhân vô hướng trên đường cong.
8. projectiveECAdd: cộng hai điểm trên đường cong với hệ tọa độ chiếu.
9. affineECAdd: cộng hai điểm trên đường cong với hệ tọa độ affine.
10. verifyLinearCombinationWithGenerator: Sử dụng kỹ thuật giả chữ ký xác thực kết quả tổ hợp tuyến một điểm trên đường cong với một phần tử sinh đường cong.
11. linearCombination: xác thực kết quả tổ hợp tuyến tính hai điểm trên đường cong.

12. scalarFromCurvePoints: hàm băm H_3 dựa trên hàm băm Keccak256.
13. verifyVRFProof: hàm xác thực bằng chứng ngẫu nhiên.
14. randomValueFromVRFProof: hàm băm H_2 dựa trên hàm băm Keccak256.

4.2.4 Thiết kế lớp Oracle

Với hệ thống trình bày trong đồ án, Oracle có hai nhiệm vụ chính. Thứ nhất đó là lắng nghe sự kiện trên nhật ký (logs) của blockchain. Thứ hai đó là mỗi khi nghe được sự kiện thì lập tức tính toán bằng chứng ngẫu nhiên rồi gửi tới cho hợp đồng xác thực VRFCoordinator.

Hình 4.5 mô tả thiết kế của Oracle kế thừa lớp VRF và lớp Getlogs. Về cơ bản lớp VRF dùng để triển khai Oracle giống hệt hợp đồng VRF để triển khai VRFCoordinator.



Hình 4.5: Thiết kế Oracle

Trong lớp Secp256k1 để xây dựng lên lớp VRF (bảng 4.12) có thuộc tính là các tham số công khai của đường cong Secp256k1. Các hàm chức năng với mục đích tính toán số lớn các điểm, tích vô hướng một số với một điểm dựa trên hai hệ tọa độ biểu diễn chiếu (projective) và hai chiều (affine).

Khác ở chỗ hợp đồng VRF với mục đích xác thực bằng chứng ngẫu nhiên hơn nữa trong môi trường của Ethereum đã cài đặt sẵn đường cong Secp256k1. Về phía Oracle thì lớp VRF có chức năng sinh bằng chứng từ khóa bí mật. Do đó có chút khác biệt ở chức năng và các thuộc tính của lớp.

1. G: phần tử sinh của đường cong.
2. PUBLIC_KEY: khóa công khai.
3. PRIVATE_KEY: khóa bí mật sinh số ngẫu nhiên

Secp256k1
Private: GROUP_ORDER FIELD_SIZE G PRIVATE_KEY PUBLIC_KEY
Internal: uint246 mulmod addmod invmod projectiveAdd projectiveSub hashToCurve projectiveMul projectiveECAdd projectiveECMul affineECAdd affineECMul

Bảng 4.12: Lớp Secp256k xây dựng lên VRF

4. addressFromPoint: hàm tính địa chỉ từ một điểm trên đường cong dựa trên thuật toán sinh địa chỉ của Ethereum.
5. vRF: hàm tính bằng chứng ngẫu nhiên theo thuật toán VRF trình bày trong 3.
6. proofVRF: hàm tính bằng chứng ngẫu nhiên phù hợp với hợp đồng xác thực.

Tổng quan lớp GetLogs được thể hiện ở bảng 4.14. Trong đó Module Ethers giúp Oracle kết nối với node trên mạng Ethereum. Lắng nghe sự kiện trên mỗi transaction thông qua chữ ký của sự kiện.

GetLogs
Private: pointer
Interal: parseLog

Bảng 4.14: Lớp Getlogs xây dựng lên Oracle

4.2.5 Thử nghiệm

Trong thử nghiệm thực tế, đồ án dùng các công nghệ được thể hiện trên bảng 4.15:

VRF
Private: SQRT_POWER PRIVATE_KEY PUBLIC_KEY Internal: HASH_TO_CURVE_HASH_PREFIX SCALAR_FROM_CURVE_POINTS_HASH_PREFIX VRF_RANDOM_OUTPUT_HASH_PREFIX
Internal: bigModExp squareRoot ySquared isOnCurve fieldHash newCandidateSecp256k1Point hashToCurve encodePacked addressFromPoint scalarFromCurvePoints vRF proofVRF

Bảng 4.13: Lớp VRF xây dựng lên Oracle

Công nghệ sử dụng	
Ví người dùng	Metamask
Oracle	Nodejs
Hợp đồng thông minh	Solidity
Kết nối với Node	Influra
Môi trường phát triển	Hardhat
Mạng thử nghiệm	Rinkeby
Chuẩn tiền mã hóa	ERC20

Bảng 4.15: Công nghệ sử dụng

Ngoài ra đồ án còn sử dụng thư viện:

1. bn.js: thư viện tính toan số lớn trên nodejs.
2. sol2uml: thư viện phục vụ mục đích vẽ biểu đồ lớp.
3. openzeppelin/contracts: thư viện chuẩn ERC20.
4. ethers: thư viện giúp kết nối với node Influra.
5. chai: thư viện giúp việc kiểm thử code.

