MỤC LỤC

PHẦN 1

BLOCKCHAIN CƠ BẢN

1 CÔNG NGHỆ SỔ CÁI

1. CƠ SỞ DỮ LIỆU TẬP TRUNG VÀ PHI TẬP TRUNG

Hệ thống thông tin

Công nghệ thông tin (*Information Technology*), là áp dụng Khoa học máy tính (*Computer Science*) để giải quyết các bài toán trong thực tế theo nghĩa xem các chương trình máy tính (*computer programs*) như hệ thống biến đổi thông tin hay dữ liệu đầu vào (*input*) thành thông tin hay dữ liệu đầu ra (*output*) dựa trên các thuật toán (algorithm) đã được lập trình (*programming*). Theo đó,

Thông tin được xem là thành phần trọng tâm của các hệ thống máy tính.

Một số chương trình máy tính, đặc biệt các hệ thống hỗ trợ quản lý, gọi chung là hệ thống thông tin (*information systems*), dữ liệu có thể được xem như '*trái tim*' của hệ thống trong khi chương trình là cài đặt mô phỏng các quy trình nghiệp vụ (*business process*). Trong các hệ thống thông tin này, dữ liệu được tổ chức và được tập hợp thành các cơ sở dữ liệu (*database*), là tập hợp các dữ liệu thể hiện (*instance*) của các thực thể (*entity*) cần quản lý.

Cơ sở dữ liệu

Về hình thức, Cơ Sở Dữ Liệu, viết tắt là **CSDL** hay **DB**, là một tập hợp các dữ liệu có liên quan với nhau, được tổ chức và được lưu trữ, truy cập từ hệ thống máy tính. Cấu trúc cho các thực thể cũng như quan hệ giữa các thực thể thường được mô hình hóa (*data modelling*) để có thể lưu trữ và xử lý hiệu quả nhất có thể. Mô hình thực thể-kết hợp (**ERM** – *Entity Relationship Model*)[1] là mô hình ý niệm (*concept model*) đang được sử dụng phổ biến cho quá trình thiết kế mô hình dữ liệu. Trong tài liệu này, để đơn giản, ta sẽ giới hạn mô hình chỉ gồm một thực thể trong một lược đồ quan hệ (*relation scheme*). Về mặt ý niệm, mỗi thực thể được quản lý qua các thuộc tính (*attributes*) có quan hệ nội tại với nhau trong cùng một thực thể. Về mặt vật lý, mỗi lược đồ quan hệ, gọi tắt là một quan hệ (**R** – *Relation*), được cài đặt như một bảng 2-chiều (*table*) và được lưu trữ trong một tập tin (*file*). Lưu trữ (*storing*) và truy xuất dữ liệu trong cơ sở dữ liệu, gọi chung là truy vấn (*querry*) được hỗ trợ bởi các hệ quản trị cơ sở dữ liệu (**DBMS** – *Data Base Management Systems*).

[1] ERM do Peter Chen đề xuất năm 1976 (The Entity-Relationship Model: Toward an Unified View of Data, MIT, 1976)

Hệ quản trị CSDL

Hệ quản trị cơ sở dữ liệu – **DBMS**, là (hệ thống) phần mềm được dùng để hỗ trợ việc lưu trữ (*storing*), truy xuất (*access*) và thực thi các truy vấn (*query*) trên CSDL. DBMS có thể xem như giao diện (*interface*) trung gian giữa người dùng cuối và cơ sở dữ liệu, cho phép người dùng tạo (*create*), đọc (*read*), cập nhật (*edit*) và xóa (*delete*) dữ liệu trong cơ sở dữ liệu. DBMS có thể được phân loại theo các tiêu chí khác nhau về mô hình dữ liệu, về phân phối cơ sở dữ liệu hoặc về số lượng người dùng. Các DBMS được sử dụng rộng rãi nhất là hệ quản trị quan hệ, phân tán, phân cấp, hướng đối tượng và mạng. DB cũng như DBMS có thể được lưu trữ và xử lý tập trung (*centralization*) hay không tập trung (*non-centralization*).

CSDL tập trung và phân tán

Cơ sở dữ liệu tập trung (**CDB** – *Centralized Data Base*) là cơ sở dữ liệu được định vị, lưu trữ và ***duy trì ở một vị trí duy nhất***. Vị trí này thường là máy tính trung tâm hoặc hệ thống cơ sở dữ liệu trung tâm. Trong hầu hết các trường hợp, cơ sở dữ liệu tập trung sẽ được khai thác bởi một cơ quan hay một tổ chức. Người dùng truy cập cơ sở dữ liệu tập trung thông qua mạng máy tính sau khi đã được cấp quyền truy cập vào hệ thống trung tâm, để tìm kiếm (*search*) hay duy trì (*maintain*) cơ sở dữ liệu đó.

Cơ sở dữ liệu phân tán (**DDB** – *Distributed Data Base*) là một loại cơ sở dữ liệu bao gồm nhiều cơ sở dữ liệu được kết nối với nhau và trải rộng trên các vị trí vật lý khác nhau. Dữ liệu được lưu trữ ở nhiều vị trí vật lý khác nhau, có thể được quản lý độc lập với các vị trí vật lý khác.

Cụm từ ***duy trì ở một vị trí duy nhất*** trong khái niệm **CDB** hàm ý rằng ngay cả CSDL được lưu trữ phân tán ở nhiều vị trí vật lý hay địa lý khác nhau thì việc ***duy trì, hay rộng hơn là giám sát, vẫn được thực hiện ở một vị trí hay một người có quyền cao nhất***. Theo đó, ta vẫn có thể xem một DDB là CDB khi việc duy trì tập trung vào 'một' người. Ngược lại, ta gọi là **CSDL phi-tập trung**.

CSDL phi-tập trung

Cơ sở dữ liệu phi-tập trung (*Decentralized Data Base*), chính xác phải gọi là dữ liệu phi tập trung (*decentralized data*) là dữ liệu được tạo (*create*), lưu trữ (*storing*) và truy xuất (*access*) một cách độc lập từ nhiều vị trí khác nhau và không bị giám sát bởi bất kỳ tổ chức hay cá nhân nào.

Như vậy, CSDL phi-tập trung là cả việc lưu trữ lẫn xử lý đều là phân tán. Điều này dẫn đến những thách thức trong việc duy trì một CSDL chung, khi dữ liệu được phát triển thành CSDL phi-tập trung – **Decentralized Database**.

CSDL tập trung hay CSDL phi-tập trung?

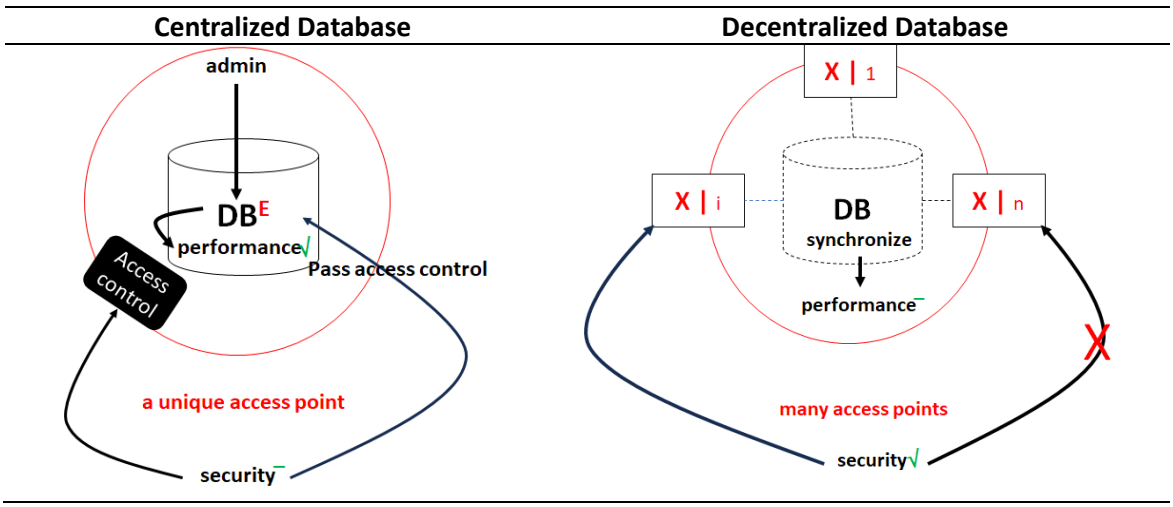
Cả CSDL tập trung lẫn phi-tập trung đều có những ưu thế và bất lợi riêng của chúng. Câu hỏi tự nhiên đặt ra là chọn CSDL nào cho một hệ thống thông tin: tập trung hay phi-tập trung?

Để trả lời, xét (xem hình vẽ minh họa) 2 tiêu chí quan trọng khi thiết kế một hệ thống thông tin: tiêu chí hiệu năng của truy vấn (*performant*) và tiêu chí bảo mật của hệ thống (*security*).

Trước hết, cần lưu ý CSDL được tạo lập cho mục đích chính là tìm kiếm. Các thao tác khác như thêm (insert/append/add), xóa (delete/remove), hay cập nhật (edit) CSDL có thể không đòi hỏi phải xử lý cấp bách.

Về tiêu chí hiệu năng: Do CSDL tập trung được lưu trữ và xử lý tại một điểm nên hiệu năng tìm kiếm cũng như duy trì CSDL là dễ dàng và hiệu quả. Truy vấn với CSDL tập trung đảm bảo được tiêu chí hiệu năng. Ngược lại, với CSDL phi-tập trung, do dữ liệu được lưu trữ tại nhiều điểm khác nhau, CSDL chung chỉ là ý niệm – là ảo, với các thể hiện vật chất – thực, nằm ở các vị trí (vật lý cũng như địa lý) khác nhau nên việc đồng bộ (*synchronize*) để duy trì cho CSDL được luôn nhất quán là thách thức cho tiêu chí hiệu năng đối với CSDL phi-tập trung.

Về tiêu chí bảo mật dữ liệu: Do tính tập trung, CSDL tập trung chỉ có một trọng điểm tấn công vì thế dữ liệu không thực sự an toàn-bảo mật, nhất là khi người có quyền cao nhất (*admin*) có toàn quyền và có thể làm sai lệch dữ liệu. Ngược lại, CSDL phi-tập trung, để tấn công, kẻ tấn công phải thay đổi được dữ liệu ở mọi điểm lưu trữ, mà sẽ bất khả thi nếu số điểm lưu trữ đủ nhiều.



Tóm lại, việc chọn CSDL tập trung hay phi-tập trung cần phải được xem xét cẩn thận và tùy thuộc ứng dụng của hệ CSDL.

CÔNG NGHỆ SỔ CÁI PHI TẬP TRUNG

Sổ cái

Thu nhỏ lại hoạt động của xã hội theo những giao dịch (*transaction*) như trao đổi, vay mượn, buôn bán,… trực tiếp giữa các cá nhân trong cộng đồng với nhau. Để đảm bảo có vay thì phải trả, có bán thì mới mua,… các hoạt động cần có người làm chứng, nói cách khác, mọi giao dịch cần minh bạch, không có gì khuất tất. Mô hình quản lý cộng đồng theo các giao dịch đơn giản này có thể được triển khai với khái niệm quen thuộc: sổ-cái (*ledger*). Theo đó, ta hình dung, ở nơi công cộng đặt một sổ cái mà mọi người đều có thể đọc và ghi lên đó công khai. Trong làng, ai vay của ai hay ai trả ai cái gì đều tự nguyện ghi lên sổ cái đó. Mọi tẩy xóa là không chấp nhận. Hình ảnh sổ cái công cộng (*public ledger*) với các giao dịch

công khai như hình dưới, cho ta ý niệm ban đầu về việc vận hành ổn định xã hội một cách minh bạch và công khai. Chẳng hạn, sổ cái ghi

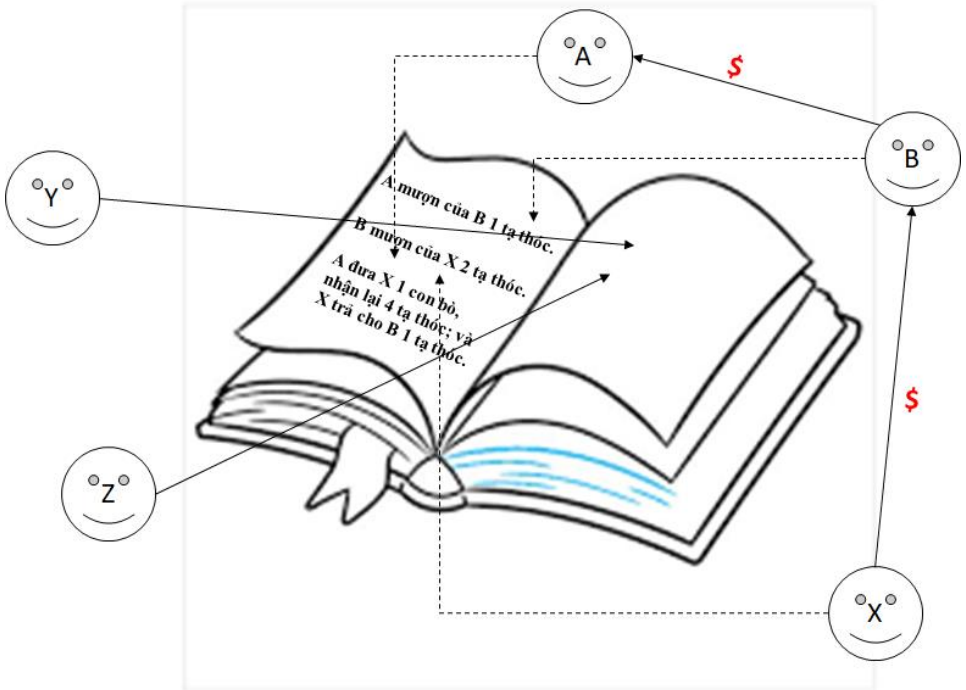
- "A mượn của B 1 tạ thóc."

- "B mượn của X 2 tạ thóc."

- "A đưa X 1 con bò, nhận lại 4 tạ thóc; và X trả cho B 1 tạ thóc."

- …

Bằng cách quy đổi mọi thứ về một thứ ta gọi là "tiền tệ", ký hiệu $, ta có mô hình kinh tế tiền tệ đơn giản như hình vẽ sau.



Sổ cái phi tập trung

Để đảm bảo sổ cái không bị phá hủy, vì một lý do nào đó như một "chúa chổm" nào đó có thể hủy sổ cái để từ đó không còn bằng chứng nào chứng minh người đó đang nợ như chúa Chổm. Vấn đề này có thể giải quyết bằng cách "phi tập trung hóa sổ cái – **Decentralized Ledger**", theo đó, sổ cái sẽ được giao cho n người giữ, mỗi người tự sao chép và cập nhật sổ cái mỗi khi có một giao dịch công khai xảy ra. Bằng cách đó, các cá nhân trao đổi tài sản với nhau chỉ phải công bố giao dịch của mình cho mọi người thấy. Những người đang giữ bản sao sổ cái sẽ xác minh (validate transaction) các giao dịch đó, và nếu hợp lệ, sẽ ghi giao dịch hợp lệ (validated transcaction) vào sổ cái mình đang giữ. Một sổ cái phi tập trung như thế đảm bảo 3 tính chất sau cho mọi giao dịch công khai:

– **Minh bạch**: mọi giao dịch được thực hiện công khai với các nhân chứng là những cá nhân đang giữ sổ cái.

– **Toàn vẹn**: không ai có thể sửa đổi nội dung của giao dịch đã được ghi xuống nếu không thể thông đồng được với mọi người giữ sổ cái.

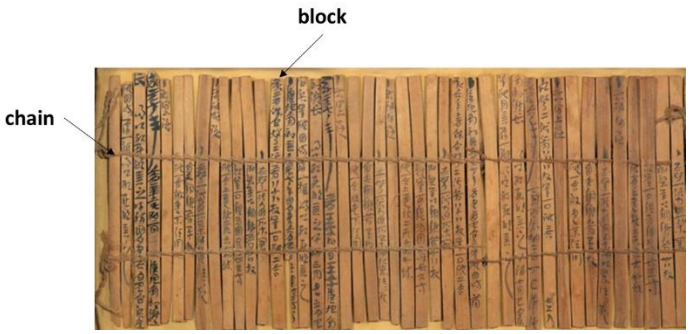
– **An toàn**: sổ cái không bị phá hủy nếu còn lại một vài người đang giữ sổ cái.

Công nghệ hiện thực hóa khái niệm sổ cái thỏa 3 tính chất trên gọi là **công-nghệ-sổ-cái-phi-tập-trung (DLT– Decentralized Ledger Technology)**.

BLOCKCHAIN – MỘT CÀI ĐẶT CHO SỔ CÁI PHI TẬP TRUNG

Blockchain và sổ cái

Blockchain viết liền, khác với khi viết "block chain" mà mang nghĩa "chain of blocks" được dịch là chuỗi khối, là khái niệm mới, được đặt ra để mô tả một dạng CSDL có tính chất bất biến một khi dữ liệu đã được ghi lên CSDL đó. Theo đó, blockchain có nghĩa gần với khái niệm sổ cái (ledger), nhưng không phải sổ cái thông thường mà là một sổ cái đặc biệt, các thông tin một khi đã được "khắc" vào sổ thì không thể bị hủy hoại, theo nghĩa dữ liệu chỉ có thể được 'khắc' thêm vào chứ không thể xóa hay thay đổi.



Trong tài liệu này, chúng tôi sẽ vẫn giữ nguyên thuật ngữ ***blockchain*** hoặc có thể sử dụng ***sổ cái phi tập trung*** hay **sổ cái blockchain** khi phải chuyển ngữ thuật ngữ này.

Hai thành phần chính làm nên blockchain gồm: **block** và **chain**.

(1) **Block**: Được hiểu là một ***trang*** của sổ cái mà nội dung là những giao dịch đã được ghi lên đó. Thuật ngữ ***block*** còn hàm ý tính *bền vững* của trang theo nghĩa không thể bị hư hại cũng như không thể sửa đổi những gì đã ghi trên đó.

(2) **Chain**: Được hiểu như một chất liệu 'đặc biệt' dùng ***xích*** (chain) trang sau vào liền với trang trước. Đặc biệt theo nghĩa nếu xích này bị hủy hoại thì toàn bộ sổ cái không thể dùng được nữa.

Công nghệ blockchain và công nghệ thông tin

Trong khoa học máy tính, khái niệm blockchain[2] hoàn toàn có thể được cài đặt bằng một cấu trúc dữ liệu đặc biệt kiểu danh sách liên kết (*list*) trong đó mỗi block là một trang (*page*) các bản ghi (*record*) lưu trữ thông tin của các giao dịch (*transaction*)[3], và một số thông tin quản lý khác.

Blockchain là một **list** (*danh sách*) đặc biệt. Đặc biệt ở chỗ với **list**, để liên kết thì block sau giữ địa chỉ (*address*) của block trước (trong nhiều ngôn ngữ liên kết này được thể hiện qua kiểu con trỏ (pointer)); nhưng trong **blockchain**, block sau giữ mã-chứng-thực (**MAC** – *Message Authentication Code*) của block trước, chính là trị băm mật mã (*cryptographic-hash value*) của thông điệp (message) là toàn bộ nội dung của block trước.

Như vậy, blockchain gần với khái niệm tập hợp (*set*) hơn là khái niệm mảng (*array*) hay danh sách liên kết (*list*) theo nghĩa phải duyệt tuần tự (sequence search) để tìm block hay Tx cần tìm. Điều này phù hợp với lưu trữ dạng tập tin (*file*) trên thiết bị phần cứng.

Blockchain và khoa học máy tính

Bên cạnh việc tổ chức sổ cái dạng blockchain, mỗi block là bản ghi (*record*) chứa tập hợp các giao dịch (*Tx*), và được lưu trên thiết bị phần cứng như một CSDL phi tập trung (*decentralized database*), theo nghĩa CSDL được lưu trữ trên nhiều máy một cách độc lập với nhau nhưng vẫn đảm bảo tính nhất quán của CSDL chung. Như vậy, để duy trì sổ cái cần (**i**) một hạ tầng kiến trúc mạng và các thuật toán truyền thông mạng hiệu quả và (**ii**) các thuật toán để đồng bộ dữ liệu giữa các CSDL được lưu trữ trên các điểm lưu trữ khác nhau.

Hạ tầng mạng cho blockchain

Vấn đề truyền thông mạng được giải quyết và hiện thực trên kiến trúc mạng ngang hàng (**P2P network** – *peer-to-peer network*), ở đó các nút mạng liên kết vào trao đổi thông tin

[2] Trong tài liệu này, thuật ngữ gốc blockchain sẽ được sử dụng. Cũng vậy, thuật ngữ block và chain cũng sẽ được giữ nguyên thay vì chuyển ngữ. Trong một số trường hợp, chúng tôi sử dụng thuật ngữ sổ cái blockchain hay sổ cái phi tập trung.

[3] Trong tài liệu này, từ Tx sẽ được dùng để kí hiệu cho giao dịch – transaction.

trực tiếp, không qua trung gian. Các nút mạng được trang bị trình cài đặt các thuật toán để quét mạng liên tục và sẵn sàng cập nhật CSDL khi có một block mới xuất hiện.

Đồng bộ dữ liệu

Tính nhất quán của CSDL chung được đảm bảo qua các thuật toán, được biết với tên **thuật-giải-đồng-thuận** (*consensus algorithm*), được xây dựng dựa trên nguyên tắc đồng-thuận-số-đông, theo đó, block mới sẽ được thêm vào chain hiện đang dài nhất hiện tại, được đồng thuận là CSDL hiện hành.

Một cách tổng quát, blockchain là một CSDL phi-tập trung (decentralized database. Nghĩa là, cùng một CSDL được lưu trên nhiều máy chủ ở nhiều vị trí vật lý cũng như địa lý khác nhau.

CÔNG NGHỆ BLOCKCHAIN

Công nghệ blockchain (*blockchain technology*) là nền tảng (*platform*) cung cấp các cơ chế cũng như công cụ cho phép hình thành và duy trì một mạng blockchain (*blockchain network*).

Mạng blockchain

Mạng blockchain hình thành từ các nút (*node*) mạng hoạt động độc lập, giao tiếp với các nút khác theo giao thức mạng ngang hàng. Mỗi nút tham gia mạng blockchain với một định danh (*identification*) duy nhất và được xem là tài khoản của người sở hữu nút đó, vì thế, dưới góc nhìn tài chánh-ngân hàng, định danh này chính là danh khoản hay số tài khoản (*account number*) của người đó.

Có 2 loại nút trong mạng blockchain gọi là nút sử dụng (*user node*) dữ liệu và nút duy trì (*miner node*) dữ liệu của CSDL blockchain.

Nút sử dụng dữ liệu blockchain – user node

User node: Là bất kỳ thiết bị nào có thể kết nối mạng để thực hiện các giao dịch liên quan đến thông tin được lưu giữ trong CSDL blockchain. Để giao dịch các dữ liệu của riêng mình, user node phải có khóa sử dụng tài khoản, một dạng mã mật như mã nhận dạng cá nhân (*Pin Code – Personal Identification Number*) được dùng để sử dụng thẻ ATM của ngân hàng. Khóa sử dụng tài khoản và số tài khoản có một quan hệ toán học chặt chẽ. Thực chất chúng là cặp khóa gồm khóa cá nhân (*private key*) và khóa công khai (*public key*) của một hệ mã công khai hiện đại (advanced public key cryptosystem), và địa chỉ (*address*) được phái sinh từ khóa công khai đó. Nhiều mạng chuỗi khối còn cho phép sử dụng trực tiếp khóa công khai làm địa chỉ tài khoản.

Nút duy trì mạng blockchain – miner nodes

Miner node: Là những máy chủ (*server*), đủ mạnh để có thể lưu giữ toàn bộ CSDL blockchain, từ block đầu tiên đến block mới nhất và còn tiếp tục cập nhật thêm các block tiếp nữa. Do tính chất bất biến của dữ liệu lưu trữ trong mạng blockchain, các block chỉ được thêm vào CSDL chứ không thể xóa hay sửa đổi. Hơn nữa, chỉ các block hợp lệ (*validated block*) mới được gắn thêm (*chain*) vào blockchain. Các miner nodes có nhiệm vụ kiểm tra (*verify*) các giao dịch hợp lệ tập hợp trong block hợp lệ và tranh đua cùng các miner

node khác trong một trò chơi (*game*) được thiết kế riêng cho từng blockchain, để trở thành nút tạo ra block mới nhanh nhất. Block mới này sẽ được gắn vào blockchain và cuộc đua tạo block mới lại tiếp tục.

Miner node cũng có thể có thông tin riêng của nó. Nghĩa là miner node hàm ý cũng là user node khi tham gia các giao dịch trao đổi thông tin của nó với các nút mạng khác trong cùng một mạng blockchain.

Các công nghệ cốt lõi của blockchain

Như ta đã thấy, blockchain hay sổ cái phi tập trung thực chất là một CSDL phi-tập trung được lưu trữ trên nhiều nút mạng trong mạng blockchain. CSDL này chỉ được phép "gắn" thêm vào, mọi thao tác có thể làm thay đổi dữ liệu đã lưu như sửa đổi (*edit*) hay xóa (*delete*) đều không được phép. Nguyên do là khi một miner node thay đổi dữ liệu nó đã lưu, đồng nghĩa miner node đó tự loại mình ra khỏi mạng các nút duy trì sổ cái và chỉ có thể là nút sử dụng.

Dữ liệu trong CSDL blockchain hay sổ cái phải đảm bảo tính toàn vẹn (*integrity*).

Tính toàn vẹn của dữ liệu trong sổ cái phi tập trung được đảm bảo bởi công nghệ mật mã hiện đại bao gồm mã băm để chứng thực thông điệp và khóa sử dụng thông tin tài khoản.

Mọi dữ liệu được băm bằng một hàm băm mật mã (*cryptographic hash function*) và được ký (*sign*) bằng khóa cá nhân của người tạo ra dữ liệu đó.

CSDL blockchain chỉ cho phép xem (*view*) và thêm các block hay trang hợp lệ vào.

Một trang hợp lệ là block được tạo bởi miner node chiến thắng (*winner*) trong một trò chơi (*game*) được thiết kế riêng cho từng mạng blockchain.

Như vậy, ba lý thuyết cũng như công nghệ nền tảng làm trụ cột cho blockchain gồm: **lý thuyết trò chơi**, **mật mã hiện đại**, và **mạng ngang hàng**.

Lý thuyết trò chơi

Lý thuyết trò chơi là khoa học nghiên cứu các mô hình toán học về tương tác chiến lược giữa các tác nhân. Lý thuyết trò chơi có ứng dụng trong mọi lĩnh vực khoa học xã hội, cũng như logic, khoa học hệ thống và khoa học máy tính. Các khái niệm về lý thuyết trò chơi cũng được sử dụng rộng rãi trong kinh tế học. Lý thuyết trò chơi tiên tiến được áp dụng cho nhiều mối quan hệ hành vi và hiện tại được xem như một thuật ngữ chung cho khoa học về việc ra quyết định hợp lý ở người cũng như máy tính.

Lý thuyết trò chơi được phát triển rộng rãi vào những năm 1950 và đã được áp dụng cho các quá trình tiến hóa vào những năm 1970, mặc dù những khái niệm tương tự đã có từ những năm 1930. Lý thuyết trò chơi đã được thừa nhận rộng rãi như một công cụ quan trọng trong nhiều lĩnh vực.

Tài liệu này không nghiên cứu nhiều về lý thuyết trò chơi cũng như công nghệ mạng ngang hàng mà tập trung chủ yếu vào công nghệ mật mã hiện đại.

Mạng ngang hàng

Mạng ngang hàng (**P2P**) là mạng các máy tính trong đó hai hoặc nhiều máy tính có thể chia sẻ thông tin mà không yêu cầu phải có máy chủ hoặc phần mềm máy chủ riêng biệt.

Ở dạng đơn giản nhất, mạng **P2P** được tạo ra khi hai hay nhiều máy tính được kết nối và chia thông tin mà không cần thông qua một máy chủ riêng. Mạng **P2P** có thể là một kết nối đặc biệt — một số máy tính được kết nối qua Universal Serial Bus để truyền tập tin (*file*). Mạng **P2P** cũng có thể là một cơ sở hạ tầng cố định liên kết nhiều máy tính trong một văn phòng nhỏ bằng dây cáp, gọi là mạng cục bộ (**LAN** – Local Area Network). Hoặc mạng **P2P** có thể là mạng ở quy mô toàn cầu, trong đó các giao thức và các ứng dụng đặc biệt thiết lập mối quan hệ trực tiếp giữa những người dùng qua Internet.

Việc sử dụng mạng **P2P**, ban đầu trong kinh doanh diễn ra sau khi triển khai các máy tính cá nhân độc lập vào đầu những năm 1980. Các máy tính cá nhân lúc bấy giờ có ổ cứng độc lập và CPU tích hợp. Các thiết bị mạng thông minh cũng có các ứng dụng tích hợp, nghĩa là chúng có thể được triển khai trên máy tính để bàn mà không cần kết nối chúng với máy tính lớn.

Cũng như lý thuyết trò chơi như lưu ý ở trên, tài liệu này không tập trung nhiều vào công nghệ mạng **P2P** mà sử dụng các giao thức tổng quát của mạng **P2P** cho mục đích chia sẻ thông tin.

Mật mã hiện đại

Mật mã học là kỹ thuật bảo mật thông tin được truyền thông qua việc sử dụng mã mật để chỉ những người có quyền sử dụng thông tin mới có thể hiểu và xử lý nó. Do đó, công nghệ mật mã có thể ngăn chặn việc truy cập trái phép vào thông tin. **Crypt**o**graph**y được cấu thành từ tiền tố "*crypt*" [4]có nghĩa là "*ẩn*" và hậu tố "*graph*" có nghĩa là "*viết*" – viết ẩn nghĩa hay mật mã. Các kỹ thuật được sử dụng để bảo vệ thông tin được khai thác từ các khái niệm toán học và tập các phép tính theo thuật toán (*algorithm*) để chuyển đổi thông điệp (*message*) theo những cách khiến việc giải mã nó trở nên khó khăn. Các thuật toán này được sử dụng để tạo khóa mã hóa (*encryption key*), chữ ký số (*digital signature*), xác minh (*authentication*) để bảo vệ tính riêng tư của dữ liệu (*data privacy*), và để bảo vệ các giao dịch bí mật như giao dịch thẻ tín dụng và thẻ ghi nợ.

Quá trình chuyển đổi một văn bản rõ (*plain text*) thành văn bản mật mã (*cipher text*), là tạo ra văn bản sao cho chỉ người nhận văn bản hợp pháp mới có thể giải mã nó, được gọi là mã hóa (*encryption*). Quá trình chuyển đổi ngược lại từ văn bản mã (*cipher text*) thành văn bản rõ (*plain text*) được gọi là giải mã (*decryption*).

Các hệ mã hiện đại (*advanced cryptosystem*) có các đặc tính cốt lõi sau:

– Tính bảo mật: Thông tin chỉ có thể được truy cập bởi người hợp pháp và không người nào khác có thể truy cập thông tin mật này

[4] Crypt còn có nghĩa là mộ, nên có thể hiểu mật mã là "chôn" giấu xuống.

– Tính toàn vẹn: Thông tin không thể được sửa đổi trong quá trình lưu trữ hoặc luân chuyển giữa người gửi và người nhận hợp pháp mà không phát sinh thông tin bổ sung.

– Chống từ chối: Người tạo cũng như người gửi thông tin không thể phủ nhận thông tin của mình đã gửi.

– Xác thực: Danh tính của người gửi và người nhận có thể xác nhận, cũng như điểm đến hay nguồn gốc của thông tin được xác nhận.

Có ba dạng mật mã: **mã đối xứng** (*symmetric cryptosystem*), hàm băm mật mã hay gọi tắt là **hàm băm** (*hash function*), và **mã bất đối xứng** (*asymmetric cryptosystem*).

Mật mã khóa đối xứng: Là một hệ thống mã hóa trong đó người gửi và người nhận sử dụng một khóa chung duy nhất (*pre-share key/secrete key/key*) để mã hóa và giải mã thông điệp. Mã đối xứng được thực hiện nhanh và đơn giản nhưng vấn đề nằm ở chỗ làm sao thiết lập khóa chung một cách an toàn giữa người gửi và người nhận. Các hệ mật mã khóa đối xứng phổ biến nhất là hệ mã hóa dữ liệu (DES – *Data Encryption System*) và hệ mã hóa nâng cao (AES – Advanced Encryption System).

Hàm băm: Không sử dụng khóa trong thuật toán mật mã này. Trị băm có độ dài cố định và phụ thuộc hoàn toán vào văn bản khiến nội dung của văn bản không thể phục hồi (do tính nén của trị băm). Nhiều hệ điều hành cũng như nhiều hệ thống thông tin sử dụng hàm băm để mã hóa mật khẩu. Các hàm băm phổ biến là MD5 (*Message Digest 5*) thường được dùng bảo mật mật khẩu trong các hệ thống thông tin; và SHA-2 (*Secure Hash Algorithm 2*) hiện được các mạng blockchain sử dụng.

Mật mã khóa bất đối xứng: Trong hệ thống này, một cặp khóa với hai khóa khác nhau được sử dụng để mã hóa và giải mã thông tin. Khóa công khai (*public key*) của người nhận được người gửi sử dụng để mã hóa và khóa cá nhân (*private key*) của người nhận được người nhận sử dụng để giải mã. Khóa công khai và khóa cá nhân phải khác nhau và có quan hệ toán học chặt chẽ. Mặc dù khóa công khai có thể được công bố cho mọi người được biết thì chỉ người nhận, làn người có khóa cá nhân tương ứng mới có thể giải mã. Thuật toán mã hóa khóa bất đối xứng phổ biến nhất là thuật toán RSA. Hiện tại, hệ mã dựa trên đường cong elliptic (ECC – Elliptic Curve Cryptogsystem) đang được sử dụng trong các mạng blockchain do tính hiệu quả và an toàn của nó.

2 MÃ HÓA ỨNG DỤNG

MÃ HÓA THÔNG TIN

Mật mã học (*cryptology*) là ngành khoa học nghiên cứu các phương pháp liên quan đến bảo mật thông tin và truyền thông qua việc sử dụng các kỹ thuật biến đổi thông tin. Một cách tổng quát, mật mã học bao gồm (**i**) mã hóa (*cryptography*) nhằm nghiên cứu cách xây dựng các hệ mã và (**ii**) thám mã (*cryptoanalysis*) là nghiên cứu cách phân tích các hệ mã đã được công bố:

CRYPTOLOGY = CRYPTOGRAPHY + CRYPTOANALYSIS

Ví dụ mở đầu

Bằng cách sử dụng bảng ký hiệu chuẩn của Hoa Kỳ về trao đổi thông tin – **ASCII** (American Standard Code for Information Interchange), với chữ 'A' được biểu diễn bằng 1 byte (8 bit nhị phân) có dạng 01000001 tương ứng với số 65 trong hệ thập phân, 'B' là 66, …, 'Z' là 90. Và ta giả sử thêm dấu cách (khoảng trắng) có mã 64 (thực ra 64 là ký hiệu của '@'), thì chuỗi ký tự:

S = 'KHOA CONG NGHE THONG TIN' ứng với dãy số

M = 75727965$_{64}$67797871$_{64}$78717269$_{64}$8472797871$_{64}$847378.

Dòng số trên chỉ là cách mã hóa theo quy ước (*encode*) không phải là mật mã (*encrypt*) vì mọi người có thể lấy từng cặp số trong dãy trừ đi 64 sẽ được lại dòng chữ gốc (*decode*). Ta gọi chung cách này là số hoá.

Ví dụ 1: giả sử lấy từng cặp số **m** trong chuỗi số và biến đổi thành **c** theo công thức

$c = E_k(m) = 64 + (m - 64 + k)\%26$

Lấy bí mật **k** = 3, ta được dãy số

C = 78758268$_{67}$70828174$_{67}$81747572$_{67}$8775828174$_{67}$877681,

và tra ngược bảng mã ASCII, ta được chuỗi ký tự

W = NKRD FRQJ QJKH WKRQJ WLQ.

Ta thấy, chỉ những ai biết **k** = 3 mới đọc được ý nghĩa của chuỗi W.

Ví dụ 2: Giả sử với **k** = 4018975632 và ta lấy từng chữ số **m** trong dãy số M biến đổi thành số **c** theo công thức:

c = σ(m), với σ(0123456789) = k = 4018975632, nghĩa là σ(0) = 4, σ(1) = 0, … σ(8) = 3 và σ(9) = 2.

Từ dãy số M = 75727965 67797871 78717269 8472797871 847378,

ta được dãy số C = 67616257 56626360 63606852 3961626360 396863

Và sử dụng m = σ^{-1}(c), có thể dễ dàng lật ngược dãy số và có thể chuyển sang chuỗi ký tự đọc được.

Ví dụ 3: Tương tự, ta có thể dùng k = σ(0123456789) = 0185947263.

Ví dụ 4: Lấy kết quả của ví dụ 3, biểu diễn lại bằng cách gom 8 số lại thành nhóm và cộng các nhóm lại. Kết quả được bao nhiêu chia cho p = 7919, giữ lại phần dư, ta được

h = H(C) = (67616257 + 56626360 + 63606852 + 39616263 + 60396863) % 7919 = 6945.

Rõ ràng, từ h = 6945, rất khó để biết nó được tính từ C.

Định nghĩa hình thức

Các ví dụ trên cho ta ý niệm ban đầu về các hệ mật mã (*cryptosystem*), mà là một ánh xạ (*mapping*) biến đổi từng thành phần của chuỗi ký tự cần mã hóa sau, khi đã số hóa (*digitalize*), được gọi là thông điệp (*message*) hay bản rõ (*plain text*) thành các thành phần của dãy số mật, gọi là bản mã (*cipher text*) bằng cách kết hợp với một thông tin mật, gọi là khóa (*key*), để chỉ người có khóa tương ứng mới có thể hiểu được thông điệp gốc. Một cách hình thức, ta định nghĩa một hệ mã như sau:

Một hệ mã là một ánh xạ E_k "*khả nghịch*" đi từ không gian các thông điệp $\mathcal{M}$ vào không gian các bản mã $\mathcal{C}$ với k là một khóa trong không gian khóa K:

$E_k: \mathcal{M} \rightarrow \mathcal{C}$, sao cho

$\forall m \in \mathcal{M}, k \in K, c = E_k(m) \in C, E_{k'}^{-1} \equiv D_{k'}(c) = m.$

Hàm E_k(m) gọi là hàm mã hóa (encryption), mã hóa thông điệp m thành bản mã c, sử dụng khóa mã hóa k (*encryption key*). Và hàm $D_{k'}$(c) là hàm giải mã (decryption) mà giải mã bản mã c thành bản rõ m, sử dụng khóa giải mã k' (*decryption key*).

Tùy thuộc độ phức tạp của hàm mã hóa, hoặc tùy thuộc quan hệ giữa khóa mã hóa và khóa giải mã, hoặc có sử dụng khóa hay không mà ta có thể phân biệt các hệ mã khác nhau.

Phân loại các hệ mã

Bí mật hoàn toàn – Hàm mã hóa là ánh xạ đơn giản

Như trong ví dụ 1, cho dù có bí mật khóa k thì chỉ bằng cách thử (vét cạn) qua 26 trường hợp, bản rõ có thể được phục hồi từ bản mã được cho. Trong trường hợp này, phải bí mật cả thuật toán (hàm mã hóa) lẫn khóa mã hóa.

Công bố thuật toán và giữ bí mật thông tin khóa – k ≡ k'

Các ví dụ 2 và ví dụ 3 minh họa hệ mã thuộc loại này. Trong các hệ mã này, khóa bí mật là dãy số, mà là hoán vị của 10 chữ số (hệ thập phân), và hàm mã hóa chỉ là thay thế thông điệp m bằng hoán vị tương ứng σ(m) của nó. Trong ví dụ này, không gian phải vét cạn là 10! = 3628800 khả năng. Nếu hoán vị của 100 ký hiệu (hệ bách phân) thì khả năng vét cạn 100!, trường hợp này sẽ là bất khả thi. Hệ mã dạng này được gọi là mã đối xứng (*symmetric cryptosystem*) do khóa giải mã có thể được suy ra dễ dàng từ khóa mã hóa. Khóa mã hóa là bí mật chung giữa hai đối tác trong hệ thống, vì thế phải được giữ bí mật khóa, hệ mã có tính chất này còn được gọi là mã (khóa) bí mật (*secret key cryptosystem*). Rõ ràng khóa phải được quy ước trước giữa hai đối tác, ta cũng có thể gọi là hệ mã quy ước khóa trước (*pre-share key cryptosystem*).

Công bố thuật toán và công bố một phần thông tin khóa – k ≠ k'

k trong ví dụ 3 có tính chất đáng lưu ý là $y = \sigma(x) = x^3 \% 11$ và $\sigma^{-1}(y) = y^{11} \% 11 = x$. Như vậy, nếu ký hiệu e = k = 3 là khóa mã hóa thì ta có thể dùng d = k' = 7 để giải mã. Theo đó, mã hóa bằng một khóa, khóa mã hóa **e** (*encryption key*), và giải mã bằng khóa khác, khóa giải mã **d** (*decryption key*). Như vậy, sử dụng hệ mã này ta có thể công khai khóa mã hóa e, gọi là khóa công khai (*public key*), và giữ bí mật khóa giải mã d, gọi là khóa cá nhân (*private key*). Nên hệ mã dạng này được gọi là hệ mã (khóa) công khai (*public key cryptosystem*) hay còn có tên khác là hệ mã bất đối xứng (*asymmetric cryptosystem*) do tính khác biệt của các khóa mã và giải mã.

Không sử dụng khóa – k = ''

Ví dụ 4 minh họa cách hoạt động của hệ mã không sử dụng khóa. Hệ mã này đơn giản là biến đổi chuỗi thông điệp dài bất kỳ thành một chuỗi có chiều dài cố định. Hàm biến đổi E(.) này, giờ được ký hiệu là H(.), được gọi là hàm băm mật mã, và giá trị h = H(S) băm một chuỗi S được gọi là trị băm. Hàm này đơn giản chỉ là giấu thông tin (gốc) và thường được sử dụng cho các mục đích xác thực.

Các mạng blockchain sử dụng chính hai loại hệ mã: mã công khai và hàm băm, hệ mã bất đối xứng được sử dụng với mục đích bảo mật thông tin và không phổ biến.

Nguyên lý thiết kế các hệ mã

Về hình thức, ta thấy, để xây dựng một hệ mã, đơn giản chỉ là xác định 3 không gian: **không gian các thông điệp** ($\mathcal{M}$), **không gian các bản mã** (C), **không gian khóa** (K), và thiết kế thuật toán cho hàm khả nghịch E. Tuy nhiên, do yêu cầu bảo mật, các hệ mã phải được thiết kế để đảm bảo an toàn, bảo mật và hiệu quả khi dùng. Trong thực tế, các hệ mã thường được thiết kế theo nguyên lý Kerchkoff.

Nguyên lý Kerchkoff

– Hệ mã phải được chứng minh không thể bị thám mã về mặt toán học (*analyzing*), hơn nữa, phải không thể bị phá vỡ trong thực tế (*hacking*).

– Hệ mã nếu có bị kẻ xâm nhập khống chế cũng sẽ không bị bất kỳ sự xâm phạm nào vào hệ thống, và không gây bất kỳ sự bất tiện nào cho người dùng.

– Khóa mã hóa phải dễ trao đổi, dễ nhớ và có thể thay đổi.

– Bản mã có thể được truyền trên các kênh không an toàn.

– Thiết bị mã hóa và các văn bản phải có tính động và dễ sử dụng.

– Cuối cùng, hệ mã (chương trình) cần dễ sử dụng, không đòi hỏi phải có kiến thức và các quy tắc phức tạp phải tuân thủ.

Một cách đơn giản là hệ mã phải có tính bảo mật.

Về tính bảo mật

Một hệ mã, trước hết phải bảo mật và kháng được mọi tấn công hệ thống (*hacking*) cũng như phân tích thuật toán (*analyzing*).

Tấn công hệ mã

Các cuộc tấn công thường được phân loại theo hành động được kẻ tấn công thực hiện. Theo đó, tấn công có thể là thụ động hoặc chủ động

Tấn công thụ động: Mục tiêu chính là giành được quyền truy cập (trái phép) vào thông tin.

Tấn công tích cực: Một cuộc tấn công chủ động liên quan đến việc chủ động thay đổi thông tin theo một cách nào đó qua việc tiến hành một số quy trình trên thông tin.

Các giả định kẻ tấn công có thể có

Môi trường xung quanh hệ thống mật mã: Kẻ tấn công có thể thâm nhập vào môi trường, nơi các thiết bị hay hệ mã đang hoạt động.

Chi tiết về sơ đồ mã hóa: Kẻ tấn công có được chi tiết về các lược đồ, các giao thức mật mã.

Có của bản mã: Kẻ tấn công được cho là có thể thu thập được không giới hạn các bản mã.

Có *sẵn bản rõ và bản mã tương ứng*: Kẻ tấn công được cho là có thể thu thập được tập (vô hạn) các cặp bản rõ và bản mã.

Từ các giả định trên, các kỹ thuật tấn công có thể được phân loại như sau.

Phân loại tấn công

Tấn công chỉ dựa vào bản mã: **COA** – **Cipher text Only Attack**

Tấn công chỉ dựa vào bản mã (COA– Cipher text Only Attack) là mô hình thám mã khi kẻ tấn công chỉ có tập bản mã. Mặc dù chỉ có các bản mã nhưng trong thực tế, kẻ tấn công có thể có một số tri thức về bản rõ. Chẳng hạn, kẻ tấn công có thể biết ngôn ngữ của bản rõ (tiếng Việt, tiếng Anh, …) hoặc/và biết được phân phối các ký tự trong bản rõ. Cũng vậy, dữ liệu cũng như các thông điệp trong các giao thức, trong nhiều hệ thống, thường là một phần của bản rõ và có thể đoán hoặc biết trước.

Tấn công thành công khi phục hồi được bản rõ tương ứng (tấn công một phần), hoặc phục hồi đươc khóa mã hóa (tấn công toàn phần). Thậm chí, thu thập được bất kỳ thông tin nào về bản rõ ngoài những gì kẻ tấn công đã biết vẫn được coi là thành công. Chẳng hạn, nhiều giao thức gửi bản mã liên tục nhằm duy trì trạng thái bảo mật đường truyền, việc phân biệt các tin nhắn thực với các tin nhắn giả có thể xem là thành công, thậm chí dự đoán được về

sự tồn tại của một số thông điệp thực sự cũng sẽ tạo điều kiện thuận lợi cho việc phân tích lưu lượng truy cập.

Tấn công biết bản rõ – **KPA**

Tấn công biết bản rõ (KPA– Known Plaintext Attack) là mô hình thám mã trong đó kẻ tấn công thu thập được tập các bản rõ và bản mã tương ứng. Tập dữ liệu này có thể được sử dụng để phục hồi một số thông tin bí mật như khóa cá nhân bí mật và từ điển mã (code book).

Tấn công chọn bản rõ – **CPA**

Tấn công chọn bản rõ (CPA– Chosen Plaintext Attack) là mô hình thám mã, giả định rằng kẻ tấn công có thể thu thập được các bản mã của các bản gốc hắn muốn. Mục tiêu của tấn công này là lấy thông tin nhằm làm giảm tính bảo mật của sơ đồ mã hóa.

Tấn công từ điển – **Dictionary Attack**

Tấn công từ điển dựa vào việc thử tất cả các chuỗi trong danh sách được sắp xếp trước. Tấn công từ điển có thể sử dụng các danh sách có sẵn trên Internet chứa hàng trăm triệu mật khẩu được thu thập từ các vụ vi phạm dữ liệu đã xảy ra. Ngoài ra có thể sử dụng phần mềm bẻ khóa áp dụng vào các danh sách đó để tạo ra các biến thể khác, như thay thế các chữ số bằng các chữ cái. Tấn công từ điển thường thành công vì phần lớn người dùng phổ thông có xu hướng chọn mật khẩu ngắn và là những từ dễ nhớ hoặc mật khẩu thông dụng. Tấn công từ điển cũng có thể thành công vì nhiều kỹ thuật tạo mật khẩu thông dụng đều nằm trong danh sách có sẵn. Kết hợp với việc tạo mẫu của phần mềm bẻ khóa, khả năng thành công càng cao.

Tấn công vét cạn **– Brute Force Attack**

Tấn công vét cạn là tấn công mà kẻ tấn công kiểm tra một cách có hệ thống tất cả mật khẩu và nhóm các mật khẩu có thể có cho đến khi tìm được mật khẩu. Ngoài ra, kẻ tấn công có thể đoán khóa bằng cách sử dụng hàm phái sinh khóa.

Trên lý thuyết, tấn công vét cạn là tấn công thám mã có thể được áp dụng cho bất kỳ bản mã nào. Tấn công như vậy được sử dụng khi không thể tận dụng các điểm yếu khác trong hệ thống mã hóa (nếu có).

Khi đoán mật khẩu, tấn công vét cạn hiệu quả khi kiểm tra tất cả mật khẩu ngắn, đối với mật khẩu dài, các phương pháp khác như tấn công từ điển hiệu quả hơn. Mật khẩu, nhóm mật khẩu và khóa dài khó bị bẻ khóa hơn so với những mật khẩu ngắn.

Tấn công vét cạn sẽ không hiệu quả với cách làm xáo trộn dữ liệu mã hóa khiến kẻ tấn công khó nhận ra khi nào mã đã bị bẻ khóa hoặc chí ít cũng khiến kẻ tấn công phải làm nhiều việc hơn để kiểm tra từng lần đoán. Tấn công vét cạn cũng được dùng để đánh giá độ an toàn của một hệ thống.

Tấn công vét cạn thực chất là áp dụng của tìm kiếm vét cạn. Kỹ thuật chung là liệt kê tất cả các ứng cử viên và kiểm tra từng ứng cử viên.

Tấn công sinh nhật – **Birth Attack**

Tấn công sinh nhật là một kiểu tấn công bằng cách khai thác tính chất toán học của bài toán sinh nhật trong lý thuyết xác suất. Tấn công sinh nhật dựa trên nguyên lý chuồng bồ câu,

theo đó, nếu số chim câu nhiều hơn số chuồng thì chắc chắn có chuồng có nhiều hơn 1 con chim câu, nghĩa là có đụng độ. Tấn công sinh nhật được dùng để tìm ra xung đột trong hàm băm. Có giả thuyết là máy tính lượng tử có thể thực hiện các cuộc tấn công sinh nhật hiệu quả. Tuy nhiên, giải thuyết này còn nhiều tranh cãi.

Tấn công qua trung gian – **MIMA**

Tấn công qua trung gian (MIM– Man in the Middle Attack) là tấn công mạng trong đó kẻ tấn công chủ động chuyển tiếp và có thể thay đổi thông tin liên lạc giữa các đối tác đang liên lạc mà người trong cuộc không nhận ra có kẻ đứng giữa họ. Kẻ tấn công phải có khả năng chặn tất cả các tin nhắn trao đổi giữa hai nạn nhân và tiêm những tin nhắn có chủ đích của hắn vào. Việc này trong nhiều trường hợp có thể thực hiện dễ dàng. Chẳng hạn, trong vùng sóng wifi không mã hóa, kẻ tấn công có thể phá vỡ tiến trình xác thực lẫn nhau.

Tấn công kênh ph – **SCA**

Tấn công kênh phụ (SCA – Side Chanel Attack) là tấn công trên thông tin bổ sung có thể được thu thập từ cách giao thức hoặc thuật toán máy tính được triển khai, chứ không phải do sai sót trong thiết kế giao thức hoặc thuật toán hoặc do những sơ suất nhỏ, nhưng có khả năng gây thiệt hại nghiêm trọng trong quá trình triển khai. Thông tin về thời gian, mức tiêu thụ điện năng, rò rỉ điện từ và âm thanh là những ví dụ về thông tin bổ sung có thể bị khai thác làm tiền đề cho các cuộc tấn công kênh phụ. Tấn công kênh phụ thường yêu cầu kiến thức kỹ thuật cao về hoạt động của hệ thống.

Tấn công định thời – **Timming Attack**

Tấn công định thời là một dạng của tấn công kênh phụ trong đó kẻ tấn công thâm nhập hệ thống mật mã và phân tích thời gian thực hiện các thuật toán mật mã. Mọi thao tác logic trong máy tính đều cần có thời gian thực thi và thời gian này dài ngắn khác nhau phụ thuộc đầu vào; nếu đo chính xác được thời gian các thao tác, kẻ tấn công có thể suy ngược lại đầu vào. Việc phục hồi thông tin bí mật dựa trên thông tin thời gian có thể dễ hơn nhiều việc sử dụng phương pháp thám mã đã biết ở trên. Đôi khi thông tin về thời gian cũng được kết hợp với thám mã để tăng tốc độ phục hồi thông tin mật.

Thông tin có thể bị rò rỉ qua việc đo thời gian phản hồi các truy vấn cụ thể. Tấn công định thời có thể được áp dụng cho bất kỳ thuật toán nào có sự thay đổi thời gian phụ thuộc vào dữ liệu. Việc loại bỏ sự phụ thuộc vào thời gian là khó trong một số thuật toán sử dụng các chỉ thị cấp thấp thường có thời gian thực hiện khác nhau.

Tấn công theo thời gian thường không được xem xét trong giai đoạn thiết kế hệ mã vì chúng phụ thuộc quá nhiều vào việc triển khai và có thể được phát sinh một cách vô ý khi tối ưu hóa trình biên dịch. Vì thế, cần lưu ý tấn công khai thác thời gian thực thi ngay trong quá trình thiết kế thuật toán hay giao thức bảo mật.

Tấn công phân tích năng lượng – **Power Analysis Attacks**

Phân tích năng lượng là một hình thức tấn công kênh phụ trong đó kẻ tấn công nghiên cứu mức tiêu thụ năng lượng của thiết bị phần cứng mật mã. Tấn công này dựa vào các đặc tính

vật lý cơ bản của thiết bị. Bằng cách đo dòng điện, có thể biết được một lượng nhỏ thông tin về dữ liệu đang được xử lý.

Phân tích năng lượng trong trường hợp đơn giản (SPA – Simple Power Analysis) liên quan đến việc phân tích số liệu thống kê về năng lượng hoặc biểu đồ hoạt động của dòng điện theo thời gian. Hiện đại hơn là phân tích năng lượng vi phân (DPA – Diferential Power Analysis), một hình thức phân tích năng lượng hiện đại, cho phép kẻ tấn công tính toán các giá trị trung gian trong quá trình tính toán mật mã thông qua việc phân tích thống kê dữ liệu được thu thập từ nhiều hoạt động mật mã.

Tấn công phân tích lỗi – **Fault analysis Attacks**

Phân tích lỗi vi sai (DFA – Differential Fault Analysis) là một dạng tấn công kênh phụ vào các hê mã, cụ thể là thám mã. Nguyên tắc là tạo ra các lỗi làm cho các hoạt động mã hóa có thể tiết lộ trạng thái bên trong của chúng.

HỆ MÃ KHÓA CÔNG KHAI

Mã khóa công khai

Hệ mã khóa công khai (*public key cryptosystem*), hay mã bất đối xứng (*asymmetric cryptosystem*), là các hệ mã sử dụng cặp khóa có quan hệ toán học với nhau, R(e, d). Mỗi cặp khóa gồm một khóa công khai (*public key*) **e**, và một khóa cá nhân (*private key*) **d**. Các cặp khóa được tạo bằng thuật toán mật mã dựa trên một bài toán khó. Bài toán khó được sử dụng để tạo ra hàm một chiều. Tính bảo mật của mật mã khóa công khai phụ thuộc vào việc giữ bí mật khóa cá nhân cũng như bài toán khó cụ thể; khóa công khai có thể được công bố mà không ảnh hưởng đến bảo mật của hệ mã.

Trong hệ mã khóa công khai, bất kỳ ai có khóa công khai e đều có thể mã hóa thông điệp để, tạo ra bản mã, nhưng chỉ những người biết khóa cá nhân d tương ứng mới có thể giải mã để phục hồi lại bản rõ.

Chữ ký số

Trong hệ thống chữ ký số, người gửi sử dụng khóa cá nhân **d** cùng với thông điệp để tạo ra chữ ký cho thông điệp đó. Bất kỳ ai có khóa công khai **e** tương ứng quan hệ R(e, d), đều có thể xác minh xem chữ ký có khớp với thông điệp hay không. Người giả mạo không biết khóa cá nhân sẽ không thể tìm ra bất kỳ cặp thông điệp/chữ ký nào vượt qua quá trình xác minh bằng khóa công khai.

Ứng dụng của mã công khai

– Ứng dụng rõ ràng nhất của hệ mã công khai là mã hóa thông tin liên lạc để cung cấp tính bảo mật của một tin nhắn mà người gửi mã hóa bằng khóa công khai của người nhận, chỉ có thể được giải mã bằng khóa cá nhân tương ứng của người nhận.

– Một ứng dụng khác của mã công khai là chữ ký số. Lược đồ chữ ký số có thể được sử dụng để xác thực người gửi. Hệ thống chống từ chối sử dụng chữ ký số để đảm quyền tác giả của một tài liệu hoặc thông tin liên lạc.

– Các ứng dụng khác được xây dựng trên nền tảng mã công khai có thể kể đến: tiền kỹ thuật số (cryptocurrency), trao đổi/thiết lập khóa cho các hệ mã đối xứng, chứng thực, và các dịch vụ đánh dấu thời gian và các giao thức chống từ chối.

Điểm yếu của các hệ mã công khai

Như mọi hệ thống liên quan đến bảo mật, điều quan trọng là xác định các điểm yếu tiềm ẩn. Ngoài việc lựa chọn thuật toán khóa bất đối xứng yếu hoặc độ dài khóa quá ngắn, rủi ro bảo mật chính là khóa cá nhân bị lộ. Các điểm yếu này tuỳ thuộc vào các hệ mã được thiết kế như thế nào.

Ngoài ra, với sự ra đời của điện toán lượng tử, nhiều thuật toán khóa bất đối xứng được coi là dễ bị tấn công và các kế hoạch kháng lượng tử mới đang được phát triển để khắc phục vấn đề này.

HÀM BĂM MẬT MÃ

Hàm băm

Hàm băm mật mã (**CHF** – Cryptographic Hash Function) là một thuật toán băm, ánh xạ của chuỗi nhị phân với số bit bất kỳ thành chuỗi nhị phân có kích thước cố định là n bit và có các tính chất:

– Xác suất tìm ra một chuỗi từ một trị băm n-bit cụ thể là 2^{-n} (đối với bất kỳ hàm băm tốt nào). Do đó giá trị băm có thể được sử dụng làm đại diện cho thông điệp;

– Kháng đụng độ loại một: Việc tìm chuỗi đầu vào khớp với giá trị băm cho trước (tiền ảnh) là bất khả thi.

– Kháng đụng độ loại hai: Việc tìm ra hai chuỗi có cùng trị băm là không thể

Hàm băm mật mã có nhiều ứng dụng trong bảo mật thông tin, đặc biệt là trong chữ ký số (*digital signature*), mã xác thực thông điệp (MAC – *Message Authentication Code*) và các hình thức xác thực khác. Chúng cũng có thể được sử dụng để lập chỉ mục dữ liệu trong bảng băm, để lấy dấu vân tay, để phát hiện dữ liệu trùng lặp hoặc nhận dạng tính duy nhất các tập tin và mã kiểm tổng (*checksum*) để phát hiện lỗi dữ liệu. Trị băm mật mã đôi khi được gọi là dấu vân (*fingerprint*), mã kiểm tra hoặc đơn giản là trị băm, cho thấy các ứng dụng có thể có của hàm băm.

Khác với hàm băm mật mã, hàm băm phi mật mã được sử dụng trong các bảng băm và để phát hiện các lỗi vô tình, việc xây dựng chúng thường không có khả năng chống lại các tấn công có chủ ý. Ví dụ: tấn công từ chối dịch vụ vào các bảng băm có thể xảy ra nếu các xung đột được tìm thấy dễ dàng.

Tính chất

Hàm băm mật mã thường được thiết kế để nhận đầu vào là một chuỗi bit có độ dài thay đổi và tạo ra giá trị băm có độ dài cố định.

Hàm băm mật mã phải có khả năng chống lại tất cả các kiểu tấn công phân tích mật mã đã biết. Trong mật mã lý thuyết, mức độ bảo mật của hàm băm mật mã đã được xác định qua các thuộc tính sau:

Kháng tiền ảnh loại một

Cho trước trị băm h, sẽ khó tìm được bất kỳ thông điệp m nào sao cho hash(m) = h. Khái niệm này có liên quan đến hàm một chiều. Các hàm không có tính chất này có thể bị tấn công tiền ảnh loại một.

Kháng đụng độ yếu

Với đầu vào m_1, khó tìm được đầu vào $m_2 \neq m_1$ sao cho hash(m_1) = hash(m_2). Tính chất này còn được gọi là khả năng kháng đụng độ yếu. Các hàm không có tính chất này dễ bị tấn công tiền ảnh loại hai.

Kháng đụng độ mạnh

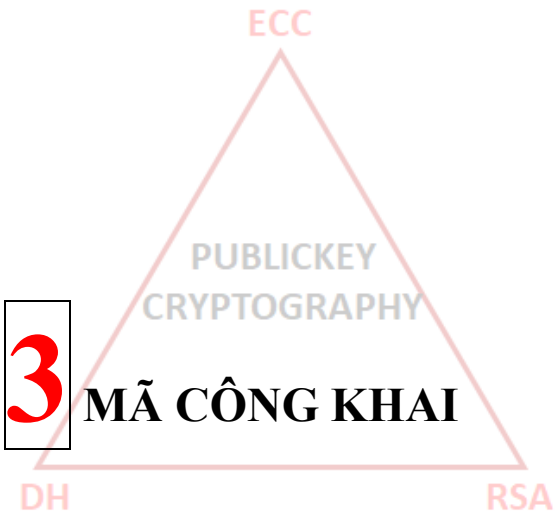
Rất khó để tìm được hai thông điệp khác nhau m_1 và m_2 sao cho hash(m_1) = hash(m_2). Cặp thông điệp như vậy được gọi là đụng độ. Tính chất này còn được gọi là khả năng kháng đụng độ mạnh. Trị băm cần dài ít nhất gấp đôi giá trị cần thiết cho khả năng kháng tiền ảnh; nếu không thì đụng độ có thể được tìm ra bằng tấn công sinh nhật.