Trong thử nghiệm đồ án, trình tự được thực hiện the trình tự:

1. Triển khai đồng tiền thay thế (Alter token) đối với hệ thống theo chuẩn ERC20 của Oppenzepelin tại địa chỉ
   $0xFF6afB47AcB97A849ba51B20E57438d340EE8fbD$
2. Địa chỉ triển khai đồng tiền ERC20 sẽ gửi một lượng tiền nhất định ERC20 tới cho hợp đồng khách hàng để phục vụ mục đích đăng ký dịch vụ.
3. Các địa chỉ Oracle và khách hàng đều phải chưa một lượng tiền ETH (native token) để nhằm mục đích triển khai đăng ký.
4. Triển khai hợp đồng điều phối VRFCoordinator tại địa chỉ
   $0xFE5725db462CC0a4ACa15FD9317298b1a52582b5$
5. Đăng ký làm Oracle cho địa chỉ
   $0x45eFc65fF7e9ed85664924633416129Bc340Dcb8$
6. Đăng ký Oracle cho địa chỉ
   $0x9Fb0dcaD42e3eBde8D1329a36A3F49dfbCE2141C$
7. Triển khai hợp đồng khách hàng và yêu cầu số ngẫu nhiên.
8. Đăng ký sử dụng dịch vụ sinh số ngẫu nhiên đối với hợp đồng khách.

4.3 Đánh giá

Thử nghiệm cho thấy kết quả ngẫu nhiên được trả về trong thời gian nhanh nhất là 2 block. Thời gian mỗi block sẽ phụ thuộc vào từng mạng chuỗi khối có máy ảo Ethereum. Điều này sẽ là nhược điểm của hệ thống sinh số ngẫu nhiên có kiểm chứng bởi vì cần ít nhất một block để gửi yêu cầu và cần ít nhất một block để xác thực và gửi kết quả trả lại hợp đồng khách hàng. Đổi lại nhược điểm đó là hệ thống sẽ giảm rủi ro bị thao túng đi nhiều lần do mô hình phân tán của các Oracle.

Trong quá trình thực hiện, điều em tâm đắc nhất đó là đề xuất mô hình phân tán cho bên Oracle. Đây là một quá trình khó khăn bởi vì việc tăng yêu cầu số lượng Oracle cho một yêu cầu ngẫu nhiên sẽ làm tăng một lượng chi phí cho yêu cầu. Hơn nữa việc cho phép đăng ký làm bên cung cấp dịch vụ sẽ phải có cơ chế phạt các Oracle không trung thực, hay bị từ chối dịch vụ,...Chưa kể tới khi có quá nhiều Oracle thì việc tăng số lượng nhà cung cấp dịch vụ sẽ khiến phần thường nhận được các bên cung cấp nhỏ.

Điều tâm đắc trong quá trình thử nghiệm đó là thiết kế thuật toán lắng nghe sự kiện yêu cầu ngẫu nhiên cho các Oracle. Việc thiết kế này sẽ phải xem xét theo chữ ký sự kiện và được phát ra từ hợp đồng điều phối. Sẽ có một con trỏ lưu lại chỉ dấu đã quét cho Oracle. Mỗi một chu kỳ (đảm bảo nhỏ hơn thời gian trung bình thợ

đào công bố block mới) sẽ lấy dữ liệu một lần. Khi nghe được yêu cầu sẽ triển khai thuật toán VRF. Cũng mỗi chu kỳ này các Oracle sẽ xem xét các định danh yêu cầu trên hợp đồng thông minh, nếu quá 10 block mà yêu cầu chưa được thực hiện thì Oracle sẽ thực hiện điều này.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

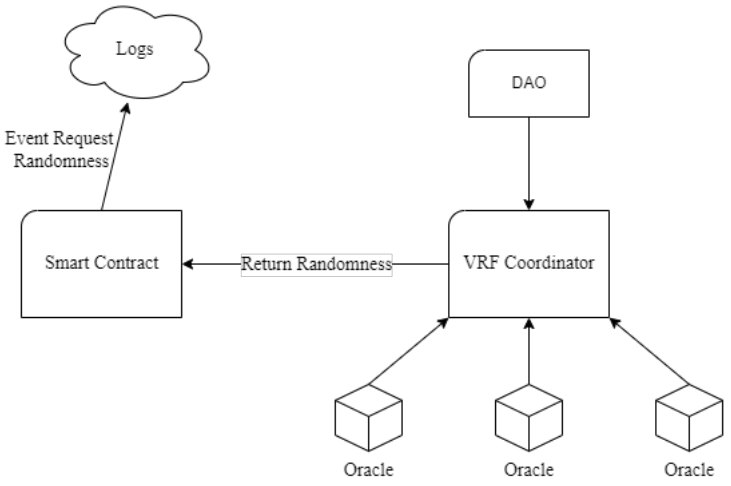
Như vậy đồ án đã đạt được mục tiêu đề ra:

1. Giải quyết bài toán sinh số ngẫu nhiên trên mạng chuỗi khối.
2. Đề xuất giải pháp giảm thiểu rủi ro, tăng tính phân tán trong mô hình sinh số ngẫu nhiên có thể kiểm chứng.

Ngoài ra, qua quá trình nghiên cứu và hoàn thành đồ án, em cũng rèn luyện thêm kỹ năng tìm kiếm, đọc tài liệu chuyên ngành; chọn lọc, tổng hợp kiến thức và chế bản đồ án bằng LaTeX.

Trong giai đoạn phát triển tiếp theo, em đề xuất hai ý tưởng phát triển đồ án.