Khả năng kháng đụng độ ngụ ý khả năng kháng tiền ảnh hai nhưng không bao hàm khả năng kháng tiền ảnh.

Các tính chất này có nghĩa là kẻ xấu không thể thay thế hoặc sửa dữ liệu đầu vào mà không làm thay đổi trị băm, gọi là bản tóm tắt của văn bản đầu vào. Vì vậy, nếu hai chuỗi có cùng một bản tóm tắt, ta có thể kết luận chúng giống hệt nhau. Khả năng kháng tiền ảnh loại hai ngăn kẻ tấn công tạo một văn bản có cùng trị băm với văn bản mà kẻ tấn công không biết. Khả năng kháng đụng độ mạnh ngăn kẻ tấn công tạo hai văn bản khác nhau có cùng trị băm.

Một hàm đáp ứng các tiêu chí trên vẫn có thể có những tính chất không mong muốn. Hiện tại, các hàm băm mật mã phổ biến rất dễ bị tấn công kéo dài độ dài: cho trước hash(m) và len(m), không phải m, bằng cách chọn một giá trị m′ phù hợp, kẻ tấn công có thể tính toán hash(m ‖ m′), trong đó ‖ là phép ghép nối. Tính chất này có thể được sử dụng để phá vỡ các sơ đồ xác thực đơn giản dựa trên hàm băm. Việc tạo mã chứng thực thông điệp dùng hàm băm – HMAC (Hash Message Authentication Code) khắc phục được vấn đề này.

Trong thực tế, khả năng kháng đụng độ là không đủ cho nhiều ứng dụng thực tế. Ngoài khả năng kháng đụng độ, kẻ tấn công không thể tìm thấy hai thông điệp có nội dung cơ bản giống nhau; hoặc suy ra được bất kỳ thông tin hữu ích nào về dữ liệu, mà chỉ dựa trên bản tóm tắt của dữ liệu. Cụ thể, hàm băm phải hoạt động giống hàm ngẫu nhiên nhất có thể, nên thường được gọi là sấm ngữ ngẫu nhiên (oracle) trong các bằng chứng về bảo mật, trong khi vẫn có tính tất định và tính hiệu quả. Điều này loại trừ các hàm như hàm băm xác suất dựa trên biên đổi Fourier– SWIFFT, có thể được chứng minh chặt chẽ là có khả năng kháng đụng độ với giả định rằng một số bài toán là khó tính toán, nhưng, với một hàm tuyến tính, không thỏa mãn các tính chất bổ sung thêm này.



3 MÃ CÔNG KHAI

CẤU TRÚC NHÓM, VÀNH, TRƯỜNG

Nhóm (*group*), vành (*ring*), và trường (*field*) là 3 cấu trúc đại số quan trọng thường được sử dụng để thiết lập các hệ mã, đặc biệt là các hệ mã công khai phổ biến hiện nay.

Nhóm

Phép toán 2-ngôi

Trên tập hợp G khác rỗng, ta định nghĩa một phép toán hai ngôi, ký hiệu $\otimes$,

$\otimes$: $G \times G \rightarrow G$,

Sao cho với mọi phân tử a, b, c thuộc G, thỏa 4 tiên đề nhóm sau:

(1) $a \otimes b \in G$ (tính đóng).

(2) $a \otimes (b \otimes c) = (a \otimes b) \otimes c$ (tính kết hợp).

(3) $\exists e \in G$: $a \otimes e = a = e \otimes a$, e được gọi là phần tử đơn vị của G.

(4) $\exists a^{-1} \in G$: $a \otimes a^{-1} = e = a^{-1} \otimes a$, a^{-1} được gọi là phần tử nghịch đảo của a.

Tập G với phép toán $\otimes$ tạo thành một cấu trúc đại số, gọi là nhóm (*group*), ký hiệu $(G, \otimes)$.

Ví dụ

VD1. Tập các số nguyên $\mathbb{Z}$ với phép cộng thông thường là một nhóm $(\mathbb{Z}, +)$.

Thực vậy, với mọi số nguyên x, y, z, ta có (1) x + y cũng là số nguyên, (2) (x + y) + z = x + (y + z) = x + y + z, (3) 0 là phần tử đơn vị: x + 0 = x = 0 + x, và (4) -x là phần tử nghịch đảo của x: x+ (–x) = 0.

VD2. Dễ dàng kiểm tra tập các số nguyên chẵn, ký hiệu $2\mathbb{Z}$, với phép cộng thông thường $(2\mathbb{Z}, +)$ thỏa 4 tiên đề nhóm.

VD3. Tập các số lẻ $\mathbb{Z}l = \{2n + 1, n \in \mathbb{Z}\}$ không phải là nhóm với phép cộng thông thường vì cộng hai số lẻ sẽ cho kết quả là một số chẵn không thuộc $\mathbb{Z}l$.

VD4. Tập các số thực $\mathbb{R}$ với phép nhân thông thường không phải nhóm vì số thực 0 không có nghịch đảo. Nhưng $(\mathbb{R}\backslash\{0\}, *)$ là nhóm.

Nếu phép toán ⊗ có tính giao hoán, nghĩa là a ⊗ b = b ⊗ a với mọi a, b thuộc G, thì nhóm (G, ⊗) được gọi là nhóm giao hoán.

Các nhóm ví dụ vừa khảo sát ở trên đều là nhóm giao hoán. Ví dụ sau minh họa một nhóm không giao hoán.

VD5. Tập tất cả các ma-trận vuông cấp n khả nghịch $\mathcal{M}_n$ với các phần tử thực, cùng với phép nhân ma-trận $(\mathcal{M}_n, \cdot)$ là nhóm không giao hoán. Thực vậy, (1) Tích của 2 ma-trận vuông khả nghịch cấp n là ma-trận khả nghịch, (2) (A·B)·C = A·(B·C) đúng với mọi ma-trận vuông cùng cấp, (3) A·I = A = I·A đúng với mọi ma-trận vuông cấp n và I là ma-trận đơn vị cấp n, (4) Do $\mathcal{M}_n$ là tập tất cả các ma-trận cấp n khả nghịch nên A và A^{-1} cũng thuộc về $\mathcal{M}_n$. Mặt khác, luôn tồn tại hai ma trận vuông cấp n khả nghịch A, B thỏa A·B khác B·A, nên $(\mathcal{M}_n, \cdot)$ là nhóm không giao hoán.

Do tính chất kết hợp của phép ⊗ nên (x ⊗ x) ⊗ x = x ⊗ (x ⊗ x) = x ⊗ x ⊗ x. Trong trường hợp tổng quát, n phần tử x kết hợp với nhau, x ⊗ … ⊗ x, được ký hiệu và định nghĩa như sau:

$$x \otimes \ldots \otimes x = \begin{cases} e, n = 0 \\ x^n, n > 0 \end{cases}$$

Nhóm con

Cho (G, ⊗) là nhóm, nếu H ⊂ G và (H, ⊗) là nhóm thì H được gọi là nhóm con của G.

Ví dụ $(2\mathbb{Z}, +)$ trong VD2 ở trên là nhóm con của $(\mathbb{Z}, +)$. Hay tập các số hữu tỷ không chứa phần tử 0, $\mathbb{Q}\backslash\{0\}$ với phép nhân thông thường là nhóm con của nhóm $(\mathbb{R}\backslash\{0\}, *)$.

Cho nhóm (G, ⊗) và tập con G ⊃ S ≠ {} và có hữu hạn các phần tử. Đặt H là tập tất cả các phần tử tạo thành bằng cách kết hợp các phần tử trong S theo phép ⊗.

Nếu (H, ⊗) là nhóm, ta gọi (H, ⊗) là nhóm sinh bởi S, ký hiệu (<S>, ⊗) hay đơn giản là <S>.

Nếu S chỉ có 1 phần tử, S = {a}, thì H = <S> = <a> được gọi là nhóm đơn sinh, hay nhóm tuần hoàn (*cyclic group*). Phần tử a gọi là phần tử sinh của H.

VD6. $< 1 > = (\mathbb{Z}, +)$.

VD7. Đặt $\mathbb{Z}_7^+ = \{1, 2, 3, 4, 5, 6\}$ và $*_7$: $\mathbb{Z}_7 \times \mathbb{Z}_7 \to \mathbb{Z}_7$ xác định bởi x $*_7$ y = x*y % 7 = z là phần dư của phép chia của tích x*y cho 7. Thì $(\mathbb{Z}_7, *_7)$ là nhóm. Thực vậy, (1) $\forall x, y \in \mathbb{Z}_7, x *_7 y \in \mathbb{Z}_7$, (2) $(x *_7 y) *_7 z = x *_7 y *_7 z$ đúng với mọi $x, y, z \in \mathbb{Z}_7$, (3) 1 là phần tử đơn vị của $\mathbb{Z}_7$, và (4) $1^{-1} = 1$, $2^{-1} = 4$, $3^{-1} = 5$, $4^{-1} = 2$, $5^{-1} = 3$, $6^{-1} = 6 \in \mathbb{Z}_7$. Từ đây ta viết $\mathbb{Z}_7$ thay cho $\mathbb{Z}_7^+$ nếu không có ghi chú đặc biệt nào khác.

Định lý Lagrange

Nhóm (G, ⊗) là nhóm hữu hạn nếu số phần tử của G hữu hạn. Đặt |G| = n < ∞, thì n được gọi là bậc (order) của nhóm G, ký hiệu Ord(G).

Bên cạnh khái niệm bậc của nhóm, các phần tử của nhóm cũng có khái niệm bậc.

Bậc của một phần tử a ∈ (G, ⊗), ký hiệu ord(a), là số nguyên dương n nhỏ nhất sao cho $a^n = e$, với e là phần tử đơn vị của nhóm G.

Từ khái niệm bậc của nhóm và của phần tử thuộc nhóm, ta có kết quả:

Nếu bậc của phần tử a thuộc nhóm G bằng với bậc của nhóm, ord(a) = Ord(G), thì <a> = G và a được gọi là phần tử sinh của G.

Định lý Lagrange sau cho biết quan hệ về bậc giữa nhóm G và các nhóm con H của G, và được phát biểu như sau:

Nếu H là nhóm con của G thì bậc của H là ước của bậc của G: $Ord(G) \vdots Ord(H)$

Như vậy, nếu bậc của nhóm G là số nguyên tố thì G chỉ có 2 nhóm con là H = {e} và chính nó, H = G. $\mathbb{Z}_7^+$ trong **VD7** chỉ có 2 nhóm con {1} và $\mathbb{Z}_7^+$. Trong nhóm $\mathbb{Z}_7^+$, phần tử $3 \in \mathbb{Z}_7^+$ có bậc bằng 6, và <3> = {$3^0 = 1$, $3^1 = 3$, $3^2 = 2$, $3^3 = 6$, $3^4 = 4$, $3^5 = 5$}. Một cách tổng quát, ta có

$(\mathbb{Z}_p^+, \%)$ là nhóm đơn sinh nếu p là số nguyên tố.

Vành

Xét tập hợp khác rỗng R với 2 phép toán + và · được định nghĩa trên R có tính chất

(1) + và · có tính đóng.

(2) + và · có tính kết hợp.

(3) Tồn tại phần tử đơn vị cho phép cộng kí hiệu là 0 và cho phép nhân kí hiệu là 1.

(4) Phép + có tính giao hoán.

(5) Tồn tại phần tử nghịch đảo cho phép cộng (+) và nhân (·).

(6) Phép nhân phân phối với phép cộng: a · (b + c) = a·b + a·c, (b + c)·a = b·a + c·a.

R với hai phép + và · định nghĩa như trên gọi là vành (*ring*), ký hiệu (R, +, ·).

VD8. Tập các ma trận vuông cấp n với phép cộng và nhân ma-trận là một vành.

VD9. Tập số nguyên ℤ với các phép cộng và nhân thông thường là vành.

VD10. $\mathbb{Z}_n$ với các phép cộng (modulo n) và phép nhân (modulo n) là vành.

Trường

Vành (F, +, ·) khác rỗng với 2 phép toán + và · định nghĩa như trên, nếu mọi phần tử khác 0 của F khả nghịch, thì F được gọi là trường (*field*).

Trường hữu hạn là trường có số phần tử hữu hạn.

MẬT MÃ KHÓA CÔNG KHAI

Nguyên lý thiết kế

Hàm một chiều có cửa mật

Trong các hệ mã công khai (*public key cryptosystem*) hay bất đối xứng (*asymmetric crypyosystem*), các tiến trình mã hóa và giải mã sử dụng các khóa (*key*) khác nhau. Khóa mã hóa (*enctyption key*), ký hiệu **e**, có thể được công bố, nên còn gọi là khóa công khai (*public key*), nhưng khóa giải mã (*decryption key*) – **d**, phải giữ bí mật, gọi là khóa cá nhân (*private key*). Các hệ mã công khai thường được xây dựng dựa trên ý tưởng ***hàm* 1-*chiều có cửa mật*** (*one-way function with trapdoor*). Một cách hình thức, được định nghĩa như sau:

Cho các tập hữu hạn khác rỗng S và T. Hàm 1-chiều $f: S \to T$ là hàm khả nghịch có các tính chất:

(1) f dễ thực hiện. Nghĩa là cho $x \in S$, $y = f(x) \in$ T là dễ tính theo nghĩa có thể tính được với độ phức tạp tối đa là đa thức.

(2) f^{-1}, hàm ngược của f, khó thực hiện. Nghĩa là cho $y \in$ T, rất khó tìm được $x \in$ S sao cho $f(x) = y$ hay $x = f^{-1}(x)$.

(3) f^{-1} có thể tính được với thời gian tối đa đa thức khi có thêm một số thông tin cửa mật (*trapdoor*).

Một số ví dụ

VD1. f: pq $\to$ n là hàm 1-chiều. Trong đó p và q là hai số nguyên tố lớn và khác nhau, Thực vậy, tích pq có thể dễ thực hiện chỉ bằng phép nhân số nguyên lớn (với độ phức tạp đa thức); nhưng tính f^{-1}: n $\to$ (p, q) được chứng minh là bài toán khó, đó chính là bài toán phân tích ra thừa số nguyên tố (**IFP** – *integer factorization problem*) với độ phức tạp mũ. Nhưng f^{-1} tính p sẽ trở nên dễ nếu ta có *cửa mật* q, khi ấy, chỉ phải thực hiện 1 phép chia số nguyên lớn.

VD2. $f_{e,n}$: $x \to x^e \pmod n$ là hàm 1-chiều, với $n = pq$ là tích 2 số nguyên tố bí mật, lớn và phân biệt, còn e là số nguyên nguyên tố cùng nhau với $\varphi = (p - 1)(q - 1)$. Phép tính $x^e \pmod n$ có độ phức tạp đa thức, nhưng f^{-1}: $x^e \to x$ chỉ tính được khi biết cửa mật p (hay q), đó chính là hàm 1-chiều ở ví dụ 1. Nếu biết cửa mật p, việc tính $\varphi = (p - 1)(q - 1)$ dễ, dễ tìm được d sao cho tích $de = 1 \pmod \varphi$ bằng thuật toán Euclide. Khi ấy, ta có thể dễ dàng tính lại được x theo công thức $x = (f^{-1})^d = (x^e)^d = x^{ed}$.

Quy trình thiết kế hệ mã công khai

Nguyên lý cơ bản để thiết kế một hệ mã công khai là định nghĩa được hàm-một-chiều-có-cửa-mật và công khai các thông tin về hàm một chiều nhưng giữ bí mật thông tin cửa mật.

Có thể có nhiều cách xây dựng hàm một chiều, nhưng cách phổ biến và dễ thực hiện là sử dụng một bài toán khó (*hard problem*) đã và còn đang được công nhận trong lĩnh vực khoa học máy tính. Trong khoa học máy tính, bài toán khó là bài toán có thuật giải (*algorithm*) nhưng chương trình máy tính (*computer program*) tính nó là không thể thực hiện trong thời gian chấp nhận được. Các bài toán khó đang được sử dụng phổ biến trong mã hóa-mật mã phải kể đến gồm:

(**IFP**) *Integer Factorization Problem* – bài toán phân tích số nguyên lớn ra các thừa số nguyên tố. Bài toán được mô tả như sau:

IFP. Cho 2 số nguyên tố lớn và khác nhau p, q. Việc tính n = pq là dễ, nhưng việc tìm lại p hay q từ phân tích n là rất khó, trừ khi biết q.

(**DLP**) *Discrete Logarithm Problem* – bài toán logarithm rời rạc, gọi tắt là ***log rời rạc***. Bài toán được phát biểu như sau:

DLP. Cho g là phần tử sinh (*generator*) trên trường hữu hạn (*finite field*) F, và *x* là một số nguyên. Việc tính $y = g^x \in$ F là dễ, nhưng để tính *x* khi biết *y*, g là rất khó.

Bên cạnh đó, bài toán giải hệ phương trình lấy nghiệm nguyên gần đúng của một hệ phương trình tuyến tính có số phương trình ít hơn số ẩn cũng được xem là khó và có thể được dùng để xây dựng các bài toán khó trên đó. Các bài toán như bài toán tổng con (*subset problem*) hay các bài toán liên quan đến dàn (*lattice*) là những ví dụ cho ứng dụng này. Tài liệu này giới hạn trong các hệ mã phổ biến đang được sử dụng cho chuỗi khối (*blockchain*) dựa trên 2 bài toán **IFP** và **DLP**.

Hệ mã RSA

RSA – Rivest–Shamir–Adleman, viết tắt tên của 3 tác giả phát minh ra hệ mã RSA. RSA được xây dựng dựa trên giả thuyết khó giải bài toán phân tích ra thừa số. Ta sẽ phân tích và cài đặt định lý RSA sau:

Định lý RSA

Định lý (RSA). Cho p và q là 2 số nguyên tố khác nhau. Đặt n = pq, φ = (p – 1)(q – 1), và chọn 2 số e và d thỏa ed % φ = 1, với % là phép chia lấy phần dư. Với mọi $m \in \mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$, nếu $c = m^e \,\%\, n$ thì $m = c^d \,\%\, n$.

Lớp tương đương modulo n– **Equivalence class (mod n)**

Trong định lý RSA, $\mathbb{Z}_n$ với phép + (module n), định nghĩa bởi $x + y \,(mod\, n) = (x + y) \,\%\, n$ và phép * (modulo n), định nghĩa bởi $x * y \,(mod\, n) = xy \,\%\, n$, hình thành nên một vành hữu hạn *n* phần tử, vành $(\mathbb{Z}_n, +, *)$.

Theo ký hiệu phép * (mod), biểu thức $ed \,\%\, \varphi = 1$ được viết lại thành $e * d \,(mod\, \varphi) = 1$, ký hiệu là $e * d \equiv 1 \,(mod\, \varphi), \forall e, d \in \mathbb{Z}$. Một cách tổng quát:

$y \equiv x \,(mod\, z)$ là tập tất cả các số nguyên *y* chia cho *z* dư *x*, và gọi là lớp tương đương của *x*.

Sinh khóa – **key generating**

Việc chọn *e* và *d* thỏa ràng buộc $e * d \equiv 1 \ (mod\ \varphi)$ có thể thực hiện được bằng cách chọn một số $e \in \mathbb{Z}_\varphi$ khả nghịch và *d* chính là e^{-1}: $d \equiv e^{-1} \pmod{\varphi}$. Về mặt cài đặt, có thể sử dụng thuật giải Euclide mở rộng (*extended Euclidian Algorithm*) hay còn gọi là định lý Bezout, người đã mở rộng định lý Euclide tính ước chung lớn nhất của 2 số nguyên dương, để chọn e và d.

Định lý (Bezout). Với mọi số nguyên dương a, b, luôn tồn tại hai số nguyên x, y thuộc $\mathbb{Z}$ sao cho gcd(a, b) = ax + by, trong đó gcd(a, b) là ước chung lớn nhất của a và b.

Như vậy, thuật toán tìm cặp e và d, gọi là thuật toán sinh khóa (**keyGen** – *key generating*) có thể được mô tả bằng mã giả như sau:

Algorithm keyGen(φ): #φ = (p – 1)(q – 1) với p và q là 2 số nguyên tố phân biệt.

(1) Chọn ngẫu nhiên một số lẻ $e \in \mathbb{Z}_\varphi$.

(2) Nếu gcd(e, φ) = ex + φy = 1, thì trả về d = x; nếu không, quay lại (1).

Mã hóa và giải mã – **Encryption and Decryption**

Nếu xem $m \in \mathbb{Z}_n$ là thông điệp (*message*) thì $c = m^e \pmod{n} \in \mathbb{Z}_n$ là bản mã tương ứng được mã bằng khóa công khai e. Khi ấy, để giải mã, ta thực hiện phép tính $c^d \pmod{n}$. Theo định lý RSA, ta có $c^d \equiv m \pmod{n}$. Ta sẽ sử dụng định lý Fermat nhỏ (*little Fermat*) và số dư Trung Hoa (**CRT** – *Chinese Remainder Theorem*) để chứng minh phát biểu này.

Định lý (Fermat nhỏ). Nếu p là số nguyên tố và b là số nguyên bất kỳ không phải bội số của p, thì $b^{p-1} \equiv 1 \pmod{p}$.

Ví dụ cho p = 5. Với b = 2, ta có $2^{5-1} \equiv 2^4 \equiv 16 \equiv 1 \pmod{5}$. Với b = 3, $3^4 \equiv 91 \equiv 1 \pmod{5}$. Bây giờ, cho p = 11 và chọn b = 2, $2^{10} \equiv 1024 \equiv 1 \pmod{11}$; với b = 9, $9^{10} \equiv 3486784401 \equiv 1 \pmod{11}$.

Định lý (CRT). Nếu $m_1,\ldots, m_k$ là k số nguyên đôi một nguyên tố cùng nhau (*co-prime*), nghĩa là $gcd(m_i, m_j) = 1$ với mọi i, j = 1, 2, …, k và $i \neq j$; và $a_1,\ldots, a_k$ là k số nguyên bất kỳ, thì hệ

$$\begin{cases} x \equiv a_1 \ (mod\ m_1) \\ \quad\quad ... \\ x \equiv a_k \ (mod\ m_k) \end{cases} (*),$$

có duy nhất nghiệm trong $\mathbb{Z}_m$, với $m = m_1 \ldots m_k$.

Chứng minh định lý ***CRT***. Nghiệm x của hệ đồng dư (*) được xây dựng qua thuật giải sau (phần chứng minh tính duy nhất xem như bài tập).

Algorithm CRT

(1) $m = m_1 * \ldots * m_k$

(2) $n_i = m/m_i$, $\forall i = 1, 2,\ldots, k$ #ta có $gcd(n_i, m_i) = 1$, nên $n_i^{-1} \ (mod\ m_i)$ tồn tại.

(3) $N_i \leftarrow Bezout(n_i, m_i), \forall i = 1,2, \dots, k$ #sử dụng định lý Bezout để tìm $n_i^{-1}\ (mod\ m_i)$.

(4) $x = \left(\sum_{i=1}^{k} a_i n_i N_i (mod\ m_i)\right)\ (mod\ m)$

#ta có

$x\ \%\ m_1 = a_1 * n_1 * n_1^{-1}\ \%\ m_1 + a_2 * n_2 * n_2^{-1}\ \%\ m_2 \dots + a_k * n_k * n_k^{-1}\ \%\ m_k\ \equiv 1 + 0 + \dots + 0\ (mod\ m)$

#...

$x\ \%\ m_k = a_k * n_k * n_k^{-1}\ \%\ m_k + a_k * n_k * n_k^{-1}\ \%\ m_k \dots + a_k * n_k * n_k^{-1}\ \%\ m_k\ \equiv 0 + 0 + \dots + 1\ (mod\ m)$

#Chính là hệ (*), nên x chính là nghiệm của (*).

(5) return *x*.

Chứng minh định lý **RSA – prove RSA**

Ta có $c^d \equiv (m^e)^d \equiv m^{ed}$ (mod n).

Do ed ≡ 1 (mod φ) nên có một k sao cho ed = kφ + 1. Suy ra

$m^{ed} \equiv m^{k\varphi + 1} \equiv m.m^{k\varphi}$ (mod n)

hay ta phải chứng minh $m^{k\varphi} \equiv 1$ (mod n) (*)

Trường hợp m khác p và q, ta có

$m^{k\varphi} \equiv m^{k(p-1)(q-1)} \equiv (m^{p-1} \bmod p)^{k(q-1)} \equiv 1$ (mod p) #theo định lý Fermat nhỏ $m^{(p-1)} \bmod p = 1$,

tương tự $m^{k\varphi} \equiv 1$ (mod q).

Do p và q là 2 số nguyên tố nên nguyên tố cùng nhau, đặt $X \equiv m^{k\varphi}$ (mod n) hệ

$\begin{cases} X \equiv 1\ (mod\ p) \\ X \equiv 1\ (mod\ q) \end{cases}$ (**),

có duy nhất nghiệm trong $\mathbb{Z}_{pq} = \mathbb{Z}_n$.

Ta thấy, X ≡ 1 (mod pq) ≡ 1 (mod n) là nghiệm của hệ (**).

Do tính duy nhất nghiệm, ta có $X \equiv m^{k\varphi} \equiv 1$ (mod n).

Bây giờ, giả sử m = p, ta có $m^{k\varphi} \equiv p^{k\varphi} \equiv 0$ (mod p),

và ta cũng có $m^{k\varphi} \equiv p^{k\varphi} \equiv (p^{(q-1)} \bmod q)^{k(p-1)} \equiv 1$ (mod q).

Tóm tại, ta vẫn có hệ đồng dư Trung Hoa

$\begin{cases} X \equiv 0\ (mod\ p) \\ X \equiv 1\ (mod\ q) \end{cases}$ (**),

(Tương tự cho trường hợp m = q).

Giao thức RSA

Giao thức mã công khai RSA (*RSA protocol*) là cài đặt trực tiếp của định lý RSA. Theo đó, Bob cần gửi thông điệp mật của m cho Alice, Alice và Bob theo các bước của giao thức sau:

Giao thức – **protocol**

Alice	Bob
Bước 1: sinh khóa – key generating . p, q ← **primeGen**(λ) #sinh 2 số nguyên tố lớn λ-bit . n ← pq . φ ← (p – 1)(q – 1) . e, d ← **keyGen**(φ) #định lý Bezout #Sau đó, Alice công bố (e, n) và giữ bí mật d, p, q, và φ.	(e, n) được cho công khai để ai cũng có thể gửi tin mật cho Alice.
c được chuyển trên mọi kênh, kể cả kênh không bảo mật và ai cũng có thể lấy được c.	**Bước 2**: mã hóa – encryption . $c = m^e \% n$ #Và chuyển c cho Alice trên kênh không bảo mật.
Bước 3: giải mã – decryption . $m \leftarrow c^d \% n$	chỉ Alice có khóa cá nhân d mới giải mã và đọc được c

Bên cạnh các phép tính trên các số nguyên lớn mà đã có nhiều thư viện cung cấp. Giao thức RSA cần cung cấp hàm sinh khóa – keyGen (là hàm ta đã có thuật giải từ việc cài đặt thuật giải Bezout như mã giả ở phần trước), và hàm sinh số nguyên tố lớn – primeGen. Ta có thể sử dụng định lý sau để tạo số nguyên tố lớn.

Định lý (dùng tạo số nguyên tố). Cho p là số nguyên tố lẻ và k, $1 < 2(p+1)$, là số nguyên tự nhiên không phải bội của p và đặt $n = 2kp + 1$. Các phát biểu sau là tương đương:

(i) n là số nguyên tố.

(ii) Tồn tại một số nguyên tự nhiên a, $2 \le a < n$, thỏa đồng thời

(ii.a) $a^{kp} \equiv 1 \pmod n$ và (ii.b) $\gcd(a^k + 1, n) = 1$.

Mã giả của hàm primeGen sau sinh số nguyên tố thỏa định lý trên.

Algorithm primGen(d) #tạo số nguyên tố có d ký số

(1) Chọn 1 số nguyên tố nhỏ p_1 có d_1 ký số, và tìm số $k_1 < 2(p_1 + 1)$ sao cho số $p_2 = 2k_1 + 1$ có d_2 ký số, $d_2 = 2d_1 - 1$ và có số $a_1 < p_2$ thỏa 2 điều kiện $a_1^{k_1 p_1} \equiv 1 \ (mod\ p_2)$ và $\gcd\left(a_1^{k_1} + 1, p_2\right) = 1$. Theo định lý trên, p_2 là số nguyên tố.

(2) gán $p_2 = p_1$ và lặp lại (1) cho đến khi được số nguyên tố có $d_2 = d$ ký số.

Các thuật toán sinh khóa (bao hàm cả sinh số nguyên tố), có thể được cho mã nguồn trên mạng. Tuy nhiên, cần rất thận trọng khi sử dụng các nguồn mở cho hai hàm này. Tốt nhất là

nên tự thực hiện để tránh mọi rủi ro khi triển khai cho ứng dụng hay cho sổ-cái-phi-tập-trung – blockchain.

Bên cạnh các thuật toán sinh khóa, việc áp dụng RSA vào thực tế, nhất là cho blockchain, tận dụng các tri thức về p, q giúp cho việc giải mã nhanh. Thực vậy, giả sử cần tính c^d để giải mã bản mã c. Ta thực hiện theo cách sau:

– Đặt $d_1 = d \% (p - 1)$ và $d_2 = d \% (q - 1)$. Khi đó, tồn tại n_1 và n_2 thỏa

$d = d_1 + n_1(p - 1) = d_2 + n_2(q - 1)$ và $c^d \equiv c^{d_1+n_1(p-1)} \equiv c^{d_1}c^{n_1(p-1)} \equiv c^{d_1}(mod\ p)$ (*).

Tương tự $c^d \equiv c^{d_2}(mod\ q)$. (**)

– Do p và 1 là 2 số nguyên tố phân biệt nên hệ (*) và (**) có duy nhất nghiệm trong $\mathbb{Z}_{pq} = \mathbb{Z}_n$.

Thuật giải Garner giải hệ đồng dư Trung hoa 2 phương trình hiệu quả hơn thuật giải CRT tổng quát trên.

Algorithm Garner(a, b, p, q)

giải hệ: $x \equiv a \pmod p$ và $x \equiv b \pmod q$

(1) c ← Bezout(p, q) # tính $c \equiv p^{-1} \pmod q$

(2) $u = (b - a)*c \pmod q$

(3) $x = a + u*p$

(4) return x

Độ an toàn của RSA

Độ an toàn của hệ mã RSA dựa trên giả thuyết RSA:

RSA conjecture: Mọi phương pháp thám mã RSA phải khó như giải bài toán phân tích thừa số.

Tuy nhiên, RSA có thể không an toàn trong những tình huống cụ thể. Chẳng hạn, hai trường hợp sau dễ khiến RSA bị phá vỡ.

Khóa cá nhân nhỏ: Khi khóa cá nhân d nhỏ, chiều dài bit nhỏ hơn ¼ số bit của n. Wiener đã xây dựng thành công thuật toán phục hồi khóa d nhỏ, $d < n^{1/4}$, dựa trên kỹ thuật liên phân số. Sau này, các tấn công dàn (lattice-based attack) cũng thành công khi d nhỏ. Vì thế, để an toàn, d phải lớn.

Khóa công khai có bậc nhỏ: Khi khóa cá nhân có bậc r nhỏ trong một nhóm λ(n). Khi ấy $e^r \equiv 1 \pmod{\lambda(n)}$, vì thế $m^{e^r} \equiv m\ (mod\ n)$. Đây chính là lần lặp thứ r của hàm mã hóa thông điệp m. Vì thế, để an toàn, r phải lớn.

HỆ MÃ DIFFIE-HELLMAN

Hệ mã hỗn hợp

Khác với RSA dựa trên bài toán phân tích ra thừa số, Diffie-Hellman, tên của 2 tác giả phát minh ra, được xây dựng dựa trên bài toán log-rời rạc: ***cho g, p, g^x mod p, tìm r***. Nếu như RSA thiết lập hàm một chiều có cửa mật một cách gián tiếp từ bài toán nền, thì Diffie-Hellman được thiết lập bằng cách dùng trực tiếp *x* trong g^x làm cửa mật. Và thay vì mã hóa thông điệp, Diffie-Hellman thiết lập bí mật chung giữa các đối tác trong giao thức. Sau pha thiết lập khóa, khóa bí mật chung giữa các đối tác có thể được sử dụng để trao đổi tin mật theo cách mã đối xứng. Giao thức sau minh họa một hệ mã hỗn hợp.

Allice	Bob
. p ← primeGen(λ) #Allice và Bob quy ước dùng chung một trường $\mathbb{Z}_p$ và một phần tử sinh g	
Bước 1: thiết lập khóa chung – **key exchange** . $x \xleftarrow{\$} \mathbb{Z}$ #chọn ngẫu nhiên một số ngẫu nhiên . $k_A \leftarrow g^x \% p$ (giữ bí mật x và chuyển k_A cho Bob) . $k \leftarrow k_B^x \% p$	. $y \xleftarrow{\$} \mathbb{Z}$. $k_B \leftarrow g^y \% p$ (giữ bí mật y và chuyển k_B cho Allice) . $k \leftarrow k_A^y \% p$
Sau bước này, Allice và Bob có chung khóa bí mật $k \equiv g^{xy} \pmod p$	
Bước 2: mã hóa thông điệp m - **encryption** . c ← m*k % p (chuyển c cho Bob)	
	Bước 3: giải mã – **decryption** . m ← $c*k^{-1}$ % p

Hệ mã trên sử dụng giao thức Diffie-Hellman cho pha thiết lập khóa bí mật chung (key exchange protocol). Giao thức này sử dụng giả thiết Diffie-Hellman, mà thực chất là bài toán khó log-rời rạc (**DLP** – *Discrete Logarithm Problem*), đảm bảo độ an toàn của hệ mã.

Giả thuyết Diffie-Hellman (DH conjecture). Hầu như không thể tính được g^{xy} chỉ từ g^x, g^y.

Mặc dầu vậy, nếu số nguyên tố p là số nguyên tố có dạng p = 2q + 1, với q cũng là số nguyên tố[5], hệ Diffie-Hellman sẽ bị phá vỡ. Thực vậy, sử dụng kết quả của định lý sau:

Định lý. x là phần tử sinh của $\mathbb{Z}_m \Leftrightarrow x^{\frac{\phi(m)}{q}} \not\equiv 1 \ (mod\ m), \forall q | \phi(m), q$ là số nguyên tố. Với ϕ(m) là hàm ϕ-Euler trả về số các số nguyên dương nhỏ hơn m và nguyên tố cùng nhau với m.

Nếu p = 2q + 1, thì theo định nghĩa của hàm số học ϕ-Euler, ϕ(p) = 2q, và với phần tử sinh $x \in (1, p)$ ta có $x^2 \equiv 1 \pmod p$ và $x^q \equiv 1 \pmod p$.

[5] Hai số nguyên tố p và q thỏa p = 2q + 1 là số nguyên tố Mersenne.

Vậy $1 = g^{p-1} \equiv (g^q)^2 \pmod p = \beta^2$.

Theo định lý trên, $\beta \neq 1$ và $\beta^2 = 1$, suy ra $\beta = -1$.

Khi đó, $k \equiv g^{xyq} \equiv \beta^{xy} \pmod p \in \{-1, 1\}$.

Vì vậy, tin tặc có thể sử dụng tấn công trung gian (**MiM** attack – *Man in the Middle attack*), thay hai thông điệp g^x, g^y bằng các thông điệp mới g^{xq} và g^{yq}, thì khóa bí mật chung chỉ có hai giá trị là -1 và 1.

HỆ MÃ ĐƯỜNG CONG ELLIPTIC

Mã đường cong elliptic – **ECC** (Elliptic Curve Cryptosystem) là hệ mã khóa công khai hiện đang được sử dụng trong các chuỗi khối phổ biến như Bitcoin, Ethereum.

Sẽ có bài riêng về hệ mã này. Ở đây, chúng tôi chỉ trình bài ý tưởng chính của ECC.

Để cài đặt ECC, cần thực hiện trước các phép tính nhúng thông điệp vào đường cong elliptic. Mục đích là để có thể thực hiện mã hóa thông điệp bằng ECC trong một trường F_q. Cụ thể, muốn nhúng thông điệp rõ thành một điểm trên đường cong elliptic xác định trên trường F_q, giả sử $q = p^r$ và p là số nguyên tố. Xem thông điệp m như một số nguyên $0 \leq m \leq M$, và chọn k là một số nguyên, $lower \leq k \leq upper$. Với đường cong (E): $y^2 = x^3 + ax + b$ xác định trên F_q. Thông điệp m được nhúng như sau:

(1) Tính $x = mk + j$, $j = 0, 1, 2,\ldots$ cho đến khi tìm được $x^3 + ax + b$ là số chính phương (mod p), trả về điểm $(x, \sqrt{x^3 + ax + b})$ trên (E).

(2) Để chuyển điểm (x, y) trên (E) lại thành thông điệp m, tính $m = \lfloor x/k \rfloor$.

Ta sẽ quay trở lại hệ mã đường cong elliptic trong bài sau.

4 HỆ MÃ ĐƯỜNG CONG ELLIPTIC

HỆ MÃ TỰA ELGAMAL

Giao thức Diffie-Hellman

Trước khi bắt đầu, ta nhắc lại giao thức thiết lập khóa Diffie-Hellman (DH key exchange protocol), mà độ bảo mật dựa trên bài toán khó log-rời rạc (**DLP** – *Discrete Logarithm Problem*).

Allice	Bob
. p ← primeGen(λ) . g ← generator(p) #Allice và Bob quy ước dùng chung một trường $\mathbb{Z}_p$ và một phần tử sinh g	
Bước 1: thiết lập khóa chung – **key exchange** . $x \xleftarrow{\$} \mathbb{Z}$ #chọn một số ngẫu nhiên . $k_A \leftarrow g^x \% p$ (giữ bí mật x và chuyển k_A cho Bob) . $k \leftarrow k_B^x \% p$	. $y \xleftarrow{\$} \mathbb{Z}$. $k_B \leftarrow g^y \% p$ (giữ bí mật y và chuyển k_B cho Allice) . $k \leftarrow k_A^y \% p$
Sau bước này, Allice và Bob có chung khóa bí mật $k \equiv g^{xy} \pmod p$	

Như ta thấy, mục tiêu của giao thức-DH là thiết lập một bí mật chung giữa các đối tác. Không phải là hệ mã hóa-giải mã như RSA. ElGamal đã phát triển giao thức Difier-Hellman thành một hệ mã, mô hình này có thể được dùng để thiết lập các hệ mã khác khi ta chọn hay xây dựng được một bài toán khó và sử dụng trực tiếp nó kiểu Diffie-Helman thay vì không trực tiếp như kiểu RSA. Ta sẽ gọi chung là hệ mã tựa-ElGamal (ElGamal-like) vì theo mô-tip của ElGamal như trình bày sau.

Hệ mã ElGamal

Trong mô hình này, giả sử Bob muốn gửi thông điệp mật của thông điệp rõ m cho Allice. Bob và Allice thi hành giao thức sau trên trường $\mathbb{Z}_p$, với p là số nguyên tố lớn.

Allice	Bob
Bước 1: sinh khóa – **Key generating** . p ← primeGen(λ) #tạo số nguyên tố p . g ← generator(p) #chọn phần tử sinh g . $d \xleftarrow{\$} \mathbb{Z}_p$ #chọn ngẫu nhiên 1 khóa cá nhân d . e ← g^d % p #tính và công bố khóa e	Cặp khóa (e, d) với (e, p, g) được công bố và d phải giữ bí mật.
Giá trị x ngẫu nhiên làm cho hệ mã có tính an toàn ngữ nghĩa, nghĩa là cùng bản rõ, các lần mã khác nhau sẽ cho bản mã khác nhau.	**Bước 2**: mã hóa – **Encryption** . $x \xleftarrow{\$} \mathbb{Z}_p$ #chọn ngẫu nhiên số nguyên x . $c_1 \leftarrow g^x \pmod p$. $c_2 \leftarrow m*e^x \pmod p$. C ← (c_1, c_2) #tính C và gửi C cho Allice
Bước 3: giải mã – **Decryption** . $u \leftarrow c_1^d \ (mod\ p)$. $v \leftarrow u^{-1} \pmod p$. m ← c_2*v (mod p)	Ta thấy $c_2 * v \equiv (m * e^x) * u^{-1} \equiv * (g^d)^x \left(c_1^d\right)^{-1}$ $\equiv m * g^{dx}((g^x)^d)^{-1} \equiv m * g^{dx}(g^{dx})^{-1}$ $\equiv m\ (mod\ p)$

Độ an toàn của ElGamal

Độ an toàn của hệ mã dựa trên giả thuyết Diffie-Hellman: DLP là bài toán khó.

Tuy nhiên, trong trường hợp x bị sử dụng lại, ElGamal có thể bị phá vỡ. Thực vậy, giả sử $C = (c_1, c_2) = (g^x, m * g^{dx}), và\ C' = (c_1', c_2') = (g^x, m' * g^{dx})$. Nếu biết m, c_1 và c_2, ta có thể suy ra m' như sau:

$(m^{-1} * c_2)^{-1} * c_2' = \left(m^{-1} * (m * g^{dx})\right)^{-1} * (m' * g^{dx}) = (g^{dx})^{-1} * m' * g^{dx} = m'.$

Như vậy, để an toàn, hàm sinh số ngẫu nhiên phải thực sự ngẫu nhiên.

Mô hình hệ mã tựa-ElGamal

Qua xem xét quy trình xây dựng hệ mã ElGamal đựa trên cấu trúc $(\mathbb{Z}_p, *)$, ta thấy, để xây dựng một hệ mã theo kiểu ElGamal, gọi là tựa-ElGamal (ElGamal-like) ta thực hiện các bước sau:

Bước 1: Xây dựng được một cấu trúc đại số (G, ⊗) sao cho có thể định nghĩa được bài toán khó kiểu log-rời rạc (DLP-like). Nghĩa là cho phần tử sinh g ∈ G và y = g ⊗…⊗ g = g^x ∈ G, với x là một số nguyên bí mật, thì tìm lại x từ g và y là bài toán khó.

Bước 2: Nhúng thông điệp, là một số nguyên thành một phần tử m thuộc G.

Bước 2: Sử dụng lược đồ ElGamal trên (G, ⊗) ta được hệ mã tựa-ElGamal. Nghĩa là với thông điệp nhúng m, khóa công khai e, phần tử sinh g, thì bản mã của m là

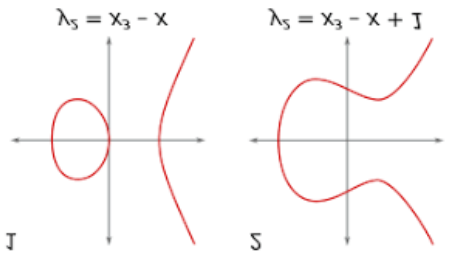
cặp C = (g^x, m⊗e^x). Bản mã C = (c_1, c_2) ∈ G×G được giải mã thành lại thông điệp nhúng m = c_2⊗((c_1)^d)$^{-1}$.

Ta sẽ sử dụng quy trình xây dựng ElGamal-like này để khảo sát và xây dựng hệ mã đường cong elliptic.

HỆ MÃ ĐƯỜNG CONG ELLIPTIC

Đường cong elliptic

Đường cong elliptic E trên trường hữu hạn Fq là tập hợp tất cả các điểm (x, y) nằm trên đường cong (E): $y^2 = x^3 + ax + b$, trong đó a, b thuộc một trường F_q và thỏa $4a^3 + 27b^2 \neq 0$. Hình dưới minh họa 2 đường cong elliptic đặc biệt.



Phép cộng điểm trên đường cong elliptic

Trên tập E = {(x, y) ∈ $\mathbb{Z}_p$×$\mathbb{Z}_p$: $y^2 = x^3 + ax + b$} ∪ {O}, ta định nghĩa phép toán 2 ngôi

+: E × E → E như sau.

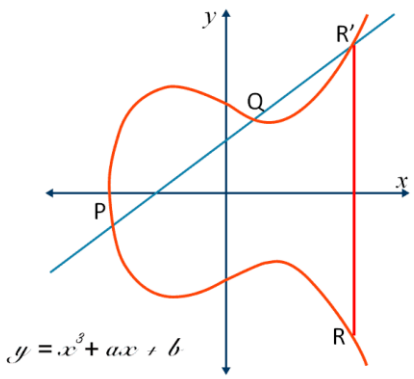
. Nếu P = (x_1, y_1) ∈ E thì –P = (x_1, -y_1).

. Nếu Q = (x_2, y_2) ∈ E, Q = –P thì R = Q + P = (x_3, y_3), trong đó

$x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda*(x_1 - x_3) - y_1$ và

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{nếu } P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & \text{nếu } P = Q \end{cases}.$$

Hình dưới minh họa quy tắc phép cộng (+) điểm trên đường cong elliptic.



Ví dụ minh họa

Xét đường cong (E): $y^2 = x^3 + ax + b = x^3 + x + 1$ xác định trên $\mathbb{Z}_{13}$ với phần tử sinh (generator) của $\mathbb{Z}_{13}$ là G = (0,1). Các điểm trên E có thể được biểu diễn the G.

VD1. Ký hiệu G = (0, 1) = (x_1, y_1) thì R = P + P = 2P = (x_3, y_3) được xác định theo

. $\lambda = \frac{3x_1^2+a}{2y_1}, (vì\ P = Q), do$ đó $\lambda \equiv \frac{3*0^2+1}{2*1} \equiv \frac{1}{2} \equiv 2^{-1} \equiv 7\ (mod\ 13)$.

. $x_3 = \lambda^2 - x_1 - x_2 \equiv 7^2 - 0 - 0 \equiv 10$ (mod 13).

. $y_3 = \lambda(x_1 - x_3) - y_1 \equiv 7(0 - 10) - 1 \equiv -69 \equiv 9$ (mod 13).

Vậy R = 2P = (10, 9).

VD2. Với P = (x_1, y_1) = (10, 9), Q = (x_2, y_2) = (1, 2), thì R = P + Q = (x_3, y_3) được xác định như sau:

. $\lambda = \frac{y_2-y_1}{x_2-x_1} \equiv \frac{2-9}{1-10} \equiv \frac{-7}{-9} \equiv \frac{6}{4} \equiv 6 * 4^{-1} \equiv 6 * 10 \equiv 60 \equiv 8.$

. $x_3 = \lambda^2 - x_1 - x_2 \equiv 8^2 - 10 - 1 \equiv 64 - 11 \equiv 1$ (mod 13).

. $y_3 = \lambda(x_1 - x_3) - y_1 \equiv 8(10 - 1) \equiv 8*9 \equiv 7.$

Vậy (10, 9) + (1, 2) = (1, 7).

Lưu ý

Khóa ECC 160 bit có độ an toàn tương đương với khóa RSA 1024 bit.

Hệ mã tựa-Elgamal trên đường cong elliptic

Các đường cong elliptic tạo thành các nhóm đơn sinh (cyclic group) và điểm G trên đường cong sinh ra mọi điểm của nhóm con đơn sinh rất hữu ích trong mật mã. Hệ mã dựa trên đường cong elliptic được gọi là hệ mã đường cong elliptic (ECC – Elliptic Curve Cryptosystem). Ký hiệu $E = \{(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p: y^2 = x^3 + ax + b\} \cup \{O\}$, với O là phần tử trung hòa ở vô cực. (E, $\oplus$) được chứng minh là hình thành nhóm giao hoán, được gọi nhóm các đường cong elliptic. Hơn nữa, bài toán log-rời rạc trên nhóm đường cong elliptic (**EDLP** – Elliptic Discrete Logarithm Problem) với mô tả:

"Cho điểm sinh G $\in$ E và Y = G $\oplus \ldots \oplus$ G = xG với x là số nguyên bí mật, tìm x" cũng được chứng minh là bài toán khó, đựa trên định lý về số điểm trên đường cong elliptic.

Định lý Hasse. Số điểm trên đường cong elliptic $E(F_{p^k})$ xác định trên trường F_{p^k} là $\#E(F_{p^k}) = p^k + 1 + t_{p^k}$, với t_{p^k} là số thỏa bất đẳng thức $|t_{p^k}| \leq 2p^{\frac{k}{2}}$.

VD3. $\#E(F_2) = 5$. Vì thế $E(F_2)$ không thể dùng cho mật mã.

Như vậy, bằng cách thay ký hiệu a*b, g^x trong $(\mathbb{Z}_p,*)$ lần lượt thành thành P+Q, xG trong (E, +), ta có hệ mã ElGamal trên đường cong elliptic.

Allice	Bob
Bước 1: sinh khóa – **Key generating** . p ← primeGen(λ)#tạo số nguyên tố p cho $\mathbb{Z}_p$. G ← generator(E) #chọn điểm sinh G . $d \overset{\$}{\leftarrow} \mathbb{Z}_p$ #chọn ngẫu nhiên 1 khóa cá nhân d . e ← dG #tính và công bố khóa e	Cặp khóa (e, d) với (e, $\mathbb{Z}_p$, G) được công bố và d giữ bí mật.
x ngẫu nhiên làm cho hệ mã có tính an toàn ngữ nghĩa, nghĩa là cùng bản rõ, các lần mã khác nhau sẽ cho bản mã khác nhau.	**Bước 2**: mã hóa – **Encryption** . $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ #chọn ngẫu nhiên số nguyên x . $c_1 \leftarrow xG$. $c_2 \leftarrow m + xe$. C ← (c_1, c_2) #tính C và gửi C cho Allice
Bước 3: giải mã – **Decryption** . $m = c_2 + (-dc_1)$	$c_2 + (-dc_1) = m + xe + (-d(xG))$ $= m + xe + (-dxG) = m + xdG + (-xdG) =$ m.

Nhúng thông điệp m vào đường cong elliptic

Mục đích là để có thể thực hiện mã hóa thông điệp bằng ECC trong một trường F_q. Cụ thể, muốn nhúng thông điệp rõ thành một điểm trên đường cong elliptic xác định trên trường F_q, giả sử $q = p^r$ và p là số nguyên tố. Xem thông điệp m như một số nguyên $0 \le m \le M$, và chọn k là một số nguyên, $lower \le k \le upper$. M, lower và upper được cho trước. Với đường cong (E): $y^2 = x^3 + ax + b$ xác định trên F_q. Thông điệp m được nhúng như sau:

Thủ tục nhúng thông điệp lên đường cong elliptic

(1) Tính x = mk + j, j = 0, 1, 2,… cho đến khi tìm được $x^3 + ax + b$ là số chính phương (mod p), trả về điểm $(x, \sqrt{x^3 + ax + b})$ trên (E).

(2) Để chuyển điểm (x, y) trên (E) lại thành thông điệp m, tính $m = \lfloor x/k \rfloor$.

VD4. Cho (E): $y^2 = x^3 + 3x$, và m = 2174, p = 4177.

Chọn k = 30 và tính x = {30×2174 + j, j = 0, 1, 2,…} cho đến khi $x^3 + 3x$ là số chính phương. Chẳng hạn khi j = 15, ta có

$x = 30 \times 2174 + 15 = 65235.$

$x^3 + 3x = (30 \times 2174 + 15)^3 + 3(30 \times 2174 + 15) = 277614407048580$

$\equiv 1444 \equiv 38^2 \pmod{4177}$.

Vậy thông điệp m = 2174 được nhúng thành điểm P(x, y) = P(65235, 38).

Để chuyển thông điệp nhúng P(65235, 38) về lại thông điệp gốc, ta tính

m = ⌊65235/30⌋ = 2174.

CÀI ĐẶT HỆ MÃ ĐƯỜNG CONG ELLIPTIC

Đường cong hữu ích

Không phải mọi đường cong elliptic đều có thể được sử dụng hiệu quả cho mật mã. Vì vậy, trong thực tiễn cài đặt, cần biết loại đường cong elliptic nào có ích trong mật mã học. Đường cong elliptic

(E): $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$,

với các hệ số thỏa biệt thức $\Delta \neq 0$, trong đó

$\Delta = -b_2^2 b_8 - 8b_4^3 - 27b_6^2 + 9b_2 b_4 b_6$,

với

$b_2 = a_1^2 + 4a_2$,

$b_4 = 2a_4 + a_1 a_3$,

$b_6 = a_3^2 + 4a_6$,

$b_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2$.

Các đường cong sau là các đường cong hữu ích cho mật mã trong thực tiễn:

- Đường cong Edwards: $x^2 + y^2 = 1 + dx^2y^2$.
- Đường cong xoắn: $–x^2 + y^2 = 1 + dx^2y^2$.
- Đường cong Montgomery: $y^2 = x^3 + ax^2 + x$.

Các ứng dụng trong thực tế thường cài đặt ECC với hai đường cong elliptic "**curve25519**" và "**curve448**"[6] xác định trên các trường nguyên tố. Những đường cong này nhằm mục đích xây dựng các hệ mã hoạt động ở mức bảo mật lần lượt mức 128-bit và 224-bit thỏa các yêu cầu bảo mật chuẩn.

Curve25519 và **curve448** có khả năng chống lại các tấn công kênh phụ (*side channel attacks*), bao gồm tấn công khai thác thời gian (*timing*) và bộ nhớ đệm (*cache*). **Curve25519** và **curve448** là các đường cong Montgomery và do đó có phiên bản Edwards tương đương. Đường cong Edwards hỗ trợ cài đặt hiệu quả các phép tác động nhóm đường cong elliptic. Cụ thể, đường cong Edwards ứng với số nguyên tố p thỏa $p \equiv 3 \pmod 4$, và đường cong Edwards xoắn ứng với số nguyên tố p, $p \equiv 1 \pmod 4$.

Curve25519

Cho mức bảo mật 128-bit, số nguyên tố $p = 2^{255} – 19$ nên được chọn vì cho hiệu suất cao trên nhiều kiến trúc. Số nguyên tố p này thỏa $p \equiv 1 \pmod 4$.

[6] https://datatracker.ietf.org/doc/html/rfc7748

Curve25519

Cho mức bảo mật 128-bit, số nguyên tố $p = 2^{255} - 19$ nên được chọn vì cho hiệu suất cao trên nhiều kiến trúc. Số nguyên tố p này thỏa $p \equiv 1 \pmod 4$.

Curve25519 và giao thức thiết lập khóa – **key exchange protocol**

Curve25519 sử dụng **hàm X25519**[1] cho giao thức Elliptic Curve Diffie-Hellman (**ECDH**):

– Alice sinh ngẫu nhiên 32 byte a[0] … a[31] và truyền K_A = X25519(a, 9) tới Bob, trong đó 9 là tọa độ u của điểm sinh và được mã hóa dưới dạng byte có giá trị 9, sau đó là 31 byte 0.

– Tương tự, Bob sinh ngẫu nhiên 32 byte b[0] … b[31], tính K_B = X25519(b, 9) và truyền cho Alice.

– Alice tính X25519(a, K_B) và Bob tính X25519(b, K_A).

Cả Allice và Bob hiện có chung K = X25519(a, X25519(b, 9)) = X25519(b, X25519(a, 9)) như một bí mật được chia sẻ.

Curve448

Cho với mức bảo mật 224-bit, số nguyên tố $p = 2^{448} - 2^{224} - 1$ được đề xuất để thực hiện trên nhiều kiến trúc. Số nguyên tố p có tính chất $p \equiv 3 \pmod 4$.

Curve448 sử dụng **hàm X448**[7] trong giao thức ECDH tương tự hàm X25519 của curve25519. Điểm khác biệt duy nhất là Alice và Bob tạo 56 bytes ngẫu nhiên (thay vì 32) và tính K_A = X448(a, 5) hoặc K_B = X448(b, 5), trong đó 5 là tọa độ u của điểm sinh và được mã hóa dưới dạng byte có giá trị 5, tất cả 55 byte theo sau có giá trị là 0.

MÃ ĐỒNG CẤU

Khái niệm đồng cấu

Một hàm *f* xác định trên D được gọi là đồng cấu (*homomorpic*) theo phép toán $\oplus$ nếu

$$f(x \oplus y) = f(x) \oplus f(y).$$

Khái niệm mã đồng cấu (*homomorphic encryption*) cũng được định nghĩa tương tự.

Hệ mã $E_{k \in K}: \mathcal{M} \to C$ được gọi là mã đồng cấu theo phép toán $\otimes$ nếu $E_k(m_1) \otimes E_k(m_2) = E_k(m_1 \otimes m_2)$.

Mã đồng cấu được áp dụng cho những ứng dụng ở đó cần phải thực hiện một số tính toán trên bản rõ mà không phải giải mã các bản mã nhận được. Chẳng hạn các hệ hợp tác tính toán đảm bảo tính bí mật thông tin, hay các hệ xác minh mà không phải cung cấp thông tin bản rõ có thể sử dụng mã đồng cấu để hiện thực.

[7] https://datatracker.ietf.org/doc/html/rfc7748

Các hệ mã đồng cấu phổ biến

Hệ mã đồng cấu RSA

RSA là hệ mã đồng cấu theo phép nhân modulo.

Thực vậy, giả sử $c_1 = RSA_{e,n}(m_1)$ và $c_2 = RSA_{e,n}(m_2)$ lần lượt là bản mã RSA của các thông điệp rõ m_1 và m_2 với khóa công khai e, n. Ta có

$c_1 \equiv m_1^e \ (mod\ n); c_2 \equiv m_2^e \ (mod\ n)$, thì

$c_1 * c_2 \ \equiv m_1^e * m_2^e \ \equiv (m_1 * m_2)^e \ (mod\ n)$,

Hay $c_1 * c_2$ chính là bản mã của thông điệp $m = m_1 * m_2$.

Hệ mã đồng cấu ElGamal

ElGamal là hệ đồng cấu theo phép nhân modulo.

Thực vậy, giả sử $C_1 = ElGamal_{e,g,p}(m_1)$ và $C_2 = ElGamal_{e,g,p}(m_2)$ lần lượt là bản mã ElGamal của các thông điệp rõ m_1 và m_2 với khóa công khai e, g, p. Ta có

$C_1 = (g^{x_1} \ mod\ p, m * e^{x_1} \ mod\ p); C_2 = (g^{x_2} \ mod\ p, m_2 * e^{x_2} \ mod\ p)$.

Đặt

$C = (g^{x_1+x_2} \ mod\ p, (m_1 * m_2) * e^{x_1+x_2} \ mod\ p)$.

Thì C chính là bản mã ElGamal của $m = m_1 * m_2$.

Một cách tổng quát,

ElGamal-like là mã đồng cấu.

Hệ mã đồng cấu Paillier

Bên cạnh 2 hệ mã công khai RSA và ElGamal-like, ta học thêm hệ mã công khai, cũng có tính đồng cấu, cũng được sử dụng phổ biến trong bảo mật thông tin, mã Paillier (*Paillier encryption*).

Hệ mã Paillier, được phát minh và đặt theo tên của Pascal Paillier vào năm 1999, là một thuật toán mã bất đối xứng xác suất mà độ bảo mật dựa trên bài toán tính lượng thứ n được cho là bài toán khó. Giả định về độ dư tổng hợp quyết định là giả thuyết về tính khó sửa mà hệ thống mật mã này dựa vào.

Lược đồ mã hóa Paillier được mô tả như sau.

Sinh khóa – **KeyGen**

(1) Chọn ngẫu nhiên 2 số nguyên tố lớn phân biệt p và q thỏa

gcd(pq, (p – 1)(q – 1)) = 1.

(2) Tính n = pq và λ = lcm(p – 1, q – 1), với lcm (*least common multiple*) là hàm tính bội số chung nhỏ nhất của 2 số nguyên dương.

(3) Chọn ngẫu nhiên số $g \in \mathbb{Z}_{n^2}^*$, với $\mathbb{Z}_{n^2}^*$ là tập các số khả nghịch (mod n^2).

(4) Kiểm tra đẳng thức $\mu = L\left(g^{\lambda} \bmod n^2\right)^{-1}$ với L(x) ký hiệu số nguyên không âm v lớn nhất sao cho (x – 1) ≥ vn.

Khóa công khai sẽ là e = (n, g), và khóa cá nhân là (λ, μ).

Lưu ý: nếu p và q có chiều dài bit như nhau, đơn giản ta chỉ cần chọn các giá trị g = n + 1, λ = ϕ(n) = (p – 1)(q – 1), và $\mu \equiv \phi(n)^{-1} \pmod p$.

Mã hóa – **Encryption**

Để mã hóa thông điệp m, 0 ≤ m < n.

(1) Chọn ngâu nhiên số nguyên r, 0 < r < n nguyên tố cùng nhau với n, gcd(n, r) = 1.

(2) Tính $c = g^m * r^n \bmod n^2$.

c chính là bản mã của m: c = $\text{Paillier}_{n,g}(m)$.

Giải mã – **Decryption**

Để giải mã bản mã $c \in \mathbb{Z}_{n^2}^*$.

(1) Tính $m = L\left(c^{\lambda} \bmod n^2\right) * \mu \bmod n$.

Thì m chính là thông điệp rõ của bản mã c.

Các tính chất đồng cấu của Paillier – **homomorphic properties**

Mã Paillier là mã đồng cấu với nhiều tính đồng cấu.

Thực vậy, ký hiệu D là hàm giải mã và E là hàm mã hóa.

<u>Đồng cấu theo phép cộng</u>

- Tích của 2 bản mã được giải mã thành tổng của 2 thông điệp.

$D(E(m_1, r_1) * E(m_2, r_2)) \bmod n^2) = m_1 + m_2 \bmod n$.

- Tích của bản mã của thông điệp này với lũy thừa theo g của thông điệp khác được giải mã thành tổng của 2 thông điệp.

$D(E(m_1, r_1) * g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n$.

<u>Đồng cấu theo phép nhân</u>

- Lũy thừa bản mã của thông điệp này với lũy thừa theo g của thông điệp khác được giải mã thành tích của 2 thông điệp.