Thứ nhất, DAO (Decentralized Autonomous Organization) sẽ là người chủ sở hữu của hợp đồng thông minh xác thực.

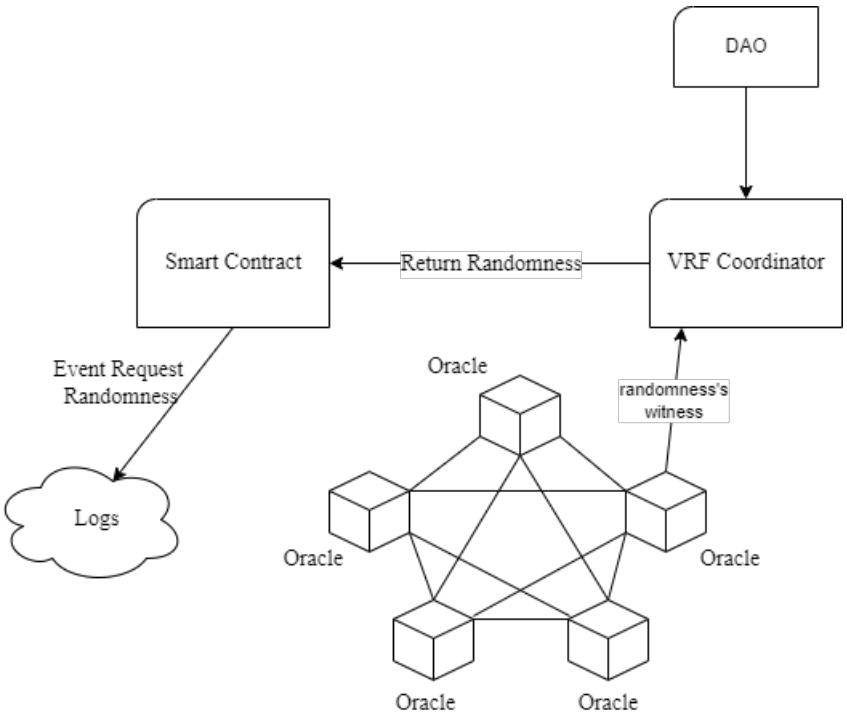


Hình 4.6: DAO tiếp quản hợp đồng thông minh VRF Coordinator

DAO giúp loại bỏ sự phụ thuộc vào một cá nhân hay tổ chức của hợp đồng xác thực. Dẫn ra cấu trúc kiểm soát phi tập trung từ việc trả thưởng cho các Oracle, trừng phạt các Oracle gian lận hay từ các tác vụ đơn giản như thêm Oracle được cung cấp dịch vụ...

Thứ hai, sử dụng kỹ thuật ngưỡng chữ ký được trình bày trong [5] để giảm thiểu chi phí tổng hợp các kết quả từ Oracle bằng cách đẩy việc này lên off-chain.

Trong mô hình được trình bày ở hình 4.7, mỗi Oracle đóng vai trò như là một node trong mạng lưới. Xác thực và tổng hợp các kết quả ngẫu nhiên trên từng node và gửi kết quả cuối cùng với chữ ký ngưỡng sẽ giảm được chi phí xác thực bằng chứng ngẫu nhiên nhiều lần với mỗi yêu cầu của hợp đồng khách hàng, mà vẫn đảm bảo được tính trung thực của các Oracle.



Hình 4.7: Mô hình sinh số ngẫu nhiên sử dụng mạng lưới các Oracle

REFERENCE

[1] J. Chen **and** S. Micali, "Algorand," *arXiv preprint arXiv:1607.01341*, 2016.

[2] Chain Link, *Verifiable source of randomness for smart contract developers*. **url**: `https://chain.link/vrf` (**urlseen** 20/07/2022).

[3] D. Papadopoulos, D. Wessels, S. Huque **andothers**, "Making nsec5 practical for dnssec," *Cryptology ePrint Archive*, 2017.

[4] T. Icart, "How to hash into elliptic curves," **in***Annual International Cryptology Conference* Springer, 2009, **pages** 303–316.

[5] D. R. Stinson **and** R. Strobl, "Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates," **in***Australasian Conference on Information Security and Privacy* Springer, 2001, **pages** 417–434.

[6] G. Wood **andothers**, "Ethereum: A secure decentralised generalised transaction ledger,"

[7] Ethereum White Paper, *Ethereum virtual machine*. **url**: `https://ethereum.org/vi/developers/docs/evm/` (**urlseen** 20/07/2022).

[8] D. Johnson, A. Menezes **and** S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, **jourvol** 1, **number** 1, **pages** 36–63, 2001.

[9] T. Pornin, "Rfc 6979: Deterministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm (ecdsa)," *The Internet Engineering Task Force*, 2013.

PHỤ LỤC

A. SƠ LƯỢC VỀ MẠNG BLOCKCHAIN

Trong phần này em sẽ trình bày sơ lược về mạng blockchain Ethereum. Một vài sự thay đổi quan trọng của Ethereum so với blockchain Bitcoin. Sự thay đổi này sẽ có vai trò quan trọng trong việc triển khai thuật toán sinh số ngẫu nhiên có thể kiểm chứng trên mạng blockchain Ethereum. Kể từ đây, nếu không nói gì thêm thì khi em nói tới mạng blockchain thì có nghĩa em đang nói tới mạng blockchain ethereum.