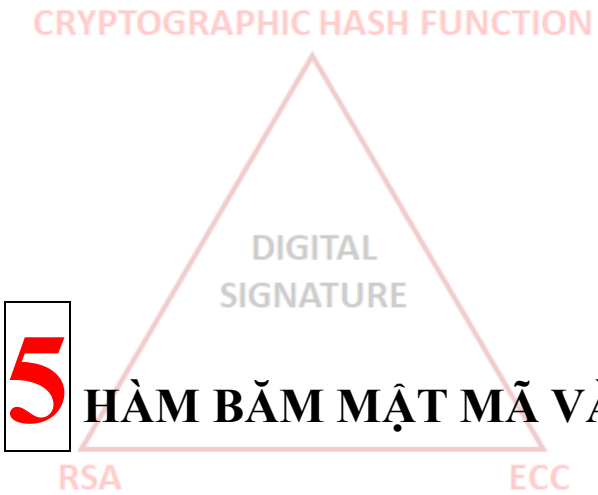
$D(E(m_1, r_1)^{m_2} \bmod n^2) = m_1 * m_2 \bmod n$,

$D(E(m_2, r_2)^{m_1} \bmod n^2) = m_1 * m_2 \bmod n.$

Tổng quát hơn, một bản mã được nâng lũy thừa k sẽ giải mã thành tích của thông điệp và hằng số,

$D(E(m_1, r)^k \bmod n^2) = k * m \bmod n.$

Tuy nhiên, với mã hóa Paillier của hai thông điệp, không có cách nào để tính bản mã của tích 2 thông điệp này mà không biết khóa cá nhân.



5 HÀM BĂM MẬT MÃ VÀ CHỮ KÝ SỐ

HÀM BĂM MẬT MÃ

Định nghĩa hàm băm

Như ta đã biết, hệ mã có thể được thiết kế là hệ bảo mật thông tin được trao đổi, mã đối xứng và mã bất đối xứng là các hệ mã dạng này; hay là hệ được thiết kế chỉ nhằm che giấu và bảo vệ tính toàn vẹn dữ liệu, hàm băm mật mã là dạng hệ mã này.

Một hàm băm mặt mã (*cryptographic hash function*), mà từ nay ta sẽ gọi tắt là hàm băm (*hash function*) là hệ mã được thiết kế để nhận vào một văn bản có chiều dài bit bất kỳ và trả về một giá trị, gọi là trị băm (hash value), là một chuỗi bit có chiều dài cố định sao cho (**i**) từ giá trị băm không thể suy ngược lại được văn bản đầu vào, hơn nữa (**ii**) còn cần có tính kháng đụng độ cao, nghĩa là hai đầu vào khác nhau, dù chỉ khác rất ít bit, thì hai trị băm tương ứng của chúng cũng khác nhau. Một cách hình thức, hàm băm được định nghĩa như sau:

H: $\{0, 1\}^* \rightarrow \{0, 1\}^n$, có các tính chất

(**1**) Kháng tiền ảnh (*pre-image resistance*). Cho x, tính y = H(x) là dễ, nhưng cho y = H(x), tính x là không thể.

(**2**) Kháng tiền ảnh phụ (*second pre-image resistance*). Cho z = H(x), rất khó để tìm ra một y khác x mà có cùng trị băm z = H(y).

(**3**) Kháng đụng độ (*collision resistance*). Rất khó tìm được 2 đầu vào x, y khác nhau mà có cùng trị băm.

Về các tính chất hàm băm

Tính kháng tiền ảnh nhằm chống lại những tấn công, ở đó, tin tặc chỉ có trị băm và muốn phục hồi lại bản gốc. Với tính chất này, hàm băm còn được gọi là hàm 1-chiều (1-*way*

function) và thường được sử dụng để bảo vệ mật khẩu trong các hệ thống có yêu cầu chứng thực người dùng.

Tính kháng đụng độ đảm bảo không thể tìm được 2 văn bản khác nhau có cùng trị băm. Tính chất này khiến việc thiết kế hàm băm mật mã trở nên khó khăn. Thực vậy, theo định nghĩa, bản chất của hàm băm là một hàm nén thông tin (không thể phục hồi, theo tính kháng tiền ảnh), vì thế, nguyên lý chuồng bồ câu (*nguyên lý Dirichlet*) khẳng định rằng sẽ có các văn bản khác nhau có cùng trị băm, nghĩa là **đụng độ là luôn tồn tại**. Hàm băm có tính chất kháng đụng độ mạnh có thể được dùng để tạo định danh (*identification*) phân biệt các văn bản với nhau. Tính kháng đụng độ làm cho kẻ tấn công hệ thống chứng thực khó tìm ra được 2 mật khẩu có cùng trị băm.

Tính kháng tiền ảnh phụ, gọi là '*phụ* (*second*)' vì nó không quan trọng bằng tính 1-chiều của hàm băm. Hơn nữa, một hàm băm mạnh, có tính kháng đụng độ mạnh thì cũng có tính kháng tiền ảnh phụ. Như vậy, có thể xem xét bỏ qua tính kháng tiền ảnh phụ để giảm bớt khó khăn khi thiết kế hàm băm cho những ứng dụng kiểu như chống lại những tấn công gian lận, ở đó, kẻ tấn công muốn thay thế văn bản gốc bằng một văn bản khác có cùng trị băm.

Thiết kế hàm băm

Nguyên lý cơ bản là thiết kế một hàm băm mà trị băm là các giá trị ngẫu nhiên chỉ phụ thuộc vào văn bản đầu vào.

Từ nguyên lý trên, thiết kế một hàm băm sử dụng một hàm trộn (*mixed algorithm*) M mà biến đổi 1 chuỗi n-bit thành 1 chuỗi n-bit, và sử dụng M như "*trái tim*" hàm băm H để trộn 2 chuỗi cùng chiều dài bit. Kích thước chuỗi tùy thuộc ứng dụng, thường là 128 hay 256 bit, và đó cùng là chiều dài bit đầu ra của hàm băm. Để băm văn bản D, thuật toán băm H thi hành các bước sau:

Algorithm H(D)

(**Bước 1**) Chia văn bản D thành các khối cùng chiều dài n bit. Chuỗi cuối cùng có thể dán (*padding*) thêm một số bit, theo quy ước trước, đề được một chuỗi n bit.

. $D = B_1 \| B_2 \| \dots \| B_k$.

Trong đó || ký hiệu phép ghép chuỗi.

(**Bước 2**) Thi hành hàm trộn M liên tục theo cách chuỗi sau sẽ trộn với kết quả biến đổi của chuỗi trước qua M.

. H_0 là chuỗi khởi tạo, còn gọi là mầm (*seed*), tính

. $H_i = H_{i-1} \oplus M(H_i)$, i = 1, 2,…, k

(phép trộn $\oplus$ thường được dùng là phép XoR bit)

(**Bước 3**) Kết quả trả về là chuỗi cuối cùng.

. return H_k.

Hàm băm có khóa

Hàm băm có khóa (*keyed hash function*) là cách sử dụng biến thể của hàm băm tổng quát H(D) bằng cách thêm một số thông tin mật k, gọi là khóa (*key*), không thuộc văn bản D, để chỉ người có khóa mới tạo lại được trị băm đúng với trị băm đã tạo với cùng khóa trên D. Chẳng hạn, có thể dùng khóa (padding nếu cần) làm mầm H_0. Hay đơn giản là thêm vào cuối văn bản D.

Hàm băm có khóa có thể được dùng tạo mã chứng thực thông điệp – MAC (Message Authentication Code).

Cài đặt hàm băm

Cài đặt dựa trên hàm mã hóa

Bản chất của các hệ mã, đối xứng cũng như như bất đối xứng, là các hàm ngẫu nhiên, nên có thể sử dụng một hệ mã E có cùng chiều dài đầu ra của hàm băm như hàm trộn M. Ví dụ, có thể sử dụng hệ mã ECC làm hàm trộn M để tạo hàm băm 256 bit.

Vì tốc độ là yếu tố quan trọng đối với hàm băm nên người ta có xu hướng tạo hàm băm bằng cách sử dụng hàm trộn M với các cách trộn khác nhau, thay vì dựa vào các bài toán khó.

Các hàm băm mật mã thông dụng

Các hàm băm được sử dụng rộng rãi nhất hiện nay:

MD5 – **Message Digest 5**

Được giới thiệu bởi Rivest, 1 trong 3 tác giả của RSA, năm 1992. MD5 được sử dụng phổ biến để bảo mật mật khẩu.

SHA – **Secure Hash Algorithm**

Là thuật toán băm chuẩn được sử dụng trong nhiều ứng dụng, đặc biệt là trong blockchain. SHA có 3 phiên bản với kích thước trị băm khác nhau:

– SHA-1. Hàm băm tạo ra trị băm theo nguyên tắc tương tự nguyên tắc Ronald L. Rivest sử dụng khi thiết kế thuật toán băm MD2, MD4 và MD5, nhưng tạo ra trị băm cho chiều dài bit lớn hơn (160 bit so với 128 bit).

– SHA-2. Gồm nhiều hàm băm do Cơ quan An ninh Quốc gia Hoa Kỳ (USA National Secure Agency) thiết kế và công bố năm 2001, bằng cách sử dụng cấu trúc Merkle–Damgård, từ một hàm nén một chiều theo cấu trúc Davies–Meyer của một mật mã khối chuyên dụng. Sau đó, được Viện Tiêu Chuẩn Hoa Kỳ (National Institute of Standards and Technology) chọn là hàm băm chuẩn với trị băm dài 256 bit.

– SHA-3. Là thành viên mới nhất của nhóm hàm băm chuẩn an toàn, được NIST công bố ngày 5 tháng 8 năm 2015. Mặc dù tên là SHA, nhưng SHA-3 có sự khác biệt về cấu trúc so với SHA-1 và SHA-2 là các SHA theo cấu trúc MD5.

SHA-3 là một hệ mã mở rộng của Keccak, được thiết kế bởi Guido Bertoni, Joan Daemen, Michaël Peeters và Gilles Van Assche, dựa trên hàm RadioGatún. Các tác giả của Keccak

đã đề xuất các cách sử dụng bổ sung cho hàm này, bao gồm mã dòng (*stream cipher*), mã xác thực (*authentication*), và lược đồ băm "*cây*" để băm nhanh hơn trên một số kiến trúc.

Cài đặt sử dụng lại các hàm băm đã biết

Với kết quả sau, ta có thể thiết kế một hàm băm mới dựa trên sự kết hợp của các hàm băm đã có.

Mệnh đề (hợp các hàm băm). Nếu

$H1: \{0,1\}^* \rightarrow \{0,1\}^{n_1}$, và

$H2: \{0,1\}^* \rightarrow \{0,1\}^{n_2}$,

là 2 hàm băm mật mã, thì

$H1 \circ H2: \{0,1\}^* \rightarrow \{0,1\}^{n_1}$, và

$H2 \circ H1: \{0,1\}^* \rightarrow \{0,1\}^{n_2}$,

cũng là các hàm băm mật mã.

SỐ NGẪU NHIÊN

Sinh được một số ngẫu nhiên là bài toán quan trọng trong thực tế, đặc biệt trong mật mã, như ta đã thấy trong các hệ mã đã học, ở đó, chẳng hạn như hệ mã ElGamal, nếu hàm sinh số ngẫu nhiên có tính ngẫu nhiên yếu, hệ mã có thể bị phá vỡ.

Lý tưởng nhất là ta có một thiết bị tạo ra dãy nhị phân ngẫu nhiên hoàn toàn gồm các số 0 và 1. Thiết bị như thế là tồn tại trong thực tế, chẳng hạn bộ đếm Geiger (*Geiger counter*). Tuy nhiên, trên thực tế, việc xây dựng bộ đếm Geiger cho máy tính lại rất tốn kém. Nên thực tế vẫn chỉ có thể tạo được các số giả ngẫu nhiên (*pseudo-random number*).

Trình sinh số giả ngẫu nhiên

Trình sinh số giả ngẫu nhiên – **PRGN** (*Pseudo-Random Number Generator*) là một hàm hai biến

$F: \mathbb{N} \times \mathbb{N} \rightarrow \{0,1\}^n$,

F(c, s) nhận vào một số nguyên không âm và một số s (mầm – *seed*), trả về bit thứ c. Quá trình có thể được mô tả như sau:

– Để bắt đầu, chọn giá trị mầm S (thực hoặc ngẫu nhiên nhất có thể).

– Thực thi liên tiếp hàm F để tính R0 = F(0, S), R1 = F(1, S), …

– Kết quả trả về là chuỗi bit giả ngẫu nhiên R0||R1||…

Trình sinh số giả ngẫu nhiên an toàn mật mã

PRNG an toàn mật mã (*Cryptographically secure PRNG*) nếu:

- Kẻ thám mã Ever biết k bit đầu tiên của chuỗi bit ngẫu nhiên, Ever cũng sẽ không có lợi thế trên hơn 50% trong việc dự đoán liệu bit tiếp theo sẽ là 0 hay 1.
- Giả sử rằng Ever có thể tìm ra các giá trị R_t, R_{t+1}, … thì điều đó cũng sẽ không giúp Ever xác định được giá trị của t bit R_0, …, R_{t-1} trước đó.

Cài đặt trình sinh số giả ngẫu nhiên

Trình sinh số giả ngẫu nhiên **PRGN** có thể được thiết kế bằng hàm băm mật mã (*hash function*) H bằng cách chọn giá trị ngẫu nhiên ban đầu S và thực thi: R_i=H (i||S).

Cũng có thể thiết kế **PRGN** từ một hệ mã, đối xứng hay bất đối xứng E_K như RSA hay AES chẳng hạn: R = E_K(C XoR S), trong đó C = E_K(D) với D là thời gian máy tính.

CHỮ KÝ SỐ

Ký dùng hệ mã công khai

Với các hệ mã công khai có tính đối xứng như RSA, bằng cách sử dụng khóa cá nhân d trong quá trình mã hóa (thay vì khóa công khai e), Allice sẽ thực hiện 'mã hóa' có thể tạo ra 'bản mã' S cho bản rõ M nhập vào. Chỉ người có khóa cá nhân d mới có thể tạo ra được S tương ứng với M, hơn nữa, mọi người đều có thể đọc được S bằng cách sử dụng khóa công khai (mà ai cũng có thể biết) e và n để 'giải mã'. Trong trường hợp thực hiện ngược như vậy,

. S được gọi là chữ ký (*signature*) của Allice trên thông điệp M, và quá trình mã hóa bây giờ được gọi là quá trình ký (*sign*).

Sign: S = RSA(M, d, n) = M^d mod n.

. Bob dùng khóa công khai thực hiện giải mã S, nếu "*đọc hiểu*" kết quả thì Bob tin chắc rằng Allice đã ký trên M. Trong thực tiễn, Bob biết M, và so sánh với kết quả giải mã. Tiến trình đó được gọi là xác minh (*verification*).

Verification: If RSA(S, e, n) ≡ M: return TRUE else: return FALSE.

Trên nguyên tắc, mọi hệ mã công khai đều có thể được dùng để tạo ra chữ ký bằng cách sử dụng khóa cá nhân trong tiến trình ký và dùng khóa công khai để xác minh. Tuy nhiên, những hệ mã không có tính đối xứng như RSA, chẳng hạn ElGamal, thì quá trình ký và xác minh sẽ phức tạp hơn.

Chữ ký ElGamal-like

Giả sử d là khóa cá nhân và e = g^d mod p là khóa công khai của Allice. Allice công khai (e, g, p), trong đó g là phần tử sinh của $\mathbb{Z}_p^+$ và p là số nguyên tố lớn. Để ký trên thông điệp M, Allice thực hiện quá trình sign.

Sign

. $k \overset{\$}{\leftarrow} PRNG(p)$ #sinh ngẫu nhiên số k nguyên tố cùng nhau với p – 1.

. $r \leftarrow g^k (mod\ p)$.

. $K \leftarrow k^{-1}\ (mod\ p)$ #tính nghịch đảo của k theo (p – 1).

. $s \leftarrow K * (M - d * r) (mod\ p - 1)$.

. Return S = (r, s) #cặp (r, s) là chữ ký của Allice trên M.

Bob muốn xác minh S = (r, s) có phải chữ ký của Allice hay không. Bob sẽ sử dụng các thông tin công khai của Allice, gồm (e, g, p) và thực hiện tiến trình verification.

Verification

. $v_1 \leftarrow (y^r * r^s) mod\ p.$

. $v_2 \leftarrow g^M mod\ p.$

. If $v_1 = v_2$: Return TRUE else: Return FALSE.

Nếu đẳng thức cuối cùng là đúng, $v_1 = v_2$, Bob chấp nhận S = (r, s) là chữ ký của Allice trên M. Thực vậy,

$s = K*(M - dr) \equiv k^{-1}*(M - dr) \pmod{(p - 1)}$.

$ks = (M - dr) \bmod (p - 1)$.

$M = (dr + ks) \bmod (p - 1)$.

Suy ra,

$v_2 \equiv g^M \equiv g^{dr + ks} \equiv (e^r * r^s) \equiv v_1 \pmod p$.

Chữ ký số trên đường cong elliptic

Thuật toán sinh chữ ký số dựa trên đường cong Elliptic – **ECDSA** (*Elliptic Curve Digital Signature Algorithm*) là thuật toán tạo chữ kí số cho văn bản, giúp chống giả mạo cũng như làm sai lệch dữ liệu, qua đó có thể xây dựng một phương pháp xác thực mà không ảnh hưởng đến tính bảo mật của văn bản gốc. ECDSA được ứng dụng trong blockchain và nhiều ứng dụng cần bảo mật và đảm bảo tính riêng tư dữ liệu.

Sinh khóa – KeyGen

(1) Chọn điểm sinh G của đường cong trên $\mathbb{Z}_p$ với p là số nguyên tố.
(2) Chọn khóa cá nhân d.
(3) Tính và công bố khóa công khai e = dG.

Ký – signature

Để ký trên văn bản D với khóa cá nhân d và p là số nguyên tố.

(1) Trước hết, chọn một số ngẫu nhiên k.
(2) Tính P bằng cách nhân k với điểm sinh G:
P = kG = (r, y).
(3) Tính trị băm của văn bản D:
z = H(D). (trong các mạng chuỗi khối H được dùng là hàm băm SHA-2).
(4) Tính
$s \equiv k^{-1}(z + d * r) \pmod p$.
(5) Chữ ký trên D là cặp S = (r, s).

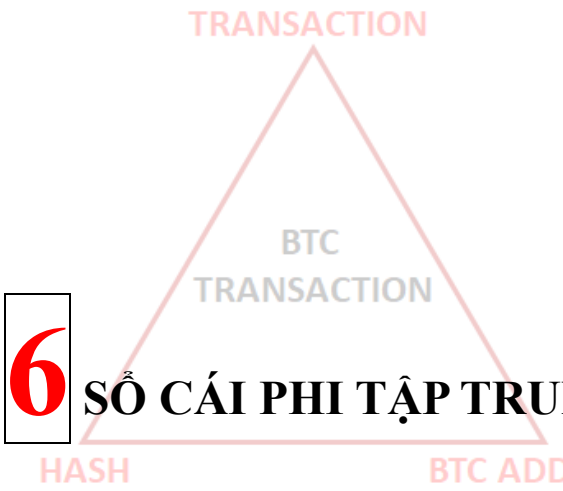
Xác minh chữ ký – **verification**

Để xác minh chữ ký, sử dụng khóa công khai e.

(1) Tính $P = (s^{-1} * z)G + (s^{-1} * r)e = (x, y)$.

(2) Nếu r == x: Return TRUE, else: Return FALSE.

Tọa độ x của điểm P = (x, y) nếu bằng r thì chữ ký (r, s) được chấp nhận. Thực vậy,
$P = (s^{-1} * z)G + (s^{-1} * r)e = (x, y)$
$= (s^{-1} * z)G + (s^{-1} * r) * dG$
$= (z + r * d)*s^{-1} G$
Mà
$s = k^{-1} * (z + d * r)$,
nên
$P = (z + r * d) * (z + r * d)^{-1} * (k^{-1})^{-1} G = kG.$



6 SỔ CÁI PHI TẬP TRUNG BITCOIN

CHUỖI KHỐI BITCOIN

Sổ cái bitcoin

Bitcoin có thể coi là cài đặt sổ cái phi tập trung thành công nhất cho đến hiện nay. Bitcoin được Satoshi thiết kế năm 2007 từ ý tưởng sổ cái phi tập trung, ở đó mọi thành viên có thể giao dịch mà không phải tin tưởng ai. Và 20/10/2008, Bitcoin được biết đến như một giao thức thanh toán ngang hàng (*peer-to-peer payment protocol*) của nhân vật bí ẩn Satoshi Nakamoto, và 9/01/2009 bắt đầu được sử dụng với trang – block khởi thủy (*genesis block*) làm quy chiếu cho mọi khối sau này. Giao dịch đầu tiên trị giá 10 Bitcoin do Satochi Nakamoto gửi cho Hal Finney, một nhà mật mã học, vào 12/01/2009. Và 5/10/2009 Bitcoin được ấn định trên sàn giao dịch (*exchanges*) với mức khởi điểm $1 = 1309.03 BTC (viết tắt của Bitcoin) hay 1 BTC = 0.00076 USD là chi phí tiêu thụ năng lượng cho một máy tính xác thực giao dịch thành công, gọi là máy đào (*minner*). Sau đó, khoảng giữa năm 2010, Bitcoin được xem như sổ cái phi tập trung (decentralized ledger) cho đến nay, và trở nên ngày càng phổ biến sau khi BTC được dùng để mua 2 chiếc bánh pizza bằng 1000 BTC, tương đương $25 ngày đó. Tiếc thay, sự thành công bày làm mọi người hiểu sổ cái phi tập trung – blockchain như một loại tiền tệ, và thuật ngữ tiền mã hóa (*cryptocurrency*) được dùng chính thức.

Nhấn mạng rằng Bitcoin hay BTC là một ứng dụng của blockchain cài đặt sổ cái phi-tập trung.

Mạng Bitcoin

Tích hợp sẵn trong giao thức Bitcoin là công nghệ sổ cái (ledger or blockchain technology). Mọi giao dịch đều được cập nhật trên hệ thống lưu trữ máy tính ngang hang (P2P – peerr to peer network), trong CSDL dạng blockchain, một sổ cái ghi lại số dư của mỗi tài khoản và lịch sử tất cả tài khoản tham gia giao dịch trước đó.

Khác biệt của mạng BTC với các giao dịch ngân hàng trực tuyến là mạng BTC không cần đến bên trung gian với CSDL tập trung (centralized database) mà ta phải tin tưởng để xác minh các giao dịch nhằm chống giao dịch lừa đảo. Công nghệ chuỗi khối giải quyết vấn đề

xài thông tin một khối (tiền mã BTC) nhiều lần, gọi là vấn đề tiêu lại (double spent) mà không cần bên trung gian thứ 3 đáng tin cậy:

Mọi thứ đều được ghi lại trên sổ cái phi-tập trung và không thể sửa đổi bất kỳ thông tin nào được ghi vào sổ cái.

Các tính chất của 'tiền tệ' BTC

Bitcoin, một loại tiền mật mã phi tập trung, an toàn, công khai, bình đẳng và ẩn danh.

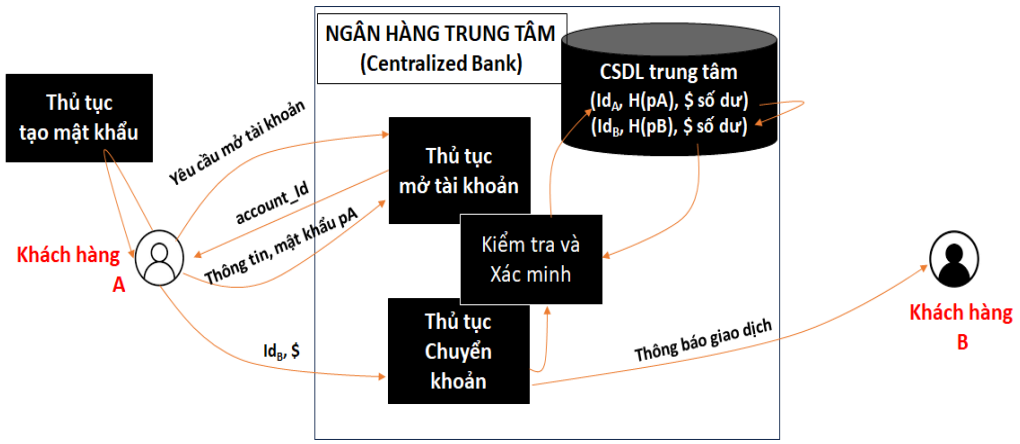
- ***Phi tập trung***: Không có cá nhân hay tổ chức nào sở hữu hoặc kiểm soát hoàn toàn được mạng bitcoin.
- ***Bảo mật***: Mạng bitcoin được xây dựng trên các thuật toán bảo mật hiện đại.
- ***Công khai***: Mạng bitcoin duy trì một sổ cái (ledger) ghi lại tất cả các giao dịch của người dùng tại mỗi nút mạng miner.
- ***Bình đẳng***: Bất kỳ ai hoặc bất kỳ tổ chức nào cũng có thể tham gia giao dịch trên mạng bitcoin.
- ***Ẩn danh***: Khi tham gia mạng bitcoin, không cần khai báo hay đăng ký thông tin cá nhân với bất kỳ tổ chức nào.

MẠNG BITCOIN – NGÂN HÀNG PHI TẬP TRUNG

Trước hết, cần lưu ý là blockchain không phải là tiền mật mã (*cryptocurrency*), nhưng do mạng bitcoin (**BTC** *network*) sử dụng công nghệ blockchain xây dựng ứng dụng cho tài chánh, mà tập trung chính vào giao dịch (**Tx** – *transaction*) nên ta mượn một số thuật ngữ trong lãnh vực ngân hàng để giải thích các thành phần cũng như nguyên lý hoạt động của mạng BTC.

Ngân hàng tập trung

Bỏ qua các khái niệm liên quan đến công nghệ cũng như dịch vụ tài chánh-ngân hàng, chúng ta chỉ tập trung vào quy trình giao dịch (chuyển khoản) trực tuyến (*online transaction*) của hầu hết các ngân hàng đang sử dụng, qua đó chúng tôi giải thích các khái niệm quan trọng liên quan đến giao dịch. Hình dưới minh họa quy trình giao dịch tổng quát, từ khởi đầu một tài khoản được mở cho đến chuyển khoản.



Tóm tắt quy trình

Để sử dụng các dịch vụ trực tuyến (*online service*) của một Ngân Hàng, trước hết, khách hàng phải tiến hành **mở tài khoản** (thường được thực hiện trực tiếp tại ngân hàng) để được cấp một số tài khoản (**ID_{Bank}** – *bank account*) và một mật khẩu mặc định (*default password*). Để sử dụng tài khoản được cấp này, người dùng phải kích hoạt (**activate**) bằng cách tạo và đổi mật khẩu qua thủ tục **tạo mật khẩu**. Các thông tin về khách hàng như: số tài khoản $ID_{Khách\ hàng}$, trị băm mật khẩu (H(password)), số dư $,… được lưu trong CSDL trung tâm của ngân hàng.

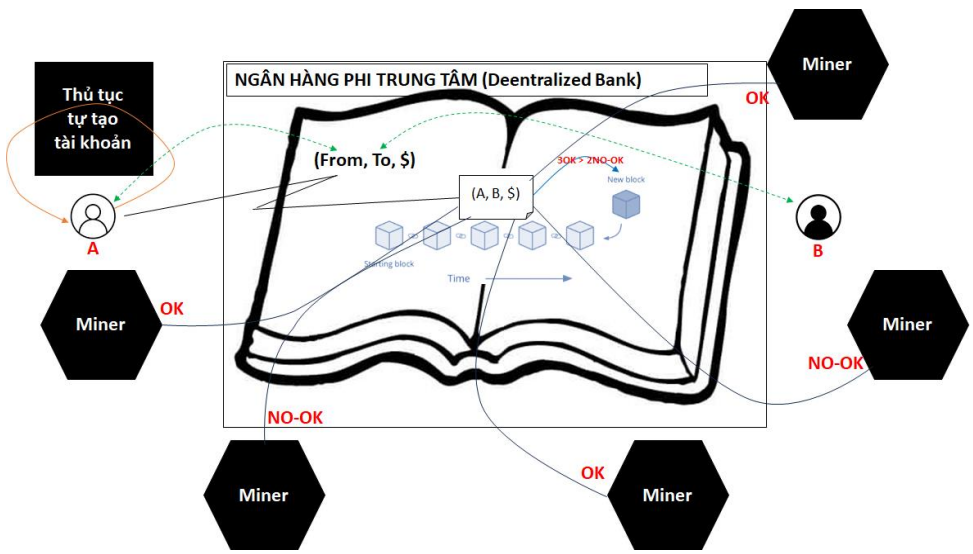
Để thực hiện giao dịch chuyển khoản (*transaction*), khách hàng cung cấp cho thủ tục **chuyển khoản** số tài khoản người nhận, số tiền cần chuyển. Nếu thủ tục **kiểm tra** xác minh thành công thì thực hiện chuyển khoản, cập nhật CSDL và thông báo cho người nhận (nếu cần); nếu không, không có giao dịch nào được thực hiện.

Hai khái niệm chính

Hoạt động của ngân hàng như mô tả ở trên xoay quanh 2 khái niệm trọng tâm:
. **Tài khoản**. Gồm 1 số tài khoản ID_{Bank}, 1 mật khẩu bí mật để chứng minh người đang thao tác trên tài khoản ID_{Bank} là chủ tài khoản đó, và số dư đang có trong tài khoản.
. **Giao dịch**. Là quy trình thủ tục để một khách hàng có thể chuyển tiền cho khách hàng khác. Các thông tin cơ bản người chuyển phải cung cấp cho dịch vụ **chuyển khoản** gồm số tài khoản người gửi (**from**), số tài khoản người nhận (**to**) và số tiền (**$**) được chuyển.
Hai khái niệm này sẽ được mô phỏng lại trong mô hình ngân hàng phi tập trung.

Ngân hàng phi tập trung

Các khái niệm và hoạt động của ngân hàng phi tập trung là tương tự trong ngân hàng tập trung, chỉ khác ở điểm cốt lõi: một bên không có và một bên có bên trung gian thứ 3 đáng tin cậy. Nếu như ở ngân hàng tập trung, vai trò chính nằm ở bên thứ 3 đáng tin chính là ngân hàng, thậm chí, khách hàng cũng chỉ là phụ ngay cả tiền trong ngân hàng ở tài khoản của khách hàng thì có khi muốn rút cũng không thể rút nếu ngân hàng không cho. Hình dưới mô phỏng quy trình hoạt động của một ngân hàng phi tập trung, chủ yếu xây dựng xoay



quanh 2 khái niệm số tài khoản ngân hàng, ở đây ta sẽ sử dụng thuật ngữ định danh người chủ tài khoản (*Identification*), và khái niệm giao dịch – **Tx** (*transaction*).

Tóm tắt quy trình

Người dùng tự do tham gia mạng BTC. Để tham gia, đơn giản người dùng chỉ tự tạo cho mình tài khoản BTC để chuyển và nhận BTC tới hay từ người khác cũng đã có tài khoản trên mạng BTC. Tài khoản là một định danh (Id) mà là một cặp (**khóa công khai**, **khóa cá nhân**), khóa công khai có thể được sử dụng trực tiếp hoặc có thể được chuyển sang **địa chỉ** (***address***) trên mạng BTC từ khóa công khai; và khóa cá nhân được sử dụng để chứng minh là chủ tài khoản hay chủ địa chỉ trên BTC. Khi ấy, người dùng với thiết bị nối mạng của mình hình thành một nút sử dung (*user node*) trên mạng BTC. User node có thể chỉ phải kích hoạt khi giao dịch.

Để thực hiện một giao dịch chuyển khoản, khác với giao dịch trên ngân hàng tập trung, ở BTC, người dùng tạo và phát tán (broadcast) giao dịch, gồm địa chỉ của mình, địa chỉ nhận, số tiền chuyển, và một số thông tin khác, lên mạng BTC. Trên mạng BTC có sẵn một số nút mạng đặc biệt gọi là nút khai thác (*miner node*), mà lưu toàn bộ CSDL sổ cái – blockchain BTC và phải luôn ở trạng thái kích hoạt. Các miner nodes tập hợp các giao dịch được phát trên mạng BTC, xác minh tính hợp lệ và chấp nhận (OK) hay không (NO-OK). Giao dịch hợp lệ là giao dịch được đa số đồng thuận là đúng đắn, sẽ được 1 miner node nào đó đưa vào block mới và gắn vào blockchain.

Địa chỉ và giao dịch BTC

Cũng như trong ngân hàng tập trung, trong ngân hàng phi tập trung BTC, cũng có khái niệm 'tài khoản' và 'giao dịch'.

– **Tài khoản BTC**: Là cặp (địa chỉ A, khóa cá nhân d), trong đó địa chỉ A được tạo từ khóa công khai e tương ứng với khóa cá nhân d, của một hệ mã công khai. BTC sử dụng mã đường cong elliptic (**ECC**).

– **Giao dịch BTC**: Cũng như giao dịch chuyển khoản trong ngân hàng truyền thống, chỉ khác ở 2 điểm chính (**i**) người dùng tự tạo giao dịch, và (**ii**) giao dịch được các nút khai thác xác minh.

THỦ TỤC TẠO ĐỊA CHỈ CHO MỘT TÀI KHOẢN BTC

Số tài khoản trên BTC được gọi là địa chỉ của tài khoản (*account address*). Địa chỉ một tài khoản cũng là địa chỉ định danh công khai (public *identification*) một nút trên mạng BTC, nút sử dụng (*user node*) cũng như nút khai thác (*miner node*). Địa chỉ BTC là một chuỗi 25 ký tự (*character*) mỗi ký tự 1 byte, được tạo từ khóa công khai của hệ mã đường cong elliptic (**ECC** – *Elliptic Curve Creyprosystem*).

Thủ tục tạo địa chỉ từ khóa công khai như trình bày dưới, trong thủ tục này, 2 hàm băm mật mã được sử dụng gồm:

– **Hàm băm SHA-2**: Hàm băm SHA-2, ký hiệu SHA256, nhận đầu vào là địa chỉ khóa công khai ECC và trả về trị băm là một số 256 bits hay chuỗi 32 bytes.

– **Hàm băm RIPEMD-160**. RIPEMD-160, viết tắt là RIPMD160 (*RIPE message digest* - 160), Trả về trị băm là một chuỗi 160 bits hay 20 bytes.

Thủ tục tạo địa chỉ BTC

Bước 1: Trước hết, sử dụng thủ tục tạo cặp khóa của ECC gồm khóa công khai e và khóa cá nhân d. Khóa công khai e là một điểm P = (x, y) trên nhóm đường cong elliptic (E, +).

Sử dụng kỹ thuật biểu diễn của SSL (*Secure Sockets Layer*), ta ghép thêm 1 byte vào điểm biểu diễn khóa công khai để biểu diễn 1 điểm của ECC bằng 65 bytes, trong đó, byte đầu tiên, tính từ trái qua, được dùng để ký hiệu mã 'kiểu lưu', 32 bytes tiếp theo biểu diễn hoành độ, và 32 bytes cuối biểu diễn tung độ của khóa công khai e:

PXY ← sparse(e) = [1 byte P | 32 bytes X | 32 bytes Y].

Các giá trị của byte P đầu tiên được quy ước như sau:

– Nếu có giá trị thập lục phân là 0x02 hay 0x03, thì chỉ lưu địa chỉ x. Địa chỉ y sẽ được tính từ x theo phương trình đương cong (E).

– Nếu giá trị là 0x04, thì lưu cả x và y.

Bước 2: Chuỗi 65 bytes kết quả của bước 1 được băm thành chuỗi 32 byte.

Z ← SHA256(PXY) = [32 bytes Z]

Bước 3: Chuỗi 32 bytes kết quả băm của SHA-2 lại được băm, bằng hàm băm RIPE MD-160, thành chuỗi 20 bytes.

U ← RIPMD160(Z) = [20 byte U]

Bước 4: Sau đó, kết quả băm U được thêm 1 bytes vào bên trái để mã hóa ý nghĩa của địa chỉ.

V ← [1 byte Q | 20 bytes U].

Các giá trị của byte đầu tiên Q, được quy ước như sau:

Nếu giá trị là 111 hay 196 thì sẽ biểu diễn địa chỉ của mạng thử nghiệm (test net), là mạng BTC để cho các nhà phát triển ứng dụng mạng BTC có thể thử nghiêm các ý tưởng thuật toán mới. Trong đó

- **111** cho biết địa chỉ tạo ra là địa chỉ khóa, còn
- **196** là địa chỉ kịch bản, địa chỉ bắt đầu của một 'chương trình' BTC.

Bước 5: Chuỗi 21 bytes kết quả lại được băm thành giá trị băm SHA-2 gồm 32 bytes. Nhưng chỉ giữ lại 4 bytes đầu tiên bên trái để làm mã kiểm tổng (*checksum*).

CS ← SHA256(V) = [4 bytes CS].

Bước 6: Ghép 4 bytes CS của kết quả với 21 bytes V của bước 5, được địa chỉ trung gian, là một chuỗi 25 bytes hay 200 bits nhị phân.

TIA ← [1 byte T | 20 bytes RIPEMD | 4 bytes SHA-2]

Bước 7: Dãy 200 bits nhị phân gồm toàn giá trị 0, 1 này rất dễ nhầm lẫn khi phải nhập vào và rất khó kiểm tra kết quả nhập đúng hay sai. Vì thế nó được chuyển sang chuỗi các ký tự ascii bằng thủ tục BASE256-2-BASE256. Thủ tục này chuyển chuỗi nhị phân sang dạng ký tự trong tập các ký tự {A,…, Z, a,…, z, 0… 9}\{+, /, O, 0} gồm 58 ký tự. Các ký tự {+, /, O, 0} được bỏ đi để tăng khả năng tránh nhầm lẫn khi nhập từ giấy vào máy tính.

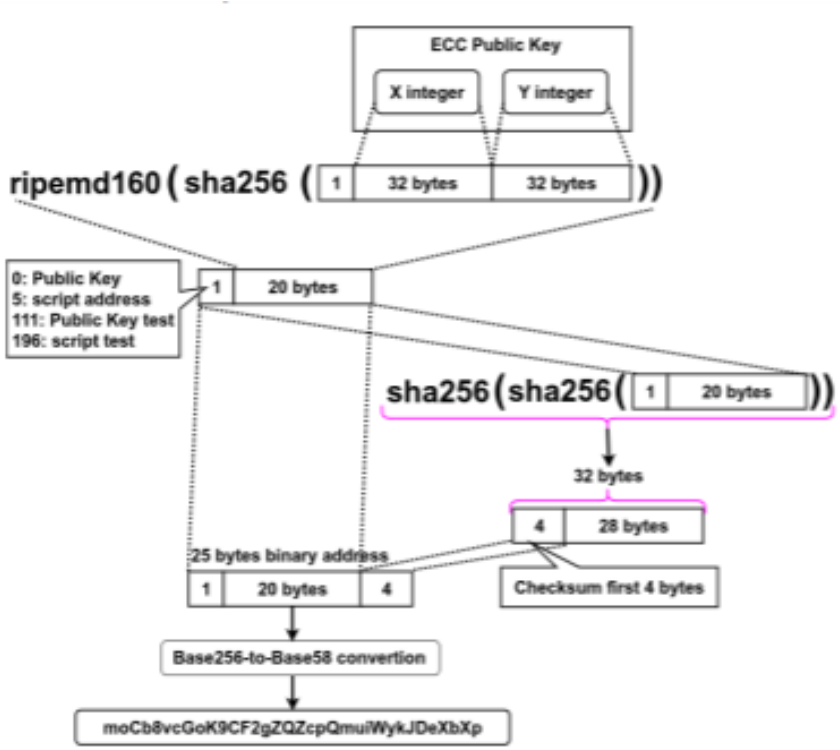
Ta biết rằng, mọi giao dịch (**Tx**) đều có địa chỉ nguồn (**from**) là địa chỉ của người gửi, địa chỉ đích (**to**) là địa chỉ người nhận, và số tiền (**$**) cùng một số thông tin khác. Sai lầm của chuỗi ký tự **from** có thể được phát hiện khi xác minh **Tx**, nhưng chuỗi địa chỉ nhận **to** thì không được dùng trong tiến trình xác minh. Vì thế, một khi **Tx** được ghi lên blockchain thì không thể sửa chữa được nữa. Nghĩa là **$** chuyển từ **from** đến một **to** không đúng ý muốn. Vì thế, thủ tục sử dụng ký tự hiển thị được nhằm hạn chế khả năng mất **$** vô lý này.

Add_BTC ← BASE256-2-BASE256(TIA).

Bước 8: Chuỗi kết quả của bước 6 chính là địa chỉ nút mạng BTC ứng với khóa công khai e ban đầu.

Return Add_BTC.

Hình sau tóm tắt quy trình tạo địa chỉ BIT.



GIAO DỊCH

Bên cạnh khái niệm địa chỉ BTC (*BTC address*), giao dịch (**Tx** – *transaction*) và thực hiện (xác minh) giao dịch là khái niệm chính trong giao dịch bitcoin.

Cấu trúc giao dịch BTC

Giao dịch (**Tx**) là trao đổi BTC và được phát lên mạng BTC và được tập hợp vào trong các block. Một giao dịch thường tham chiếu đến đầu ra của các giao dịch đã xảy ra trước đó như đầu vào (input) của giao dịch mới và chuyển tất cả giá trị BTC đầu vào này sang cho đầu ra (output) mới. Giao dịch BTC không được mã hóa nên có thể duyệt và xem nội dung mọi giao dịch trên sổ cái – blockchain. Một khi giao dịch đã được ghi lên blockchain sau quá trình được xác minh, nội dung giao dịch sẽ không thể thay đổi được nữa.

Đầu ra một giao dịch chuẩn (standard transaction) chỉ đến các địa chỉ và việc sử dụng (tiêu BTC – spending BTC) bất kỳ đầu vào nào sau nay đều cần có chữ ký liên quan, nghĩa là phải có khóa cá nhân, gọi là khóa ký (*signature key*) để có thể tiêu (*spend*) BTC đầu ra đó.

Mọi giao dịch đều hiển thị trong sổ cái – blockchain (dưới dạng thập lục phân – *hexa*) và có thể được xem bằng trình hiển thị giá trị **hex**. Trình duyệt blockchain là một trang web nơi mọi giao dịch có trong blockchain có thể được xem dưới dạng tự nhiên. Như vậy, có thể xem chi tiết các giao dịch đang xảy ra và để xác minh thanh toán. Cấu trúc tổng quát một giao dịch có dạng như mô tả trong hình dưới.

Field	Description	Size
Version no	currently 1	4 bytes
Flag	If present, always 0001, and indicates the presence of witness data	optional 2 byte array
In-counter	positive integer VI = VarInt	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer VI = VarInt	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
Witnesses	A list of witnesses, 1 for each input, omitted if flag above is missing	variable, see Segregated_Witness
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

Giao dịch 1 đầu vào – 1 đầu ra

Hình sau minh họa nội dung một giao dịch BTC với 1 đầu vào và 1 đầu ra.

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

Theo đó, đầu vào giao dịch trong hình này nhập 50 BTC từ đầu ra số 0 trong giao dịch f5d8... Sau đó, đầu ra sẽ gửi 50 BTC đến địa chỉ BTC (được biểu thị ở đây bằng hệ thập lục phân 4043... thay vì cơ sở58). Khi người nhận muốn tiêu số tiền này, anh ta sẽ tham chiếu đầu ra số 0 của giao dịch này làm đầu vào của giao dịch mới của mình.

Đầu vào giao dịch

Cấu trúc đầu vào được mô tả trong hình:

Field	Description	Size
Outpoint hash	The previous transaction that contains the spendable output	32 bytes
Outpoint index	The index within the previous transaction's output array to identify the spendable output	4 bytes
Script length	positive integer VI = VarInt	1 - 9 bytes
Script signature	Information required to spend the output (see below for details)	Variable
Sequence number	if sequence number is < 0xFFFFFFFF: Makes the transaction input Replace-By-Fee	4 bytes

Đầu vào là tham chiếu đến đầu ra của một giao dịch trước đó. Có thể có nhiều đầu vào trong một giao dịch. Tất cả các giá trị đầu vào của giao dịch mới (nghĩa là tổng giá trị BTC của các đầu ra trước đó được tham chiếu bởi đầu vào của giao dịch mới) được cộng lại và tổng (trừ đi phí giao dịch) được sử dụng hết ở đầu ra của giao dịch mới. Tx trước đó là trị băm của giao dịch trước đó. Chỉ số là đầu ra cụ thể trong giao dịch được tham chiếu. ScriptSig là nửa đầu của kịch bản (sẽ được thảo luận chi tiết hơn ở phần sau).

Tập lệnh chứa hai thành phần, chữ ký và khóa công khai. Khóa công khai phải tương ứng với trị băm được cho trong tập lệnh của kết quả. Khóa công khai được sử dụng để xác minh chữ ký của người chuyển, là thành phần thứ hai. Cụ thể, thành phần thứ hai là chữ ký ECDSA trên trị băm của nội dung giao dịch. Chữ ký cùng với khóa công khai, chứng minh giao dịch được tạo ra bởi đúng chủ sở hữu thực sự của số BTC đó. Các cờ (*flag*) trong cấu trúc giao dịch tổng quát xác định kiểu giao dịch và nhờ đó, có thể tạo các loại thanh toán khác nhau.

Đầu ra của giao dịch

Cấu trúc đầu ra được mô tả như hình dưới.

Field	Description	Size
Value	The monetary value of the output in satoshis	8 bytes
Script length	positive integer VI = VarInt	1 - 9 bytes
Script	A calculation which future transactions need to solve in order to spend it	Variable

Một đầu ra chứa mã lệnh chuyển khoản BTC. Giá trị là số Satoshi (1 BTC = 100.000.000 Satoshi). ScriptPubKey là nửa sau của kịch bản (sẽ thảo luận sau). Có thể có nhiều đầu ra và tổng giá trị đầu ra bằng tổng các giá trị đầu vào. Vì mỗi đầu ra từ một giao dịch chỉ có thể được tham chiếu một lần từ đầu vào của giao dịch sau đó, tổng giá trị đầu vào phải bằng tổng giá trị đầu ra, nếu không sẽ được xem là phí giao dịch (*transaction fee*). Chẳng hạn, nếu tổng đầu vào là 50 BTC nhưng chỉ muốn chuyển 25 BTC, mạng BTC sẽ tạo ra hai đầu

ra trị giá 25 BTC: một đến địa chỉ đích cần gửi và một gửi ngược lại (thối lại) cho địa chỉ nguồn, gọi là "tự đổi" dù gửi lại cho chính mình. Bất kỳ BTC đầu vào nào không được chuyển thành đầu ra đều được coi là phí giao dịch; nút tạo ra block (*miner node*) có thể yêu cầu phí và chèn phí vào giao dịch coinbase của block đó.

Bằng chứng

Đối với các giao dịch Segwit, có một danh sách các bằng chứng ngay đầu ra, với mỗi trường bằng chứng (proof field) tương ứng với một đầu vào cùng chỉ số. Mỗi trường bằng chứng chứa một biến cho biết số phần tử trên ngăn xếp (*stack*) của bằng chứng và bản thân các phần tử stack lại chứa các cặp biến, biểu diễn độ dài của dữ liệu và chính dữ liệu đó.

Điều quan trọng cần lưu ý là các trường bằng chứng không được tập hợp lại thành một trường "witness" ở đầu ra có cấu trúc tuần tự, mà mỗi bằng chứng được đặt trong từ khóa "witness" riêng của nó ngay dưới đầu vào tương ứng.

Xác minh giao dịch

Để xác minh rằng đầu vào hợp lệ với giá trị đầu ra được tham chiếu, mạng bitcoin sử dụng hệ kịch bản tựa-Forth (Forth-like) gồm 2 kịch bản: scriptSig đầu vào và scriptPubKey đầu ra. Hai kịch bản sẽ được thực thi để xác minh giao dịch có hợp lệ hay không. Đầu vào input là hợp lệ nếu scriptPubKey trả về kết quả true. Bằng cách sử dụng các kịch bản, người gửi có thể tạo ra các điều kiện phức tạp cần phải thỏa để có thể nhận được giá trị đầu ra output. Chẳng hạn, có thể tạo một đầu ra mà không cần bất kỳ rằng buộc nào; hay cũng có thể yêu cầu đầu vào phải được ký bằng 3 khóa khác nhau; hoặc có thể sử dụng được bằng mật khẩu thay vì khóa.

Các kiểu giao dịch

Mạng BTC cung cấp hai cặp kịch bản scriptSig và scriptPubKey cơ sở. Từ các kịch bản cơ sở, có thể thiết kế ra các loại giao dịch phức tạp hơn và liên kết chúng lại với nhau thành các hợp đồng (contract) để có thể được thực thi bằng mật mã.

Thanh toán cho một địa chỉ – **Pay-to-Pubkey-Hash**

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

Địa chỉ BTC, như ta đã biết, là 1 trị băm nên không thể cung cấp khóa công khai trong scriptPubKey. Khi chuyển khoản BTC đã được gửi đến địa chỉ bitcoin, người nhận sẽ phải cung cấp cả chữ ký và khóa công khai của mình. Tập lệnh xác minh khóa công khai đó sẽ được băm thành trị băm trong scriptPubKey, sau đó kịch bản cũng kiểm tra chữ ký tương ứng khóa công khai đó.

Quá trình xác minh được mô tả trong hình dưới.

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Kết hợp scriptSig và scriptPublickey
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Các hằng được them vào stack
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Nhân bản phần tử trên đỉnh stack
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Tính trị băm phần tử đỉnh stack
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Thêm hằng vào đỉnh stack
<sig> <pubKey>	OP_CHECKSIG	So sánh 2 phần tử trên đỉnh stack
true	Empty.	Nếu tack rỗng thì xác nhận

Thanh toán cho kịch bản băm – **Pay-to-script-hash**

scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL

scriptSig: ..signatures... <serialized script>

m-of-n multi-signature transaction:

scriptSig: 0 <sig1> ... <script>

script: OP_m <pubKey1> ... OP_n OP_CHECKMULTISIG

Địa chỉ P2SH (Pay-to-script-hash) được tạo ra với mục đích chuyển "trách nhiệm cung cấp các điều kiện để chuyển giao dịch từ người gửi tiền sang người đổi. Qua đó, cho phép người gửi nạp tiền cho một giao dịch tùy ý bằng cách sử dụng 20 bytes trị băm". Địa chỉ P2SH tương tự như trị băm 20 bytes của khóa công khai.

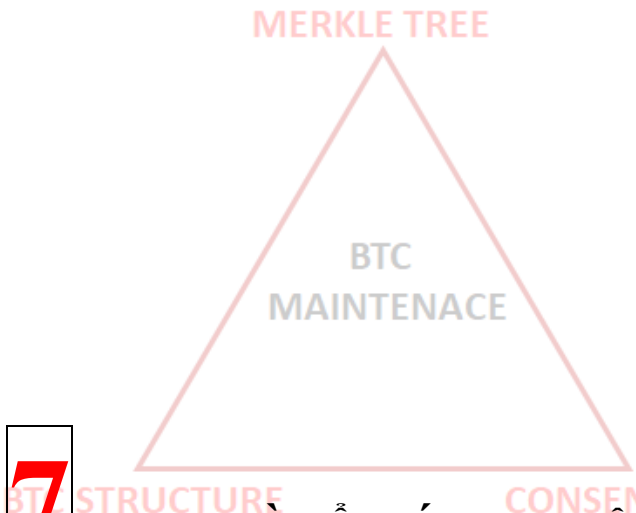
P2SH là phương tiện cho phép tạo ra các giao dịch phức tạp mà không phải giao dịch thanh toán qua địa chỉ trị băm của khóa công khai, có scriptPubKey và scriptSig xác định như trên. Không đặt ra bất kỳ giới hạn nào với kịch bản và do đó có thể thực hiện bất kỳ loại hợp đồng nào khi sử dụng các địa chỉ này.

Chẳng hạn, scriptPubKey trong giao dịch cấp vốn là kịch bản đảm bảo rằng kịch bản trong giao dịch đổi thưởng sẽ băm thành kịch bản được sử dụng để tạo địa chỉ.

Trong scriptSig ở trên, '*chữ ký*' đề cập đến bất kỳ kịch bản để đáp ứng được kịch bản sau đây.

Quá trình xác minh được minh họa qua hình sau.

Stack	Script	Description
Empty.	0 <sig1> <sig2> OP_2 <pubKey1> <pubKey2> <pubKey3> OP_3 OP_CHECKMULTISIG	Chỉ dùng kịch bản scriptSig
0 <sig1> <sig2> OP_2 <pubKey1> <pubKey2> <pubKey3> OP_3	OP_CHECKMULTISIG	Thêm hằng vào stack
true	Empty	Stack rỗng là xác minh thành công



7 DUY TRÌ SỔ CÁI PHI TẬP TRUNG

SỔ CÁI BLOCKCHAIN VÀ BITCOIN

Blockchain

Blockchain, như ta đã biết, là một công nghệ sổ cái phi tập trung và có thể xem như Internet thế hệ hai. Nếu như trong Internet thế hệ thứ nhất hiện nay, với công nghệ liên mạng toàn cầu và những cỗ máy tìm kiếm hiện đại, thông tin được lưu trữ và trao đổi một cách bình đẳng, dân chủ, thì blockchain, tạm cho là Internet thế mới, với khái niệm giao dịch (*transaction*) hứa hẹn sẽ dân chủ hóa việc trao đổi giá trị thực. Vào khoảng giữa năm 2008, đang lúc xảy ra cuộc khủng hoảng tài chính toàn cầu, thì một giao thức mới cho "Hệ thống tiền mặt điện tử ngang hàng" và tạo ra một loại tiền kỹ thuật số Bitcoin (***BTC*** *cryptocurrency*) dựa trên công nghệ blockchain, với giao dịch BTC đầu tiên được thực hiện vào ngày 12/1/2009. Khác với tiền tệ truyền thống, tiền điện tử không được phát hành bởi bất kỳ quốc gia nào; cũng không được lưu trữ trong kho tiền ngân hàng hoặc tín dụng được ghi trong tập tin điện tử ở đâu đó mà chỉ là một bảng kê tất cả các giao dịch trên quy mô toàn cầu (*global transaction*) dựa trên lý thuyết và công nghệ mạng ngang hàng (**P2P** – *Peer-to-Peer*) để xác minh và phê duyệt từng giao dịch BTC. Nhưng blockchain không chỉ là 'tiền điện tử', blockchain là công nghệ mới và có nhiều ứng dụng hơn, công nghệ này đang được đầu tư phát triển với tốc độ nhanh.

Tính chất của công nghệ blockchain

Phân tán – **distributed**

Giao thức cài đặt quy tắc tính toán phân tán nhằm đảm bảo tính toàn vẹn của dữ liệu được trao đổi giữa các thiết bị nối mạng mà không cần bên thứ ba đáng tin. Là CSDL được lưu trữ trên 'máy đào', là các máy tính tình nguyện trên toàn cầu, không có cơ sở dữ liệu trung tâm để có thể bị tấn công khai thác hay bị phá hủy hoặc tắt. Điều này có nghĩa là những thứ có giá trị cao – tiền, cổ phiếu, trái phiếu, quyền sở hữu trí tuệ, âm nhạc, xe hơi và thậm chí

cả phiếu bầu – có thể được lưu trữ và trao đổi mà không cần qua một tổ chức hoặc người trung gian nào.

Mã hóa – **encrypted**

Blockchain sử dụng mã công khai để xác thực duy trì tính bảo mật. Các giá trị của cá nhân có thể được chứng thực là của chính cá nhân đó qua công nghệ mật mã công khai.

Đầy đủ và minh bạch – **inclusive and transparence**

Với số khai công cộng (public blockchain), mọi người đều có thể xem tất cả các giao dịch trên mạng khiến không cá nhân hoặc tổ chức nào có thể che giấu giao dịch. Là nguồn mở nên ai cũng có thể tải xuống và sử dụng. Với blockchain, mọi người đều có thể giao tiếp qua "chế độ xác minh thanh toán đơn giản hóa" và có thể được sử dụng trên thiết bị di động. Không cần phải có chứng từ để được tin hoặc được quyền truy cập.

Bất biến – **immutable**

Mọi giao dịch đều được xác minh và được lưu trữ trong một trang (block) được '*khâu' (chain)* với block liền trước, tạo nên sổ cái blockchain. Mỗi block phải tham chiếu đến block liền trước mới hợp lệ. Thông tin và các giao dịch được lưu 'vĩnh viễn', và không ai có thể thay đổi sổ cái.

Lịch sử – **historical**

Là một sổ cái phi tập trung thể hiện sự đồng thuận trên mạng của mọi giao dịch đã từng xảy ra nên blockchain phải được bảo toàn trọn vẹn. Lưu trữ lịch sử cho phép truy xuất nguồn gốc của thông tin một cách dễ dàng.

Những tính chất này cho phép phát triển các ứng dụng dựa trên công nghệ blockchain cho nhiều mục đích khác nhau.

Thách thức

Tuy nhiên, nếu các vấn đề về quản trị và quản lý không được giải quyết, công nghệ chuỗi khối có thể không đáp ứng được kỳ vọng.

Nền tảng – **platform**

Cần các giao thức để quản lý khả năng mở rộng (bằng chứng cổ phần – *proof of stake*), mức tiêu thụ năng lượng, tính ổn định (chuyển từ bằng chứng công việc – *proof of work*, sang *proof of stake*) và tính mạnh mẽ (*robustness*) của cơ sở hạ tầng mạng để đảm bảo hoạt động lâu dài, thành công (trong một khoảng thời gian dài).

Ứng dụng – **applications**

Cần có một số công cụ giao tiếp thân thiện với người dùng và đào tạo các nhà phát triển lành nghề.

Cấu trúc pháp lý để quản lý – **legal structure for stewardship**

Cần một cơ quan quản trị để điều phối các vấn đề như khả năng tương tác, quyền riêng tư, bảo mật, bảo vệ danh tính và các hành động nhằm giảm mức độ không chắc chắn về mặt pháp lý xung quanh các công nghệ mới nổi.

DUY TRÌ SỔ CÁI BTC

Sổ cái BTC, hay bất kỳ blockchain nào, là CSDL phi-tập trung, nghĩa là có nhiều nơi *lưu giữ cùng một CSDL*, gọi là các nút khai thác (*miner node*). Vì thế bên cạnh tính toàn vẹn (*integrity*) của CSDL sổ cái, tính đồng bộ (*synchronization*) cần phải đảm bảo để tại một thời điểm, CSDL các block ở mọi thời điểm chỉ có thể khác nhau rất ít ở những block cuối mới thêm vào. Có hai nhiệm vụ chính phải thực hiện trên mạng blockchain:

- *Đúc block* – **minted**: Tạo block mới chứa tất cả các giao dịch đã được xác minh và gắn vào blockchain.
- *Đồng bộ hóa* – **synchronization**: Duy trì tính nhất quán của CSDL phi tập trung.

Hai nhiệm vụ này được đảm bảo thông qua cài đặt cơ chế đồng thuận số đông (*majority consensus*). Trong blockchain, CSDL hiện hành là blockchain có số block nhiều nhất. Như vậy, các miner node luôn phải cập nhật để đảm bảo CSDL đang giữ trên máy chủ của mình là CSDL mới nhất. Tính đồng thuận cũng được thể hiện qua 'cuộc đua' đúc block nhanh nhất. Miner nào có bằng chứng (*proof*) cho block mới nhanh nhất sẽ là người chiến thắng và thêm block đó vào CSDL, gắn block vào blockchain hiện hành, và mọi miner khác phải cập nhật cho động bộ với CSDL chung. Mạng BTC sử dụng khái niêm bằng chứng công việc – **PoW** (*Proof of Work*) cho nhiệm vụ này.

Bằng chứng công việc

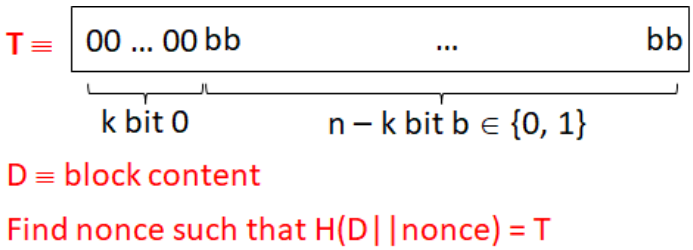
Mô tả bài toán công việc

Bằng chứng công việc – **PoW** là minh chứng chứng minh có làm (và hoàn thành) công việc bằng cách giải một bài toán đố, mà lời giải tìm được được dùng làm bằng chứng cho công việc giải bài.

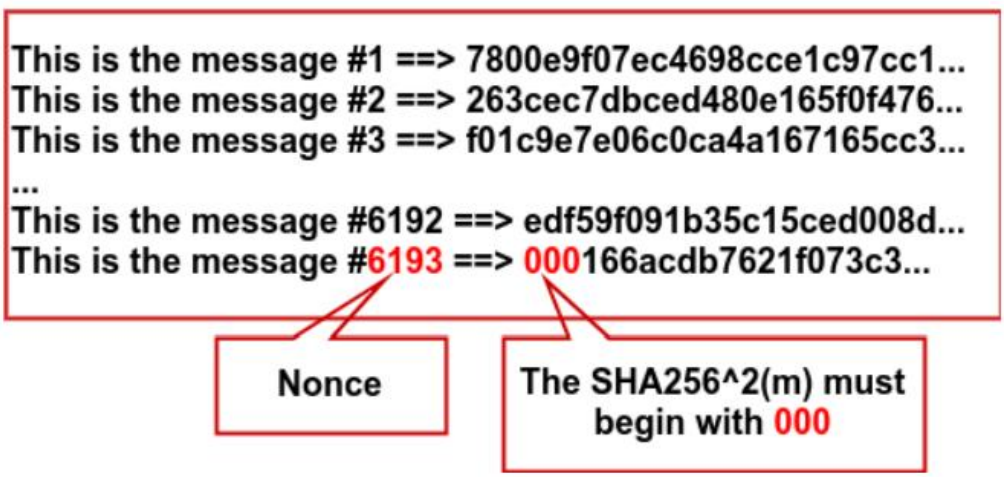
Công việc trong blockchain BTC, và nhiều blockchain khác, là *tính ngược hàm băm* của hàm băm SHA-2 1-chiều. Do tính chất 1-chiều và nhất là tính ngẫu nhiên của SHA-2, cũng như nhiều hàm băm mật mã khác, việc tính ngược hàm băm buộc phải 'vét cạn'. Ý tưởng này hình thành khái niệm 'công việc' (*work*) trong khai thác mạng BTC và được mô tả hình thức như sau.

Bài toán. Cho trước mẫu (*template*) **T**, là một chuỗi nhị phân n bit, với k bit cố định (k < n), thường là k bit cao nhất có giá trị '0', và một văn bản **D**. Bài toán là tìm ra một số ngẫu nhiên **nonce** (viết tắt của *number-once* số ngẫu nhiên dùng 1 lần) sao cho trị băm **H** là số n bit, của chuỗi ghép **D||nonce** thỏa mẫu **T**: **H(D||nonce)** $\in$ **T** = {N có n bit và $0 \le N < 2^{n-1}$}.