A.1 Sự đột phá của Ethereum

Như ta đã biết mạng blockchain Bitcoin sẽ theo dõi sự thay đổi trạng thái sở hữu của các đơn vị tiền mã hóa. Ta có thể xem Bitcoin là một hệ thống trạng thái đồng thuận phân tán trong đó các transaction làm thay đổi trạng thái. Các transaction sẽ được gom vào một block, các block tạo nên blockchain dưới cơ chế đồng thuận.

Đối với Ethereum, thay vì chỉ theo dõi sự thay đổi trạng thái sở hữu của các đơn vị tiền mã hóa mà nó theo dõi sự thay đổi trạng thái của dữ liệu lưu trữ. Ví dụ như lưu trữ cặp key - value "duongnd - blog's author". Ethereum có bộ nhớ lưu trữ cả code và dữ liệu và sử dụng các block để theo dõi sự thay đổi trạng thái qua thời gian. Giống như một máy tính lưu trữ, Ethereum có thể load code vào Ethereum virtual machine (EVM) và chạy code đó. Tuy nhiên sẽ có hai điểm khác biệt đó là sự thay đổi trạng thái của EVM hoạt động dưới thuật toán đồng thuận và trạng thái của EVM được phân tán tới tất cả các node trong mạng blockchain.

Ethereum có thể thực thi các program được lưu trữ trong EVM. Hay nói cách khác, Ethereum có thể thực thi bất kỳ chương trình nào mà máy Turing có thể làm với bộ nhớ hữu hạn.

Do vậy điểm đột phá của Ethereum đó là sự kết hợp giữa máy tính lưu trữ thực thi chương trình và blockchain phân tán.

Tuy nhiên các transaction thực hiện chương trình trên blockchain Ethereum cần được xác thực sau đó lan truyền tới các node khác trong mạng. Điều này đặt ra yêu cầu phải giải quyết các đoạn code chạy vô hạn như vòng lặp vô hạn hay hạn chế các chương trình tiêu tốn quá nhiều công tính toán khi thực thi hay xác thực. Trong khi đó máy tính lại không thể biết trước được liệu khi nào chương trình sẽ kết thúc. Từ đây Ethereum đưa ra khái niệm về gas.

Với mỗi lệnh cơ bản (opcodes) sẽ cần chỉ rõ ra số lượng gas tiêu tốn. Khi transaction được gửi thì các node sẽ thực thi chương trình và đếm chính xác số gas đã tiêu thụ. Nói cách khác khi thực hiện một transaction hoặc thực thi code hợp

đồng sẽ mất một lượng gas cố định. Ví dụ theo Ethereum Yellow Paper ([6])

1. Cộng hai số nguyên không dấu tốn 3 gas
2. Tính hàm băm Keccal mất 30 gas cộng với 6 gas mỗi 256bits của dữ liệu đầu vào
3. Gửi một transaction mất 21000 gas

Nếu vượt quá số gas giới hạn của transaction hoặc giới hạn gas của block thì transaction sẽ được hoàn lại trạng thái cũ. Chú ý rằng tuy transaction thực hiện không thành công (reverted) nhưng tài khoản gửi vẫn mất chi phí thực hiện transaction này.

Gas là một thành phần quan trọng của Ethereum và đóng vai trò kép: ngăn chặn các vòng lặp vô hạn hoặc sự lãng phí công tính toán trong mạng và là thước đo phần thưởng cho những người khai thác khối. Người khởi tạo mỗi giao dịch được yêu cầu đặt giới hạn cho số lượng tính toán mà họ sẵn sàng trả. Do đó, hệ thống gas không cho phép những kẻ tấn công gửi các giao dịch "rác", vì chúng phải trả lượng tiền tương xứng cho các tài nguyên tính toán và lưu trữ mà chúng sử dụng trên máy ảo Ethereum.

Nếu EVM kết thúc thực hiện thành công transaction mà không tiêu hết gas, chi phí sẽ được thanh toán cho người khai thác (miner) dưới dạng phí giao dịch dựa trên giá gas được chỉ định.

$$\text{miner fee} = \text{gas cost} * \text{gas price}$$

Lượng gas chưa dùng hết sẽ hoàn lại cho người thực hiện giao dịch với:

$$\text{remaining gas} = \text{gas limit} - \text{gas cost}$$

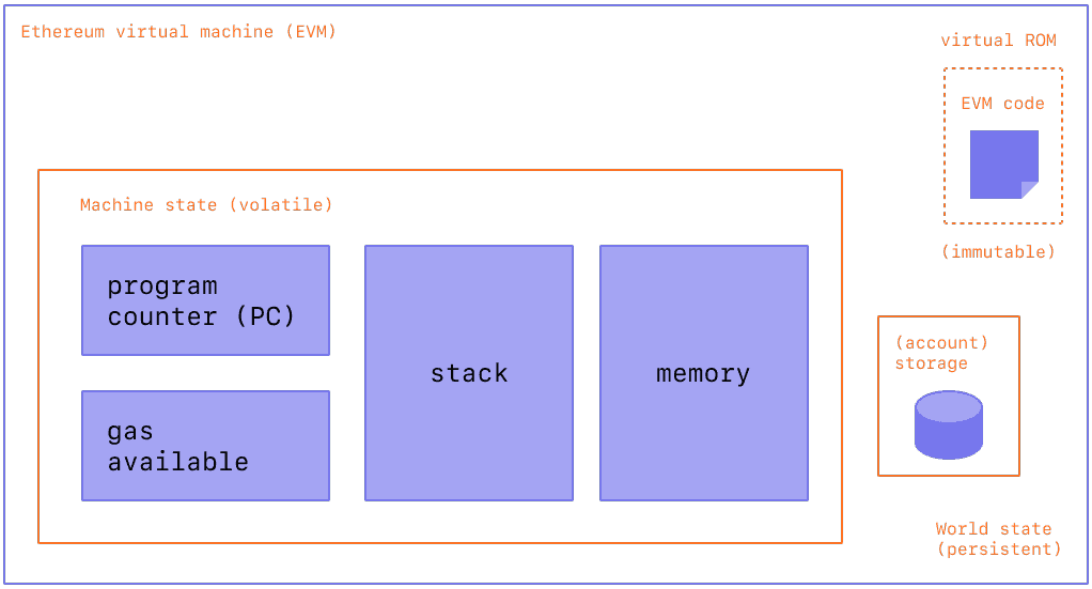
$$\text{refunded ether} = \text{remaining gas} * \text{gas price}$$

Với việc hạn chế về gas sẽ là thách thức đối với bản thân em khi triển khai thuật toán VRF cũng như triển khai hợp đồng khách để giảm thiểu chi phí thực hiện transaction.

A.2 Máy ảo ethereum

Trái tim của giao thức và hoạt động Ethereum là máy ảo Ethereum (EVM).

Ta thấy trên hình A.1 máy ảo có bộ nhớ memory, stack, PC, ROM, storage giống như một máy tính thông thường. Mỗi hợp đồng thông minh trước khi được triển



Hình A.1: Sơ đồ kiến trúc EVM (theo [7])

khai lên mạng sẽ được biên dịch dưới dạng bytes code. Phần logic code tức là các hàm (function) sẽ được lưu trữ tại EVM code, đây sẽ là phần không được thay đổi một khi đã được khởi tạo. Các trạng thái của hợp đồng thông minh sẽ được lưu trữ tại storage. Chi tiết về hợp đồng thông minh em sẽ trình bày ở phần sau của phụ lục này.

Trạng thái của máy ảo Ethereum (Ethereum world state) là tập các ánh xạ từ một địa chỉ tới một tài khoản. Trong cài đặt của Ethereum, địa chỉ được tính từ khóa công khai của người dùng đối với tài khoản người dùng. Đối với địa chỉ của hợp đồng được tính từ địa chỉ người triển khai hợp đồng và số nonce của tài khoản triển khai.

A.3 Các tài khoản

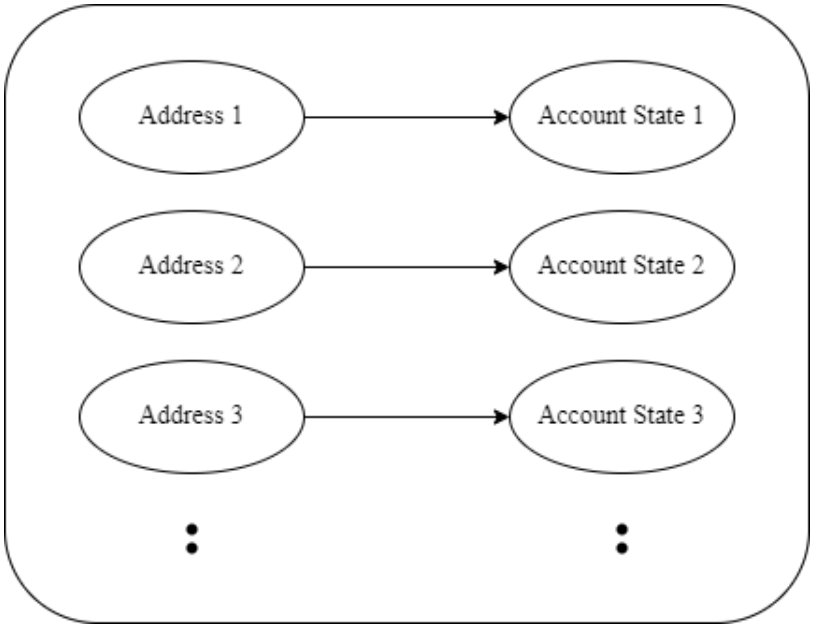
Trong thế giới Ethereum có hai loại tài khoản (Account). Đó là tài khoản thuộc quyền sở hữu của người dùng (External Owner Account) và hợp đồng thông minh (Smart contract).

Về điểm chung cả hai loại tài khoản đều có chức năng như:

1. Nhận và gửi ETH, các đồng token.
2. Tương tác với các contract đã được triển khai lên mạng.

Mỗi tài khoản trên mạng Ethereum đều có bốn trường thông tin:

1. nonce: Đếm số transaction được gửi từ account, đối với contract account thì đếm số contract được tạo mới bởi nó.



Hình A.2: Trạng thái của máy ảo Ethereum

2. balance: Số wei (đơn vị của ether) được giữ bởi account.
3. storage root: Giá trị băm của root node Merkle Patricia trie - cấu trúc dữ liệu để lưu trữ trên Ethereum.
4. code hash:Giá trị hàm băm của contract code.

Chú ý rằng đối với tài khoản người dùng thì hai trường thông tin storage root và code hash là mặc định.