Trong mạng BTC, hàm băm H được dùng là SHA-2, n = 256 bit, và số k được dùng để điều chỉnh 'tốc độ làm việc' sao cho trung bình một khối được đúc ra mất khoảng 10 phút. Hình sau mô tả bài toán tìm nonce.



Ví dụ minh họa



Mẫu **T** trong ví dụ này có 12 bits đầu tiên của trị băm là 0 (3 ký tự thập lục phân).

Bài toán: Cho **SMS** = "Đây là tin nhắn #", tìm một giá trị **nonce** sao cho SHA-2(**SMS||nonce**) = SHA-2("Đây là tin nhắn #"||nonce) khớp với một phần tử trong tập mẫu **T**, nghĩa là trị băm kết quả phải bắt đầu bằng ít nhất 3 số 0.

Lời giải: **nonce** = **6193** (trong ví dụ minh họa này).

Cài đặt PoW

Có 2 chiến lược cài đặt bài toán bằng chứng công việc: tuần tự hay ngẫu nhiên.

Cách 1: tuần tự – **sequence**

(1) $nonce \xleftarrow{\$} \mathbb{Z}_{2^k}$

(2) If SHA-2(D||nonce) ∈ T: Return nonce

(3) Else nonce = nonce + 1

(4) If (chưa hết giờ): goto(1)

Như ta biết, trung bình 1 block được tạo ra cần 10 phút, nên thời gian để duyệt tuần tự tối đa phải ít hơn 10 phút.

Cách 2: ngẫu nhiên – **random**

(1) $nonce \xleftarrow{\$} \mathbb{Z}_{2^k}$

(2) If SHA-2(D||nonce) ∈ T: Return nonce

(3) If (chưa hết giờ): goto(1)

Cộng tác cùng làm việc

Trong trường hợp có nhiều máy chủ, giả sử là m máy chủ, cùng hợp tác làm việc, công việc được chia thành m tập con không giao nhau

$T = T_1 \cup \ldots \cup T_m$, $T_i \cap T_{i\neq j} = \varnothing$, (i, j = 1,2,…, m).

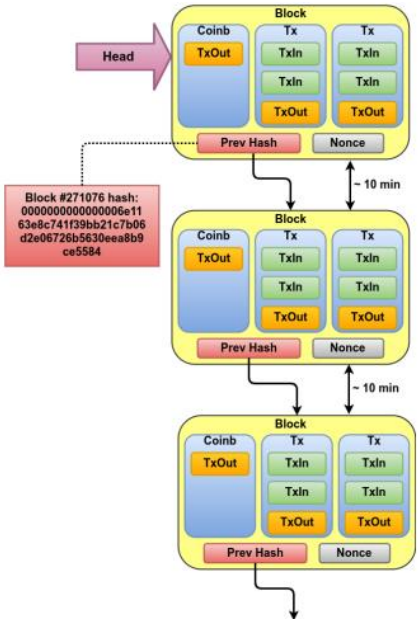
và giao cho mỗi máy làm việc trên một phần cụ thể trong thời gian cho phép. Thường thì việc chia này được tính theo năng lực làm việc của các máy chủ. Việc đo năng lực được thực hiện dựa trên hàm băm SHA-2.

Cấu trúc của block

Cấu trúc

Mạng blockchain BTC, như ta biết, là một dạng sổ cái – blockchain. Mỗi block là các trang ghi các giao dịch (**Tx** – *transaction*) và được liên kết với block trước đó trong blockchain. Mỗi block được bảo vệ bằng bằng chứng công việc (**PoW** – *Proof of Work*) là giá trị ngược của hàm băm một phần của nội dung block.

Mỗi trang – block gồm (**i**) các Tx hợp lệ, (**ii**) trị băm của block liền trước và (**iii**) một nonce. Trong đó, nonce trong một khối là lời giải của bài toán giá trị ngược của băm một phần. Hình sau mô tả các block trong blockchain.



– Mỗi block có một giao dịch đặc biệt, được gọi là coinbase.
– Coinbase là giao dịch đầu tiên trong block, chỉ có 1 đầu vào ***TxIn***, và không có bất kỳ đầu ra ***TxOut*** nào trước đó.
– Một block thường có nhiều giao dịch nhưng chỉ có một coinbase.
– Nút khai thác có thể chọn và đặt các giao dịch vào block để khai thác (*mine*). Việc chọn thường được quyết định dựa trên mức phí (*fee*) mà giao dịch phải trả.

Nút khai thác

Nút khai thác (*miner node*) là nút mạng giữ một bản sao đầy đủ của sổ cái blockchain, và giữ các cấu trúc dữ liệu phụ như bộ nhớ đệm để lưu đầu ra các giao dịch chưa được sử dụng (*un-spent*) và nhóm các giao dịch chưa được xác nhận (*un-validation*), để có thể nhanh chóng xác thực các giao dịch mới khi khai thác block mới.

Nếu giao dịch hoặc block nhận được là hợp lệ, nút sẽ cập nhật cấu trúc dữ liệu của nó và chuyển tiếp nó đến các nút mạng liên kết với nút này.

Điều quan trọng cần lưu ý là một nút mạng không cần phải tin cậy các nút mạng khác vì việc xác thực được thực hiện độc lập chỉ sử dụng các thông tin mà nó nhận được từ các nút mạng khác.

Nguyên lý duy trì sổ cái

Khi nút khai thác (miner node) tìm thấy một khối mới, nó sẽ lan truyền khối đó trên mạng chuỗi khối. Mọi nút khai thác nhận được phải kiểm tra tính hợp lệ của khối, nghĩa là xem xem liệu nó có phải là lời giải cho bài toán công việc, có là đảo ngược băm một phần hay không. Nếu đúng, các nút khai thác đó sẽ nhật CSDL của mình để đồng bộ hóa với CSDL chung:

(1) Cập nhật bộ nhớ phụ chứa đầu ra giao dịch chưa tiêu (UTXO).

(2) Cập nhật nhóm bộ nhớ giao dịch chưa được xác nhận.

Các nút mạng luôn duy trì một số kết nối nhất định tới các nút mạng khác trong mạng blockchain.

Tấn công chi tiêu nhiều lần

Mạng BTC giải quyết vấn nạn chi tiêu lại (*double spend*) theo cách phân tán mà không cần cơ quan trung ương quyết định giao dịch nào là hợp lệ. Kẻ tấn công muốn thay đổi sổ cái blockchain ở một trang – block nào đó, sẽ phải khai thác (giải lại) lại tất cả các block tính từ block muốn sửa đến block đầu tiên của blockchain BTC.

Cách duy nhất kẻ tấn công có thể thực hiện thành công điều này là phải có tốc độ băm nhanh hơn phần còn lại của toàn bộ mạng BTC. Tấn công này được gọi là tấn công 51% (*51% attack*).

Tấn công chạy đua

Tấn công khi đua (*race attack*) xảy ra khi một nút nhận giao dịch chưa được xác nhận xung đột với giao dịch nằm trong bộ nhớ chưa được xác nhận của nó, tức là chi tiêu cùng một đầu ra, khi ấy, giao dịch mới sẽ bị loại bỏ.

Do đó, các nút chỉ giữ trong bộ nhớ phụ một bản sao của giao dịch nhận được đầu tiên. Race Attack thành công khi nhà cung cấp chấp nhận thanh toán cho giao dịch chưa được xác nhận khi chỉ kiểm tra một vài nút mạng. Kẻ tấn công có thể gửi một giao dịch đến các nút gần nhà cung cấp và một giao dịch khác tới nhiều nút khác trong mạng. Do đó, chỉ các nút ở gần nhà cung cấp mới thấy giao dịch gửi tiền cho nhà cung cấp, trong khi phần còn lại của mạng chứa trong bộ nhớ phụ của chúng một khoản chi tiêu lại (*double spent*). Để chống lại tấn công này, nhà cung cấp dịch vụ cần đợi giao dịch được đưa hẳn vào block.

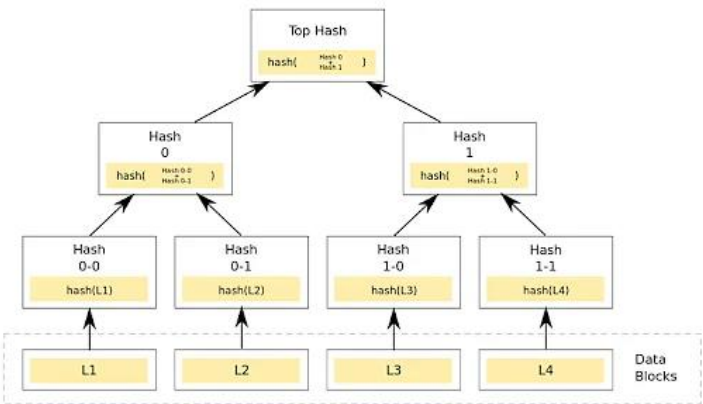
Cây băm

Để tối ưu quá trình xác minh tính toàn vẹn của các block, gồm các giao dịch, dữ liệu được lưu trữ dạng cây băm, do Merkle đề xuất, gọi là cây Merkel (*Merkle tree*), là một cấu trúc dữ liệu dạng cây trong đó mỗi nút lá, là nút ứng với các giao dịch, được gắn nhãn băm của dữ liệu và mỗi nút trung gian được gắn nhãn băm mật mã của nhãn các nút con của nó. Phần lớn cài đặt cây băm nhị phân (mỗi nút có hai nút con), nhưng tổng quát, chúng cũng có thể có nhiều nút con hơn. Cây Merkle trong Blockchain thực chất là gì và nó được sử dụng như thế nào trong blockchain BTC?

Cây Merkle

Cây Merkle (*Merkle tree*), còn được gọi là cây băm nhị phân, là một cấu trúc dữ liệu phổ biến trong khoa học máy tính. Chúng được áp dụng trong mạng BTC, và các mạng tiền điện tử khác, để mã hóa dữ liệu blockchain cho hiệu quả và an toàn hơn.

Cây Merkle là cấu trúc dữ được tạo thành từ các trị băm của nhiều khối dữ liệu khác nhau nén tất cả các giao dịch trong một block giúp xác minh nội dung nhanh chóng và an toàn trên các khối là tập dữ liệu lớn, đồng thời xác minh tính nhất quán nội dung của dữ liệu. Hình dưới minh họa một cây Merkle với khối 4 giao dịch.

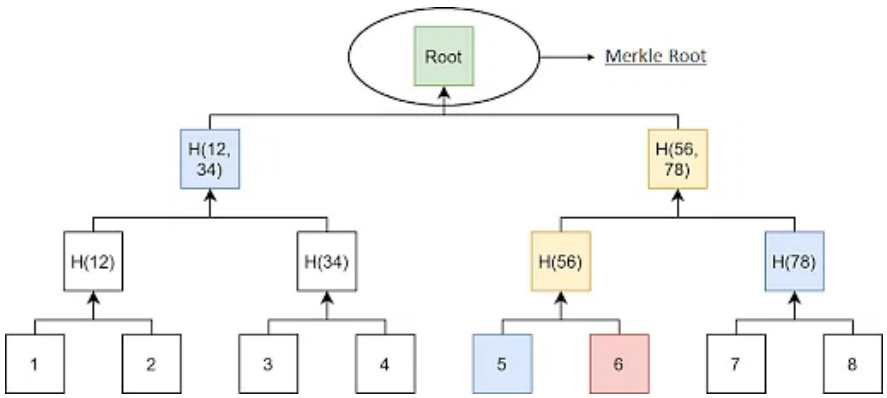


Cây Merkle trong hình trên gồm 4 nút lá (*leaf*) được gán nhãn là trị băm $H(L_i)$, i =1, 2, 3, 4, của 4 dữ liệu lần lượt là L_i, i = 1, 2, 3 ,4. Các nút tầng giữa, gọi là các nút trung gian, được gán nhãn là trị băm của hai nút con tầng lá. Nút gốc, gọi là gốc cây băm Merkle (*Merkle root*), được gán nhãn là trị băm của hai nút con tầng giữa.

Gốc cây Merkle

Gốc cây Merkle là một phương pháp toán học đơn giản để xác nhận các sự kiện trên cây Merkle. Chúng được sử dụng trong tiền điện tử để đảm bảo rằng các trang dữ liệu được gửi qua mạng ngang hàng là nguyên vẹn, không bị hư hại và không bị thay đổi.

Gốc cây Merkle cũng đóng một vai trò rất quan trọng trong việc tính toán để duy trì hoạt động của các loại tiền điện tử như BTC. Hình dưới minh họa cây Merkle với 8 nút lá có giá trị lần lượt là 1, 2,…, 8, và gốc Root.



Hoạt động của cây Merkle

Cây Merkle tính trị băm tóm tắt của tất cả các giao dịch trong một block và tạo dấu vân kỹ thuật số (*fingerprint*) cho cả block, cho phép người dùng xác minh xem một giao dịch có là một giao dịch trong khối hay không. Tiến trình tạo cây băm Merkle như sau:

– Cây Merkle được tạo bằng cách băm các cặp nút liên tục cho đến khi chỉ còn lại một trị băm; trị băm này được gọi là Merkle Root hoặc Root Hash.

– Quá trình băm được xây dựng từ dưới lên, sử dụng ID của giao dịch, là trị băm của các giao dịch riêng lẻ.

– Mỗi nút không phải nút lá là trị băm của trị băm trước đó và mỗi nút lá là trị băm của dữ liệu giao dịch.

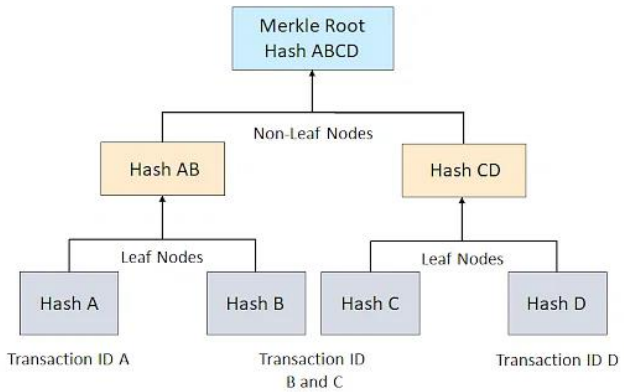
Ví dụ: A, B, C và D là bốn giao dịch, tất cả đều được gom vào cùng một block. Mỗi giao dịch sau đó sẽ được băm:

$h_A = H(A)$, $h_B = H(B)$, $h_C = H(C)$, và $h_D = H(D)$.

Tiếp đến, các trị băm trên được ghép nối với nhau:

$h_{AB} = H(h_A||h_B)$, $h_{CD} = H(h_C||h_D)$.

Cuối cùng, gốc cây Merkle được hình thành bằng cách kết hợp hai trị băm này: Root = $H_{ABCD} = H(h_{AB}||h_{CD})$.



Trong thực tế, Merkle Tree phức tạp hơn nhiều (cụ thể, mỗi ID của giao dịch gồm 64 ký tự). Tuy nhiên, ví dụ trên giúp ta có cái nhìn tổng quan về cách các thuật toán hoạt động và lý do tại sao chúng lại hiệu quả đến vậy.

Lợi ích của Merkle Tree trong Blockchain

Cây Merkle có bốn lợi ích khi áp dụng cho sổ cái blockchain:

- Xác thực tính toàn vẹn của dữ liệu.
- Chiếm ít dung lượng đĩa.
- Thông tin nhỏ khi trao đổi trên mạng.
- Xác minh hiệu quả.

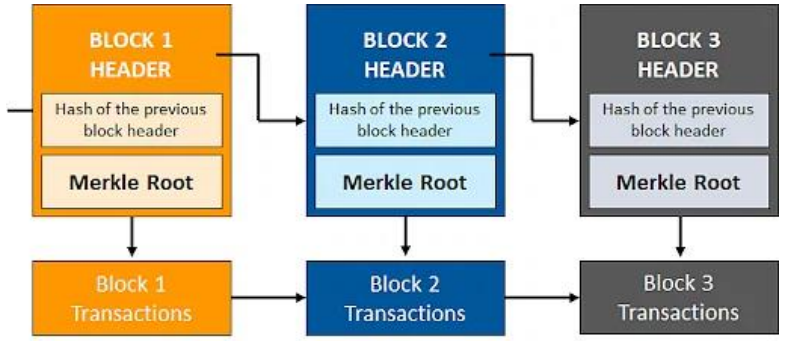
Tính cần thiết của cây Merkle cho sổ cái blockchain

Giả sử không sử dụng cây Merkle thì mọi nút trên mạng sẽ phải giữ một bản sao của mọi giao dịch BTC từng được thực hiện. Như vậy, sẽ phải lưu giữ lượng thông tin rất lớn.

Bất kỳ yêu cầu xác thực nào trên BTC sẽ yêu cầu một lượng dữ liệu khổng lồ phải truyền qua mạng, do đó, sẽ cần phải tự mình xác thực dữ liệu.

Để xác nhận tính toàn vẹn của mọi dữ liệu, sẽ cần rất nhiều sức mạnh thực hiện tính toán so sánh trên sổ cái.

Cây Merkle là một giải pháp cho vấn đề này. Chúng băm các bản ghi trong block, từ đó tách bằng chứng dữ liệu khỏi dữ liệu đó. Điều đó cho thấy chỉ cần cung cấp một lượng nhỏ thông tin trên mạng là tất cả những gì cần thiết để giao dịch có hiệu lực. Hơn nữa, có thể chứng minh rằng các biến thể sổ cái đều giống hệt nhau về sức mạnh máy tính danh nghĩa và băng thông mạng.



Merkle Tree breaking the data into tiny parts of information

Ứng dụng thực tế cây Merkle trong sổ cái blockchain

Có nhiều ứng dụng của cây Merkle đã được cài đặt.

– **Git**: một hệ kiểm soát phiên bản phân tán, là một trong những hệ thống được sử dụng rộng rãi nhất. Nó được sử dụng để xử lý các dự án của các lập trình viên từ khắp nơi trên thế giới.

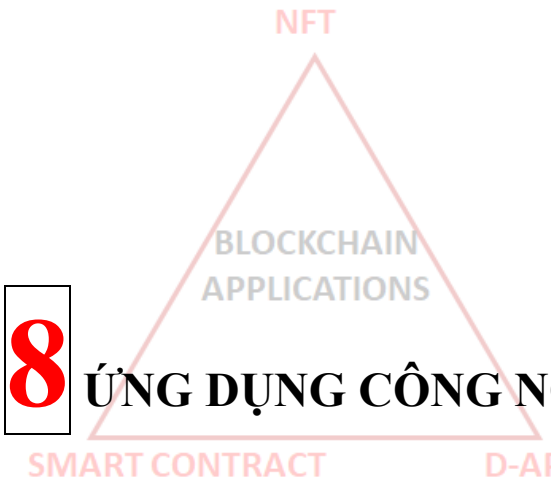
– **Hệ tập tin liên hành tinh** – **IPFS** (*Interplanetary File System*): một giao thức phân tán ngang hàng. Là nguồn mở, cho phép các máy tính tham gia và sử dụng hệ thống này.

– **Tạo chữ ký minh bạch**: Là một phần của kỹ thuật tạo ra nhật ký minh bạch của chứng chỉ có thể kiểm chứng được.

– **Amazon DynamoDB và Apache Cassandra**: sử dụng cây Merkle trong quá trình sao chép dữ liệu. Các cơ sở dữ liệu phân tán No-SQL này sử dụng cây Merkle để kiểm soát sự khác biệt.

PHẦN 2

BLOCKCHAIN ỨNG DỤNG



8 ỨNG DỤNG CÔNG NGHỆ BLOCKCHAIN

HỢP ĐỒNG THÔNG MINH

Smart contract - hợp đồng thông minh

Sau blockchain Bitcoin, một số blockchain mới xuất hiện để giải quyết các vấn đề còn tồn đọng hoặc cải tiến một số khía cạnh. Chẳng hạn, Ethereum cải tiến Bitcoin về mặt lập trình qua khái niệm hợp đồng thông minh (**SC** – *Smart Contract*).

Ethereum cho phép lập trình viên tạo ra các ứng dụng phi tập trung của mình, gọi tắt là **DApps** (*Decentralized Applications*) và triển khai trên blockchain Ethereum (*ETH blockchain*). Bằng cách này, ~~các~~ lập trình viên có thể tạo ra các chức năng và ứng dụng mới cho blockchain. Điều này làm khái niệm tiền điện tử và giá trị của blockchain ngày càng phổ biến.

Nhờ hợp đồng thông minh, lập trình viên có thể lập trình các ứng dụng sẽ chạy trên blockchain và được định nghĩa không hình thức như sau.

Hợp đồng thông minh là tập hợp các cam kết, dạng kỹ thuật số, gồm các giao thức mà các bên liên quan phải thực hiện.

Về mặt công nghệ,

Hợp đồng thông minh là các chương trình máy tính bất biến chạy một cách tất định trên máy ảo Ethereum (**EVM** – *ETH Virtual Machine*).

Lưu ý rằng việc xác định điều khoản hợp đồng thông minh một cách tổng quát là phức tạp vì nó phụ thuộc vào bối cảnh. Từ các định nghĩa trên, hợp đồng thông minh có thể được mô tả như một kịch bản được lập trình hoặc chương trình chạy trên blockchain và kích hoạt các hành động khi một sự kiện xảy ra.

Sổ cái blockchain sử dụng SC

Do hợp đồng thông minh ngày càng phổ biến, các blockchain sau thường cho phép cộng đồng phát triển ứng dụng của riêng mình trên blockchain nền, như Terra hoặc Solana, cùng nhiều blockchain khác.

Cũng có thể xây dựng một blockchain tương thích với EVM, như Binance Smart Chain (BSC), một blockchain độc lập tồn tại trên Ethereum, nhưng công cụ EVM thì được sử dụng để biên dịch các hợp đồng thông minh được triển khai trên blockchain tương thích EVM. Cách tiếp cận này cho phép sử dụng các hợp đồng thông minh giống như cách Ethereum thực hiện.

Bên cạnh đó, có các blockchain có cách tiếp cận khác, như Bitcoin chẳng hạn. BTC không sử dụng hợp đồng thông minh theo cách này mà sử dụng SC để quản lý giao dịch nhưng không tạo DApps.

ỨNG DỤNG PHI TẬP TRUNG

Dapp - ứng dụng phi tập trung

Ứng dụng phi tập trung (**DApp** – *Decentralized Applications*), trước hết, không phải là một hợp đồng thông minh có giao diện người dùng. Nếu một công ty tập trung quản lý hợp đồng thông minh và dữ liệu của họ cũng được tập trung, đó không phải là DApp, mà thực tế, nó là một ứng dụng tập trung. Một DApp được hình thành bởi các thành phần khác nhau phải được phân cấp thực sự để thực hiện mục đích DApp.

DApp là một ứng dụng có khả năng lưu trữ dữ liệu (*data storage*), giao diện người dùng (*frontend*), trình hỗ trợ (*backend*), truyền thông qua tin nhắn (*message communication*) và phân giải tên được phi tập trung hóa một phần hoặc toàn phần.

Về kỹ thuật là khó nhưng có thể thực hiện được. Nhiều công cụ phi tập trung hóa một ứng dụng đang được triển khai. Ví dụ, Pinata là một trang web lưu trữ dữ liệu phi tập trung, Ethereum Name Service cho phép mua một tên miền phi tập trung và các công cụ khác giúp tạo DApp.

Các thành phần của DApp

DApp bao gồm một số thành tố phi tập trung gồm:

– ***Trình hỗ trợ*** (*backend*): Một hợp đồng thông minh, là phần mà logic nghiệp vụ được lập trình, hay nói cách khác, DApp có thể làm gì. Backend nên tinh nhất có thể vì việc triển khai hợp đồng thông minh hoặc thực hiện một chức năng trả tốn phí (**gas**). Gas giúp hệ thống hoạt động an toàn vì nếu không có gas, có thể có chức năng được thực thi mãi mãi, gây rắc rối cho mạng blockchain. Cần phải trả chi phí giao dịch mỗi lần giao dịch được thực thi.

– ***Giao diện*** (*frontend*): Các ngôn ngữ lập trình như HTML, CSS hoặc JavaScript có thể được sử dụng để lập trình giao diện người dùng. Không cần phải có kiến thức cao về EVM bởi vì LTV không tương tác với EVM, mà chỉ sử dụng các công cụ, thư viện và các khung

nền (*framework*) để DApp trở nên thân thiện với người dùng. Giao diện người dùng này thường được lập trình với JavaScript và sử dụng các thư viện như web3.js hoặc ethers.js.

– ***Lưu trữ dữ liệu*** (*data storage*): Do phí gas cao, hợp đồng thông minh không phù hợp để lưu trữ và xử lý một lượng lớn dữ liệu. Vì lý do này, thông tin cần phải được lưu trữ ngoài chuỗi khối. Có thể đạt bằng cách sử dụng máy chủ tập trung truyền thống, nhưng phi tập trung là lựa chọn tốt nhất. Có thể lưu trữ thông tin theo cách phi tập trung dùng:

• **IPFS** – *Interplanetary File System*: Là một hệ thống tập tin phi tập trung nơi người dùng mạng P2P duy trì dữ liệu của block. Mỗi phần dữ liệu được gắn thẻ là trị băm của nó. Sau đó, người dùng có thể truy xuất dữ liệu này khi cần thiết.

• Swarm. Một hệ thống tập tin phi tập trung khác của Ethereum Foundation, tương tự như IPFS. Giống như IPFS, dữ liệu có thể được truy cập bằng hàm băm.

Giao tiếp qua tin nhắn (*message communication*): Trao đổi tin nhắn giữa các ứng dụng, người dùng hoặc các phiên bản khác nhau của DApp cũng là một phần thiết yếu. Vì lý do này, Ethereum Foundation cung cấp giao thức Whisper, một giao thức trao đổi tin nhắn qua mạng P2P phi tập trung.

Phân giải tên (*name resolution*): Ethereum Name Service (ENS), giống như DNS, nhưng hoạt động phi tập trung. DNS truyền thống dịch tên miền thành địa chỉ IP, giúp ta có thể truy cập vào địa chỉ IP này một cách thân thiện. ENS hoạt động theo cách tương tự, nhưng bây giờ khóa công khai được dịch sang cấu trúc name.eth. Khóa công khai gồm 42 ký tự thập lục phân, khó nhớ. Nhờ ENS, có thể đặt tên cho khóa công khai này để dễ nhớ và dễ sử dụng hơn. ENS có thể được sử dụng để tạo tên miền hoặc cung cấp bí danh cho địa chỉ ví, giúp việc này trở nên dễ dàng hơn khi chuyển tiền vào tài khoản này.

Các tính năng chính của DApp

Một số ưu điểm hoặc tính năng không có trong các ứng dụng tập trung.

• ***Khả năng phục hồi***: DApp backend được lưu trên một blockchain và DApp sẽ khả dụng miễn là blockchain này còn hoạt động. Có thể một phần nào đó của DApp không phi tập trung. Do đó, quyền truy cập vào ứng dụng có thể bị hạn chế, chẳng hạn, về tên miền được dịch bằng DNS truyền thống thay vì ENS. Công ty chủ trang web có thể sửa URL web hoặc hết hợp đồng với công ty DNS. Khi ấy, người dùng không thể tìm thấy trang web nhưng có thể cung cấp cho hợp đồng thông minh các công cụ như etherscan. Nếu blockchain còn hoạt động, hợp đồng thông minh luôn là một phần của nó và luôn có thể được tìm thấy bằng cách này hay cách khác.

• ***Tính minh bạch***: Vì backend là hợp đồng thông minh nên mọi người đều có thể kiểm tra mã lệnh. Hơn nữa, các giao dịch trên blockchain đều được công khai nên bất cứ ai cũng có thể kiểm tra các giao dịch được thực hiện với hợp đồng thông minh này. Bằng cách này, người dùng có thể kiểm tra xem hợp đồng thông minh có hợp pháp hay không hoặc có điều gì đó bất hợp pháp.

• ***Khả năng chống kiểm duyệt***: Giống như giao dịch blockchain, hợp đồng thông minh không thể sửa đổi sau khi được tải lên blockchain hoặc được thay thế bằng một blockchain khác.

WEB 3.0

Giới thiệu Web 3.0

Web 3.0 là thuật ngữ dùng để để chỉ ra rằng Web 3.0 tập trung vào các mục đích blockchain. Nhiều trang web và người viết sử dụng Web 3.0 và Web3 như nhau, gây ra một số nhầm lẫn cho người mới. Để tránh nhầm lẫn cho người đọc, ta phân biệt:

• ***Web 3.0***: Trang web dữ liệu có thể được xử lý bằng máy. Web 3.0 có thể được giải thích là một trang web sử dụng các công nghệ trí tuệ nhân tạo, như học máy hoặc ngôn ngữ tự nhiên, để diễn giải dữ liệu do người dùng theo cách tự nhiên. Cách này có thể hiểu Web 3.0 là "**web ngữ nghĩa**" – Semantic Web.

• ***Web3***: Là các ứng dụng phi tập trung chạy trên chuỗi khối, ứng dụng cho phép mọi người tham gia cung cấp dữ liệu. Vì các trang web này được phân bổ trên một blockchain nên thông tin được tải lên sẽ không thể sửa đổi. Chẳng hạn, không thể kiểm duyệt các tweet. Điểm khác biệt với web 2.0 là Web3 không yêu cầu dữ liệu cá nhân để thực hiện giao dịch. Ngoài ra, rất khó để một trang web3 ngừng hoạt động vì nó được duy trì bởi nhiều nút, không phải bởi một máy chủ tập trung.

Lịch sử Web3

Web3 là một khái niệm mới trong chuỗi khối. Để hiểu Web3 và nguồn gốc của nó, cần giải thích về web 1.0 và web 2.0.

• ***Web 1.0***: Còn được gọi là trang web tĩnh. Là các trang web đầu tiên xuất hiện trên Internet. Những trang web này ở chế độ chỉ đọc, nghĩa là thông tin được viết tĩnh trên trang web cho người dùng đọc. Người dùng không thể chỉnh sửa trang web.

• ***Web 2.0***: Cải tiến của web 1.0. Ở web 2.0, người dùng có thể chỉnh sửa và thêm dữ liệu vào đó. Loại web này đã có thể đọc-ghi. Mặt tiêu cực là một khi dữ liệu này đã có trên Internet thì người chủ nội dung không thể kiểm soát. Hơn nữa, nền tảng và dữ liệu tập trung, tin tặc hacker chỉ cần tấn công máy chủ để đánh cắp dữ liệu hoặc kích động hành vi xấu. Web3 xuất hiện để giải quyết những vấn đề này.

• ***Web 3.0***: Web3 là frontend của DApp được phân bổ trên blockchain. Không có dữ liệu cá nhân có liên quan vì người dùng được đăng ký bằng ví điện tử, như Metamask, Coinbase hoặc ví khác được trang web cho phép. Quá trình kết nối người dùng với trang web Web3: trình mở rộng ví được cài đặt trong trình duyệt của người dùng. Sau đó, khi người dùng muốn đăng ký trên một trang web mới, chỉ có ví phải được kết nối với trang mạng. Đây là một bước dễ dàng vì chỉ cần nhấp chuột vào nút trang web để liên kết ví vào trang web này là cần thiết. Thông tin được cung cấp bởi ví là ẩn danh vì, theo quan điểm của trang web, chỉ có một địa chỉ công khai (thực tế là khóa công khai của ví) đã được được kết nối. Không có tên người dùng hoặc email nào có thể được liên kết với một người. Dữ liệu này phi tập trung vì backend của DApp nằm trên một chuỗi khối. Nên không ai có thể kiểm soát được dữ liệu và không thể bán dữ liệu của người dùng.