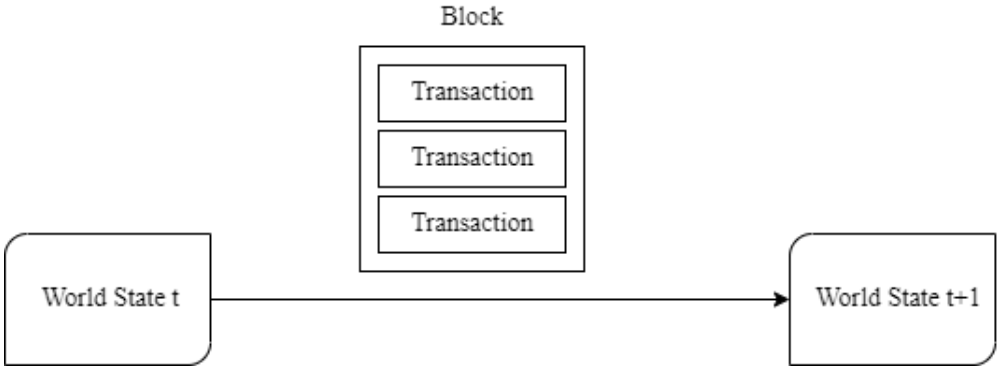
Tài khoản người dùng	Hợp đồng thông minh
Tạo tài khoản miễn phí	Mất phí triển khai hợp đồng
Bắt đầu một transaction	Không thể bắt đầu một transaction
Transactioin giữa hai người chỉ có thể là ETH/token	Transaction giữa người dùng và hợp đồng thông mình có thể là kích hoạt code, tạo hợp đồng mới...

Bảng A.1: Bảng so sánh tài khoản người dùng và tài khoản hợp đồng

Việc hiểu rõ cấu trúc của các loại tài khoản là cần thiết đối với xây dựng mô hình oracle trả về số ngẫu nhiên trong đồ án này.

A.4 Transaction

Transaction là thông điệp (message) đã ký được gửi từ External owner account (EOA). Transaction sẽ được gửi đến ethereum network xác thực và gộp vào thành một block. Transaction có thể gọi là một hàm chuyển trạng thái trong trạng thái máy ảo (state machine). Và chỉ có transaction mới làm thay đổi trạng thái máy ảo (A.3). Có hai loại transaction đó là:



Hình A.3: Transaction

1. Triển khai, tạo hợp đồng thông minh mới tới tài khoản hợp đồng (Contract creation).
2. Gọi thông điệp tới các tài khoản, cả tài khoản người dùng và tài khoản hợp đồng (Message call).

Đối với transaction tạo hợp đồng mới, phần code khởi tạo sẽ tạo ra hợp đồng thông minh mới được lưu trên trạng thái máy ảo với logic hợp đồng được lưu tại EVM code, các trường thuộc tính được lưu tại bộ nhớ lâu dài (storage).

Đối với transaction tương tác với hợp đồng đã được triển khai, dữ liệu đầu vào sẽ tương tác với logic code được lưu ở trên EVM code, thực thi và lưu giá trị (nếu có) vào bộ nhớ lâu dài (storage).

Bảng A.3 cho thấy các thành phần cấu tạo nên một transaction.

Transaction	
nonce	Số dùng một lần
gasPrice	Giá của một đơn vị gas (wei)
gasLimit	Số lượng gas tối đa cho transaction
to	Địa chỉ nhận transaction (0 nếu khởi tạo hợp đồng)
value	Số lượng ether gửi cho người nhận
v,r,s	Chữ ký transaction
init or data	Dữ liệu gửi kèm transaction

Bảng A.2: Các thành phần của một transaction

A.5 Oracle

Oracles là những hệ thống có thể cung cấp nguồn dữ liệu bên ngoài cho các hợp đồng thông minh Ethereum. Thuật ngữ 'Oracle' xuất phát từ thần thoại Hy Lạp, nơi nó dùng để chỉ một người giao tiếp với các vị thần, người có thể nhìn thấy tầm

nhìn về tương lai. Trong bối cảnh của blockchain, oracle là một hệ thống có thể trả lời các thông tin bên ngoài Ethereum. Các oracles là các hệ thống không đáng tin cậy vì chúng hoạt động trên các nguyên tắc tập trung.

Tuy nhiên ý tưởng về mạng các oracle đã được đề xuất từ trước để khắc phục yếu điểm tập trung của hệ thống, phù hợp với nguyên tắc phi tập trung của mạng chuỗi khối.

A.6 DAO

DAO (Decentralized autonomous organizations) là một cách an toàn để làm việc với một nhóm người có cùng mục tiêu. Không một cá nhân nào có quyền ra quyết định mà không được sự chấp thuận của nhóm. Mỗi hành động phải được đề xuất và bình chọn, điều này đảm bảo rằng bất cứ ai trong nhóm đều có thể tham gia vào quyền điều hành.

B. KIẾN THỨC TOÁN LIÊN QUAN VÀ KỸ THUẬT GIẢM GAS

Trong phần này em trình bày về các kiến thức toán học quan trọng có liên quan tới kỹ thuật làm giảm chi phí gas trong khi triển khai hợp đồng thông minh phục vụ cho giải quyết bài toán đặt ra ban đầu. Em sẽ trình bày một vài phương án tiếp cận một vấn đề đặt ra, việc này sẽ phục vụ cho việc phân tích, lựa chọn công nghệ sử dụng của em tại chương 4.

B.1 Các tính chất của VRF

B.2 Đường cong Elliptic Secp256k1

Hệ các điểm thỏa mãn phương trình Weierstrass trên trường F_p

Phương trình Weierstrass:

$$y^2 = x^3 + ax + b$$

Tham số	Giá trị
p	0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F
a	0x0
b	0x7
n	0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE BAAEDCE6AF48A03BBFD25E8CD0364141
Gx	0x79BE667EF9DCBBAC55A06295CE870B07 029BFCDB2DCE28D959F2815B16F81798
Gy	0x483ADA7726A3C4655DA4FBFC0E1108A8 FD17B448A68554199C47D08FFB10D4B8
f	1

Bảng B.1: Bảng tham số của đường cong secp256k1

Secp256k1 là một nhóm giao hoán với phép tính cộng có bậc là số nguyên tố. Do đó theo bổ đề B.4.2 secp256k1 là một nhóm cyclic với phép tính cộng.

B.3 ECDSA và giả chữ ký

ECDSA (Elliptic Curve Digital Signature Algorithm), là một lược đồ chữ ký dựa trên mật mã đường cong elliptic (ECC). ECDSA dựa vào phép toán của các nhóm cyclic và đường cong elliptic trên các trường hữu hạn. Các khóa và chữ ký ECDSA ngắn hơn trong RSA cho cùng mức độ bảo mật. Chữ ký ECDSA 256-bit có cùng độ bền bảo mật như chữ ký RSA 3072-bit.

Trong phần này em đưa ra một kỹ thuật giả chữ ký tận dụng precompile của

ethereum để tiết kiệm gas khi triển khai thực tế. Trước hết em sẽ đi từ các thuật toán ký và xác thực chữ ký trên đường cong elliptic. Việc này là cơ sở cần thiết cho tính đúng đắn thuật toán ecrecover.

Tham số	Ý nghĩa
CURVE	Trường hữu hạn đường cong sử dụng
G	Phần tử sinh của đường cong
n	Bậc của đường cong
sk	Khóa bí mật
pk	Khóa công khai $sk \times G$
m	message
L_n	Độ dài bit của bậc n
$Hash$	Thuật toán hash lấy L_n bit

Bảng B.2: Bảng tham số ECDSA

Để cho đơn giản, trong bảng tham số bảng B.2 ta giả sử n là một số nguyên tố. Trong thực tế ta có thể đạt được điều này nhờ việc xét nhóm con của nhóm ban đầu để thu được một nhóm cyclic.

ECDSA Signature Generation ([8])

Để ký thông điệp m, với các tham số trong bảng B.2.

1. Sinh số ngẫu nhiên $k \in [1, n-1]$.

   Trong ECDSA tất định như ethereum thì k sẽ được tính từ sk và h ([9]).

2. Tính $(x_1, y_1) = k \times G$ và $r = x_1 \mod n$. Nếu $r = 0$ quay lại bước 1.
3. Tính $k^{-1} \mod n$
4. Tính $e = Hash(m)$
5. Tính toán bằng chứng chữ ký $s = k^{-1} \times (e + r \times sk) \mod n$.
6. Signature(r, s).

ECDSA Verify Signature ([8])

Để xác thực chữ ký (r, s) lên thông điệp m, với các tham số trong bảng B.2.

1. $e = Hash(m)$.
2. Tính: $w = s^{-1} \mod n$.
3. Tính $u_1 = e \times w \mod n$ và $u_2 = r \times w \mod n$.
4. Tính $X = (x_1, y_1) = u_1 \times G + u_2 \times pk$. Nếu $X = \mathcal{O}$ từ chối chữ ký. Ngược lại $v = x_1$.
5. Chấp nhận chữ ký khi và chỉ khi $v = r$.

Nếu chữ ký (r, s) lên thông điệp m là hợp lệ thì với $s = k^{-1}(e + sk \times r)$

$$\begin{aligned} k &\equiv s^{-1}(e + sk \times r) \\ &\equiv s^{-1} \times e + s^{-1} \times r \times sk \\ &\equiv w \times e + w \times r \times sk \\ &\equiv u_1 + u_2 \times sk \mod n \end{aligned}$$

Do đó $u_1 \times G + u_2 \times pk = (u_1 + u_2 \times sk) \times G$ và $v = r$.

ECDSA Public Key Recovery From Signature Với thông điệp m, chữ ký (r, s) tính khóa công khai pk.

1. Kiểm tra $(r, s) \in [1, n-1]$. Nếu không, chữ ký không hợp lệ.
2. Tính $R = (x_1, y_1)$ với $x_1 = r$, y_1 là giá trị thỏa mãn đường cong elliptic.
3. $e = hash(m)$.
4. Tính $u_1 = -e \times r^{-1} \mod n$ và $u_2 = s \times r^{-1} \mod n$.
5. Tính $X = u_1 \times G + u_2 \times R$.
6. Trả về X

Ta sẽ chứng minh tính đúng đắn của thuật toán bắt đầu từ bước tính $X = u_1 \times G + u_2 \times R$.

$$\begin{aligned} X &= u_1 \times G + u_2 \times R \\ &= u_1 \times G + u_2 \times k \times G \\ &= (u_1 + u_2 \times k) \times G \\ &= (-e \times r^{-1} + s \times k \times r^{-1}) \times G \\ &= (-e \times r^{-1} + k^{-1}(e + sk \times r) \times k \times r^{-1}) \times G \\ &= (-e \times r^{-1} + (e \times r^{-1} + sk)) \times G \\ &= sk \times G \\ &= pk \end{aligned}$$

Cụ thể trong ethereum, hàm ecrecover sẽ trả về kết quả là address tương ứng với khóa công khai trên đường cong secp256k1.

Hàm ecrecover là một trong những precompiled của ethereum ([6]). Điều này làm tiết kiệm chi phí gas đáng kể so với cách triển khai hàm ecrecover trên một

hợp đồng thông minh khác, bất kế là cùng thuật toán với cài đặt của ethereum.

Tiếp theo em xin trình bày về kỹ thuật giả chữ ký. Kỹ thuật này sẽ tận dụng hàm ecrecover để tính toán tổ hợp tuyến tính hai điểm trên đường cong mà trong đó có một điểm là phần tử sinh của đường cong.

Cụ thể tại bước thứ năm của thuật toán **ECDSA Public Key Recovery From Signature** ta có được

$$X = u_1 \times G + u_2 \times R$$

Mục tiêu của ta là tính tổ hợp tuyến tính $u = s \times G + c \times pk$ như bước thứ nhất của thuật toán **Verifying**. Từ thuật toán precompiled ecrecover$(hash, v, r, s)$ ([6]) với $hash$ là kết quả băm của thông điệp m. Tham số v giúp chỉ rõ điểm $R = (x_1, y_1), x_1 = r$ nằm trên mặt phẳng tọa độ. r, s như trên thuật toán đã trình bày. Ta đưa ra giả chữ ký:

1. pseudoHash $= -s \times pk[0]$
2. $r = pk[0]$
3. $v = (pk[1]\%2 == 0)?27 : 28$
4. pseudoSignature $= c \times pk[0]$
5. ecrecover(pseudoHash, $v, r,$ pseudoSignature) $= c \times pk + s \times G$

Thật vậy,

$$\begin{aligned} u &= u_1 \times G + u_2 \times R \\ &= u_1 \times G + u_2 \times pk \\ &= s \times pk[0] \times pk[0]^{-1} + c \times pk[0] \times pk[0]^{-1} \times pk \\ &= s \times G + c \times pk \end{aligned}$$

Tương tự kỹ thuật giả chữ ký tính tổ hợp tuyến tính giữa hai điểm trên đường cong. Ta có thể tận dụng kỹ thuật này để xác thực tích vô hướng với một điểm bằng cách đặt tham số s bằng 0.

$$\text{scalar} \times \text{multiplicand} = \text{ecrecover}(0, v, \text{multiplicand}[0], \text{scalar} \times \text{multiplicand}[0])$$

Với tham số v giúp chỉ rõ điểm $R = (x_1, y_1), x_1 = r$ nằm trên mặt phẳng tọa độ.

B.4 Bổ đề

Bổ đề B.4.1. *Cho $(G, \cdot)$ là một nhóm giao hoán, $x \in G$, $a, b \in \mathbb{Z}$ nguyên tố cùng nhau. Nếu $x^a = x^b = 1$ thì $x = 1$.*

Ta sẽ chứng minh bổ đề bằng bổ đề của Bezout. Tồn tại $k, l \in \mathbb{Z}$ sao cho $ak+bl = 1$. Ta có:

$$x = x^{ak+bl} = x^{ak} \cdot x^{bl} = 1^k \cdot 1^l = 1$$

Bổ đề B.4.2. *Mọi nhóm giao hoán có bậc là số nguyên tố thì là nhóm cyclic.*

Gọi P là nhóm giao hoán bất kỳ có bậc là số nguyên tố p. Với mọi $x \in P$ và $x \neq 1$ ta xét

$$S = \{1, x, x^2, ..., x^{p-1}\} \subseteq P$$

Không mất tính tổng quát, giả sử rằng có hai phần tử $x^u = x^v$ với $u < v$. Do đó $x^{u-v} = 1$ hơn nữa $1 \leq u - v \leq p - 1$, $u - v$ và p nguyên tố cùng nhau. Theo bổ đề B.4.1 $x^{u-v} = x^p = 1$ ngụ ý rằng $x = 1$. Mâu thuẫn. Do vậy mọi phần tử của S là khác nhau. Mặt khác $|S| = P$ do đó $S = P$ và P là nhóm cyclic.

Bổ đề B.4.3. *Cho p là số nguyên tố lẻ. Ta sẽ có $\frac{p-1}{2}$ phần dư bậc hai và $\frac{p-1}{2}$ phần tử không phải là phần dư bậc hai trong khoảng từ $\{1, ..., p-1\}$.*

Ta có k và $-k = p - k$ có chung bình phương khi mod p. Có nghĩa là 1 và $p - 1$ có chung bình phương, 2 và $p-2$ có chung bình phương và $\frac{p-1}{2}$ và $\frac{p-1}{2}+1$ có chung bình phương khi mod p.

Do đó, ta sẽ có số bình phương khác nhau là $\frac{p-1}{2}$ - số các số là phần dư bậc hai và $\frac{p-1}{2}$ số khác không phải là phần dư bậc hai.