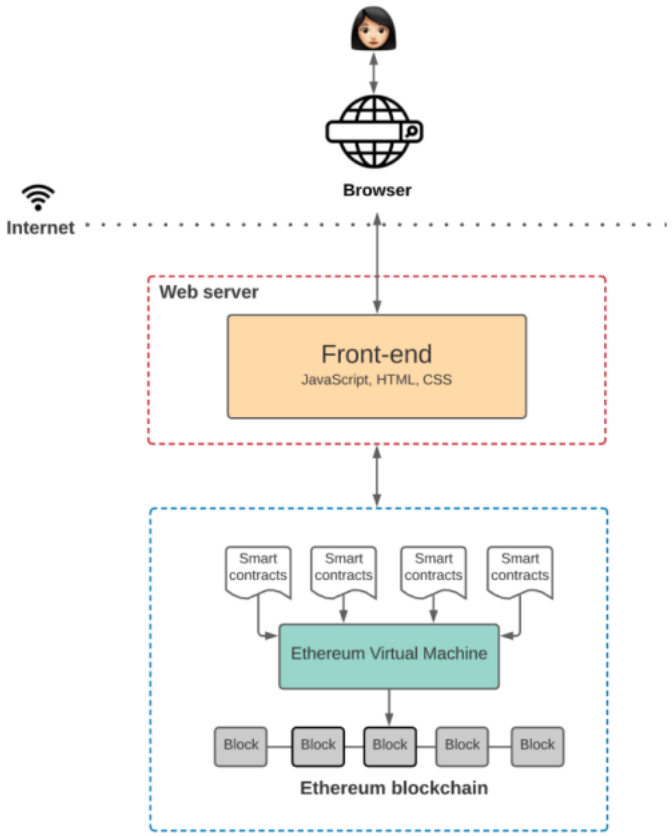
Kiến trúc Web 3.0

Tương tự như DApps, các trang web này cần hợp đồng thông minh để cài đặt nghiệp vụ. Frontend cần thiết để cho phép người dùng tương tác với nó. Thêm vào đó là các thành phần sau:

• ***Sổ cái blockchain***: Là nền tảng của Web3. Loại trang web này có hợp đồng thông minh làm backend và SC này được chạy trên một blockchain.

• **EVM**: Một máy ảo để thực hiện các hợp đồng thông minh trên blockchain.

Hình sau minh họa kiến trúc tổng quát của Web 3.0



Hình trên minh họa các thành phần Web3 và cách chúng tương tác với nhau. Thông qua trình duyệt và với kết nối Internet, người dùng có thể kết nối với bất kỳ trang web nào. Nhưng ở đây, thay vì làm việc với các máy chủ tập trung, frontend kết nối trực tiếp với hợp đồng thông minh, hoạt động như backend. Trong ví dụ này, hợp đồng thông minh được triển khai trên ETH. Sau đó EVM biên dịch và chuyển thành mã byte tương tác trực tiếp với ETH.

Hạn chế của Web3 trên Ethereum

Để xây dựng DApp trên một chuỗi, cũng cần biết các hạn chế và khả năng sổ cái blockchain nền. Blockchain ETH có một số hạn chế.

• ***Khả năng mở rộng*** – **scalability**. Mọi giao dịch phải được xử lý, xác minh và được đúc để có thể được thêm vào blockchain. Quá trình này mất trung bình từ 10 giây, là thời gian để tạo ra

một block trên Ethereum. Thời gian này là chặn dưới, vì vậy thời gian để xác minh giao dịch sẽ không ít hơn 10 giây. Trong trường hợp số giao dịch Ethereum tăng, thời gian tạo block sẽ tạo nút thắt cổ chai và nhiều giao dịch sẽ phải chờ trong nhóm giao dịch.

• ***Chi phí*** – **cost**. Một vấn đề khác liên quan đến khả năng mở rộng là chi phí giao dịch, hay còn gọi là phí gas. Một khoản phí bổ sung được thêm vào giá giao dịch cơ sở (nếu có) để tránh những giao dịch không cần thiết. Khi một giao dịch được thực hiện, khoản phí này có thể được người dùng sửa đổi trực tiếp trong toàn bộ ví. Các nút khai thác thường ưu tiên chọn các giao dịch có trả phí cao, các giao dịch với chi phí thấp hơn không bao giờ được xác minh xử lý.

• ***Khả năng truy cập*** – **accessibility**. Cần thêm các tiện ích mở rộng trình duyệt khác nhau để đăng ký người dùng vào ứng dụng Web3. Vấn đề là một số trình duyệt chưa thực sự sẵn sàng để tích hợp tất cả các yêu cầu này.

MÃ THÔNG BÁO KHÔNG THỂ THAY THẾ

Giới thiệu

Mã thông báo không thể thay thế – **NFT** (*Non-Fungible Token*), là mã (*token*) không thể thay đổi. Vì các token này không thể thay thế được nên có thể nói rằng không có hai NFT hoàn toàn giống nhau và đây là đặc điểm chính của công nghệ mới này. Có thể định nghĩa là:

NFT có thể là một loại tài sản mật mã đại diện cho một loại tài sản duy nhất.

NFT được coi là một cách kiểm tra xem ai đã sở hữu một thứ gì có giá trị. Vì thế, khi thông tin này được kiểm tra, ai đó (tổ chức, công ty, câu lạc bộ) có thể đề nghị một số lợi ích cho chủ sở hữu NFT (người sở hữu tài sản). Và đây chính là lý do hấp dẫn của NFT. Người có ảnh hưởng, doanh nhân, thương hiệu lớn, nghệ sĩ và nhiều công ty có thể xây dựng cộng đồng bằng cách sử dụng NFT. Hơn nữa, các trường hợp sử dụng khác như sưu tầm, nội dung trò chơi hoặc tác phẩm nghệ thuật là những tình huống khác nhau mà NFT có thể hữu ích. Các thuộc tính của NFT:

– ***Tính không thể phân chia*** – **indivisibility**: Là một trong những thuộc tính chính của NFT khiến nó hữu dụng. Tính không thể phân chia có nghĩa là không thể mua được một phần của NFT, chỉ có thể mua toàn bộ NFT.

– ***Quyền sở hữu*** – **ownership**: Vì không thể phân chia nên NFT chỉ được sở hữu bởi một người dùng. Nói cách khác, một NFT chỉ thuộc về một tài khoản ví. Chỉ người giữ NFT có thể chuyển nó từ tài khoản này sang tài khoản khác.

– ***Tính độc đáo*** – **uniqueness**: Khi có NFT, người giữ có thể chắc chắn rằng không tồn tại một NFT nào khác giống như NFT của mình. Thông thường, NFT được biểu thị bằng một hình ảnh, do đó việc nghĩ rằng NFT có thể bị sao chép là lý do. Trên thực tế thì có thể, nhưng thực tế là NFT được gắn thẻ với một số nhận dạng và đã liên kết một địa chỉ hợp đồng thông minh. Do đó, việc tạo NFT được thể hiện bằng các hình ảnh giống với hình ảnh gốc là có thể. Nhưng không có cách nào để liên kết NFT không nguyên gốc này với hợp đồng thông minh gốc. Khi đó, NFT ban đầu là duy nhất.

– ***Sự khan hiếm*** – **scarcity**: Đôi khi, giá trị của NFT được quyết định bởi sự khan hiếm của nó.

– ***Tính minh bạch*** – **transparence**: Như mọi giao dịch sổ cái blockchain, giao dịch NFT cũng được công khai. Do đó, có thể theo dõi một NFT cụ thể để biết NFT số lần đã được giao dịch, điều này có thể giúp biết liệu NFT được truy tìm có phải là bản gốc hay không. Tính minh bạch giúp chống lừa đảo.

THỰC HÀNH: XÂY DỰNG DAPP TRÊN MỘT MẠNG ETHEREUM

NỘI DUNG THỰC HÀNH

- Thiết lập một local ethereum blockchain
- Triển khai một smart contract trên ethereum
- Triển khai một Dapp tương tác với smart contract

CÀI ĐẶT

Private Ethereum blockchain – sử dụng Ganache

- Mạng ảo này sẽ không có node nào và không có mining time.
- Nó cũng tạo ra 10 địa chỉ Ethereum ảo.

Bước 1: Run → ***cmd***

Bước 2: Gõ lệnh sau

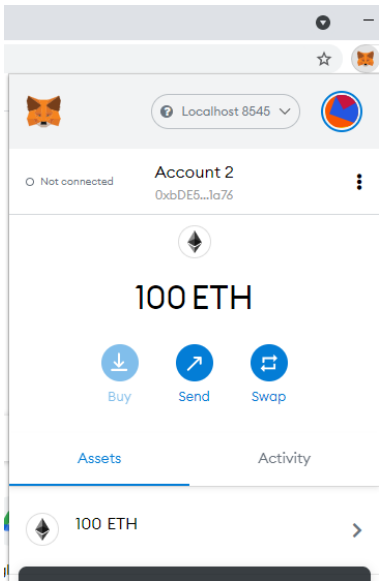
npm install -g ganache-cli

ganache-cli

Bước 3: Cài MetaMask (là một ví Ethereum (Wallet Ethereum))

https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn

Bước 4: Import 1 private key từ một trong 10 account của ethereum vào MetaMask

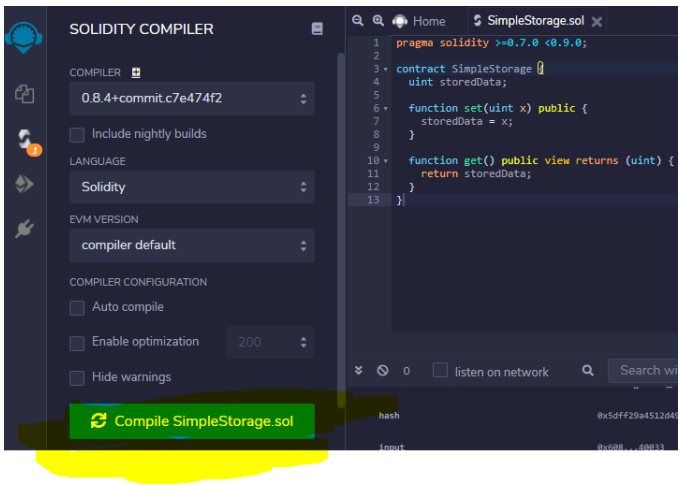


TRIỂN KHAI MỘT SMART CONTRACT TRÊN MẠNG ETH

- ***Viết smart contract trên remix***: https://remix.ethereum.org/

```
pragma solidity >=0.7.0 <0.9.0;

contract SimpleStorage {
  uint storedData;

  function set(uint x) public {
    storedData = x;
  }

  function get() public view returns (uint) {
    return storedData;
  }
}
```

- ***Biên dịch***

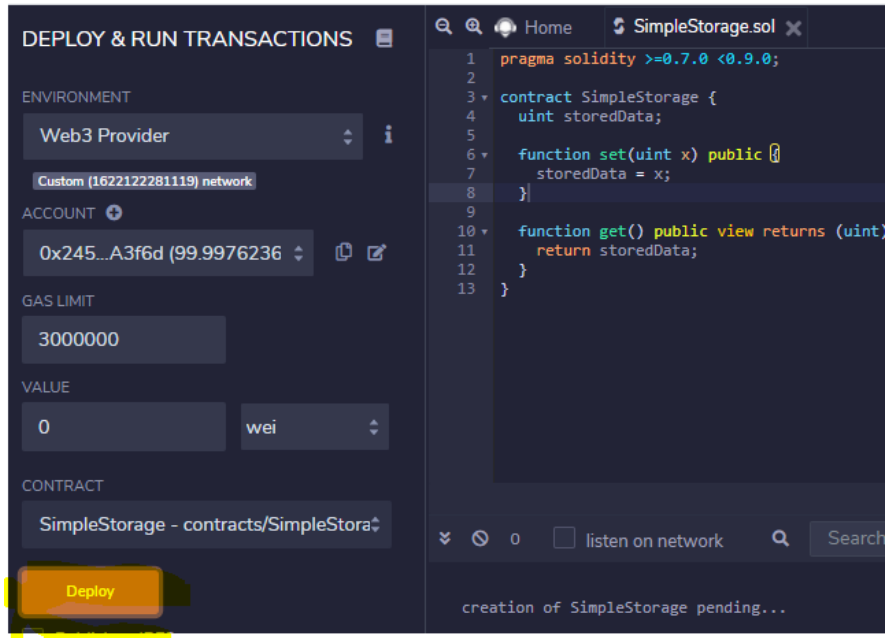


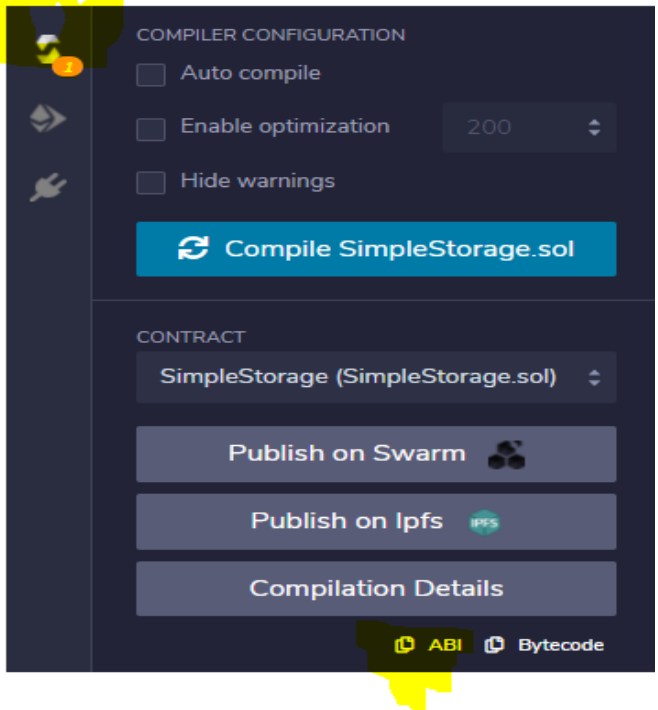
- ***Liên kết giữa ứng dụng với private Ethereum***. Sử dụng thư viện Web3 trên Web Browser và Deploy smart contract trên Ethereum

- ***Quan sát kết quả trên ganache***

```
eth_gasPrice
eth_sendTransaction

  Transaction: 0x33a047e84512312f0fee28920106955375b9c2301c8d1a088fc8c0e3daea3403
  Contract created: 0x8366fe645d6102f8e401f178631663600be14079
  Gas usage: 118819
  Block Number: 1
  Block Time: Thu May 27 2021 20:32:32 GMT+0700 (Indochina Time)

net_version
eth_getTransactionReceipt
eth_getTransactionReceipt
```

- ***Xem ABI của smart contract***

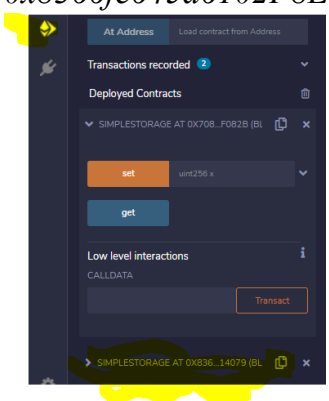




```
[
	{
		"inputs": [
			{
				"internalType": "uint256",
				"name": "x",
				"type": "uint256"
			}
		],
		"name": "set",
		"outputs": [],
		"stateMutability": "nonpayable",
		"type": "function"
	},
	{
		"inputs": [],
		"name": "get",
		"outputs": [
			{
				"internalType": "uint256",
				"name": "",
				"type": "uint256"
			}
		],
		"stateMutability": "view",
		"type": "function"
	}
]
```


- ***Địa chỉ của smart contract trên ETH***

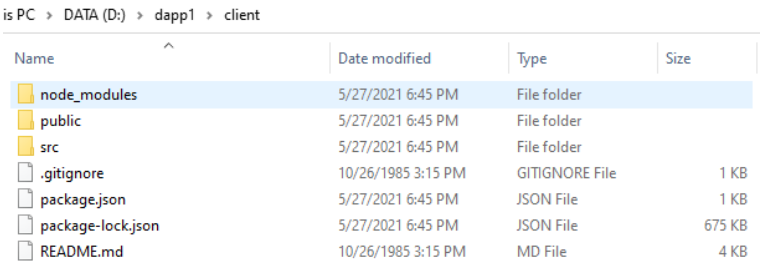
0x8366fe645d6102F8E401F178631663600Be14079



Viết Dapp

- ***Tạo Fontend***

npx create-react-app client



cd client

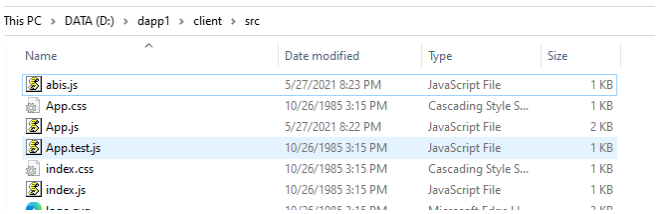
npm install web3

Chỉnh sửa file D:\dapp1\client\src\App.js

```
import logo from './logo.svg';
import './App.css';

import React, { useState } from 'react';
import Web3 from 'web3';
import { simpleStorageAbi } from './abis';
const web3 = new Web3(Web3.givenProvider);
// contract address is provided by Truffle migration
const contractAddr = '0x8366fe645d6102F8E401F178631663600Be14079';
const SimpleContract = new web3.eth.Contract(simpleStorageAbi,
contractAddr);

function App() {
        const [number, setNumber] = useState(0);
        const [getNumber, setGetNumber] = useState('0x00');

        const handleGet = async (e) => {
                e.preventDefault();
                const result = await SimpleContract.methods.get().call();
                setGetNumber(result);
                console.log(result);
        }

        const handleSet = async (e) => {
                        e.preventDefault();
                        const accounts = await window.ethereum.enable();
                        const account = accounts[0];
                        const gas = await SimpleContract.methods.set(number)
                .estimateGas();
                        const result = await
SimpleContract.methods.set(number).send({
```

```
                        from: account,
                        gas
                        })
                console.log(result);
        }

  return (
    <div className="App">
      <header className="App-header">
        <form onSubmit={handleSet}>
          <label>
            Set Number:
            <input
              type="text"
              name="name"
              value={number}
              onChange={ e => setNumber(e.target.value) } />
          </label>
          <input type="submit" value="Set Number" />
        </form>
        <br/>
        <button
          onClick={handleGet}
          type="button" >
          Get Number
        </button>
        { getNumber }
      </header>
    </div>
);}
export default App;
```

- Tạo file *abis.js*

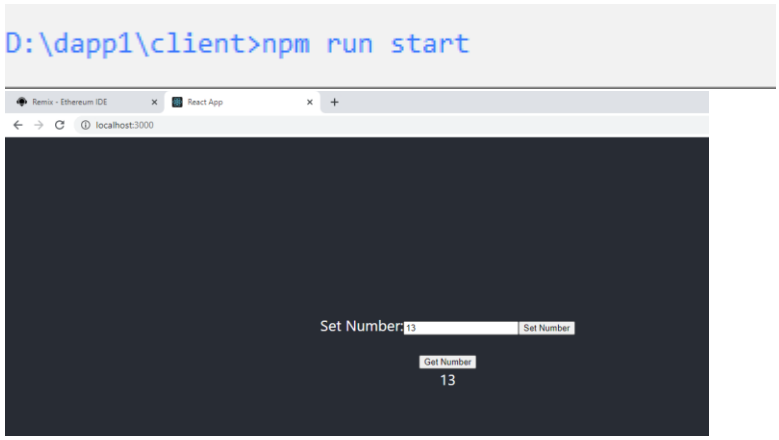


Nội dung File abis.js chưa ABI của smart contract

```
// abis.js
export const simpleStorageAbi = [
```

```
	{
		"inputs": [
			{
				"internalType": "uint256",
				"name": "x",
				"type": "uint256"
			}
		],
		"name": "set",
		"outputs": [],
		"stateMutability": "nonpayable",
		"type": "function"
	},
	{
		"inputs": [],
		"name": "get",
		"outputs": [
			{
				"internalType": "uint256",
				"name": "",
				"type": "uint256"
			}
		],
		"stateMutability": "view",
		"type": "function"
	}
]
```


- ***Chạy React***



PHẦN 3

BLOCKCHAIN NÂNG CAO

9 CHỮ KÝ BỘI VÀ TRI THỨC TRỊ-0

CHỮ KÝ BỘI

Giới thiệu

Mã khóa công khai (*public key cryptography*) là nền tảng bảo mật cho các giao dịch trên mạng Internet. Trong mã khóa công khai, khóa gồm một cặp khóa: khóa công khai (*public key*) và khóa cá nhân (*private key*). Từ khóa cá nhân có thể suy ra được khóa công khai, nhưng không thể suy ra được khóa cá nhân từ khóa công khai. Vì thế, mã khóa công khai còn được gọi là mã bất đối xứng (*asymmetric cryptography*). Cặp khóa bất đối xứng được sử dụng trong hai ứng dụng chính:

(**1**) **Trong xác thực**: Chứng minh được rằng có kiến thức về khóa riêng.

(**2**) **Trong mã hóa**: Mọi người đều có thể gửi tin mật bằng cách sử dụng khóa công khai của người nhận, nhưng chỉ người có khóa cá nhân tương ứng mới giải mã được tin mật nhận được.

Lược đồ chữ ký đường cong elliptic

Cặp khóa ECC

Ta sẽ sử dụng ký hiệu của mật mã đường cong Elliptic (*Elliptic Curve Cryptosystem*) để trình bày các giao thức, mô hình của chúng ta, theo đó, cặp khóa ECC gồm:

. $d \in \mathbb{Z}_p$: khóa cá nhân.

. $e = dG, G \in (E)$: khóa công khai.

Thuật giải chữ ký đường cong elliptic

Thuật giải chữ ký đường cong elliptics – **ECDSA** (*Elliptic Curve Digital Signature Algorithm*) là mô hình được sử dụng phổ biến trên các sổ cái blockchain, ở đó, ECC được dùng để thiết kế các giao thức ký trên thông điệp và xác thực chữ ký.

Tạo chữ ký – **Signature**

Để tạo ra chữ ký trên thông điệp m, thực hiện các bước sau:

(1) Tạo số ngẫu nhiên (dùng một lần – **nonce**) r.
(2) Tính R = rG.
(3) Tính trị băm $f = H(R||e||m)$.
(4) Tính $s = r + df$.
Thì (r, s) là chữ ký của m.

Xác minh chữ ký – **Verification**

Để xác minh (r, s) có phải là chữ ký của m hay không, sử dụng khóa công khai e = dG tương ứng với khóa cá nhân d, chỉ cần kiểm tra đẳng thức sau có đúng hay không. Nếu đúng thì chấp nhận (r, s) là chữ ký trên m:

$$sG == R + ef.$$

Thực vậy,

$$sG = (r + df)G = rG + dfG = R + dGf = R + ef.$$

Lược đồ chữ ký Schnorr

Chữ ký Schnorr là một lược đồ chữ kí tuyến tính nên có một số tính chất hữu ích khi dùng trong thực tế.

Ta biết là đường cong elliptic có tính chất tuyến tính. Thực vậy, cho hai đại lượng vô hướng x và y tương ứng với hai điểm X = xG và Y = yG trên đường cong elliptic, ta có:

$$(x + y)G = xG + yG = X + Y.$$

Lược đồ chữ ký Schnorr có dạng $s = r + fd$, cũng tuyến tính, nên chúng thích hợp để xây dựng chữ ký tổng hợp.

Tổng hợp chữ ký

Do tính tuyến tính, chữ ký Schnorr có thể được dùng để xây dựng chữ ký bội hai-trong-hai.

Alice và Bob muốn ký trên cùng một thông điệp mà không cần phải tin tưởng lẫn nhau; như thế, cả hai cần có khả năng chứng minh quyền sở hữu các khóa ký của mình và chữ ký tổng hợp chỉ hợp lệ nếu cả Alice và Bob đều cung cấp một phần chữ ký của mình.

Giả sử các khóa cá nhân được ký hiệu là d_i, và các khóa công khai là e_i. Alice và Bob cần cung cấp các số ngẫu nhiên dùng một lần – **nonce**.

(1) $e_{agg} = e_a + e_b$. #khóa công khai tổng hợp là tổng 2 khóa công khai.

(2) $f = H(R_a||R_b||e_a||e_b||m)$. #băm văn bản kết buộc với các khóa công khai và các nonce.

(3) $s_{agg} = R_a + R_b + (d_a + d_b)f = (R_a + d_af) + (R_b + d_bf) = s_a + s_b$. #hai bên cùng ký trên trị băm

Vì thế Alice và Bob có thể cung cấp R tương ứng của mỗi người.

Phân tích bảo mật

Xét lại kịch bản trên, nhưng bây giờ, Bob biết trước nonce R_a và khóa công khai e_a của Alice bằng cách chờ cho đến khi Allice tiết lộ chúng. Khi ấy, Bob cung cấp khóa công khai giả là $e'_b = e_b – e_a$ và cung cấp cho Allice số nonce là $R'_b = R_b – R_a$.

Giả sử chữ ký tổng hợp là:

$s_{agg} = R_a + R'_b + f(e_a + e'_b)$,

thì Bob cũng có thể tạo ra chữ ký này bằng kỹ thuật tấn công hủy khóa (*Key Cancellation Attack*). Thực vậy, ta có

$s_{agg}G = R_a + R'_b + f(e_a + e'_b)$

$= R_a + R_b – R_a + f(e_a + e_b – e_a)$

$= R_b + fe_b$

$= r_bG + fdG$.

Nên để tấn công, Bob chỉ phải tính:

$s_{agg} = R_b + fd_b = s_b$.

Tiêu chí chữ ký tổng hợp

Trong tấn công hủy khóa, Bob không cần biết khóa cá nhân liên kết với các giá trị P và Q đã công bố. Ta có thể ngăn Bob ký vào một tin nhắn chứng minh rằng Bob biết khóa riêng. Điều này có thể thực hiện nhưng lại cần thêm một vòng nhắn tin khác giữa các bên, cách làm như vậy ảnh hưởng nhiều đến tính sử dụng của người dùng (**UX** – *User Experiment*). Ta cần xây dựng một lược đồ ký có các tính chất sau:

- Phải được chứng minh là an toàn không cần phải cung cấp kiến thức về khóa bí mật.
- Phải thỏa mãn phương trình Schnorr, tức là chữ ký kết quả có thể được xác minh bằng biểu thức có dạng R + fX.
- Cho phép tạo chữ ký tổng hợp tương tác (**IAS** – *Interactive Aggregation Signature*), mà người ký được yêu cầu hợp tác.
- Cho phép chữ ký tổng hợp không tương tác (**NAS** – *Non-interactive Aggregation Signature*), trong đó việc tổng hợp có thể được thực hiện bởi bất kỳ ai.
- Cho phép mỗi người ký cùng ký vào một tin nhắn, m.
- Cho phép mỗi người ký ký vào tin nhắn của riêng mình, m_i.

MuSig, trình bày sau là lược đồ tổng hợp chữ ký thỏa tất cả các tiêu chí của chữ ký tổng hợp

Lược đồ chữ ký tổng hợp

Ta sẽ mô tả lược đồ, **MuSig** tương tác, trong đó mỗi bên ký đều ký trên cùng một thông điệp. Mô hình hoạt động như sau:

(1) Mỗi người ký có một cặp khóa.

(2) Mỗi người ký chia sẻ cam kết về số dùng một lần nonce công khai của mình (ta sẽ bỏ qua bước này trong phần trình bày). Bước này là cần thiết để ngăn chặn một số loại tấn công khóa lừa đảo.

(3) Mỗi người ký công bố khóa công khai e_i và số ngẫu nhiên $nonce_i$ của mình.

(4) Mọi người đều tính "*khóa chung*" E như nhau:

$. l = H(d_1 || \dots || d_n)$,

$. a_i = H(l || d_i)$,

$. d = \sum_{i=1}^{n} a_i d_i$.

Lưu ý rằng thứ tự khóa công khai là tất định. Nên cần sử dụng một số quy ước xác định thứ tự này, chẳng hạn như thứ tự từ điển.

Ký

Để mọi người tham gia cùng ký trên thông điệp m, mọi người ký thực hiện.

(1) Mọi người tính số ngẫu nhiên chung $R = \sum R_j$.

(2) Tính thách thức f = H(R||d||m).

(3) Tính và cung cấp $s_i = r_i + d_i a_i f$.

Chữ ký tổng hợp là $s = \sum s_i$.

Lưu ý rằng điểm khác biệt duy nhất ở đây so với chữ ký Schnorr chuẩn là có thêm hệ số a_i.

Xác minh chữ ký

Để xác minh chữ ký s, chỉ cần kiểm tra đẳng thức

sG == R + fd

Thực vậy,

$sG = \sum s_i G$

$= \sum (r_i + d_i a_i f) G$

$= \sum r_i G + \sum d_i G a_i f$

$= \sum R_i + f \sum e_i a_i$

= R + fd

Khả năng kháng tấn công hủy khóa

Lược đồ MuSig có khả năng kháng tấn công hủy khóa. Thực vậy, giả sử Bob cung cấp các thông tin giả mạo $R_f = R_b - R_a$, $e_f = e_b - e_a$.

Cả Alice và Bob tính thông tin chung sau:

$. l = H(d_a || d_b)$,

$. a_a = H(l||d_a)$,

$. a_f = H(l||d_f)$,

$. d = a_a d_a + a_f d_f$,

$R = R_a + R_f, R_f = R_b$,

$f = H(R||d||m)$.

Nếu Bob giả mạo chữ ký bằng $s_b = R_b + d_s f$, thì với giả sử d_s không phải khóa cá nhân của Bob.

$s_b G = R + ed$

$= (r_b + d_s f)G$

$= R_b + f(e_a d_a + a_f d_f)$

$= R_b + f(a_a d_a + a_f d_b - a_f d_a)$

$= (r_b + f a_a d_a + f a_f d_b - f a_f d_a)G$

$d_s f = f a_a d_a + f a_f d_b - f a_f d_a$

$d_f = a_a d_a + a_f d_b - a_f d_a$,

Bob tấn công thất bại.

Tấn công dùng lại

Điều quan trọng là phải chọn một nonce mới cho mỗi lần ký. Cách tốt nhất để thực hiện việc này là sử dụng trình tạo số giả ngẫu nhiên được bảo mật bằng mật mã (**CSPRNG** – *Cryptographic Secure Pseudo-Random Number Generator*).

Nhưng ngay cả trong trường hợp này, giả sử kẻ tấn công có thể lừa ta ký một thông điệp mới bằng cách "dùng lại" chữ ký tạo ra một phần chữ ký tổng hợp. Tại thời điểm này, kẻ tấn công cung cấp một thông điệp khác, e' = H(…||m') để ký. Do không nghi ngờ, mọi người tính thành phần chữ ký của mình:

$s_i' = r_i + f' a_i d_i$.

Do kẻ tấn công có quyền truy cập tập đã ký những lần trước $s_i = r_i + f a_i d_i$. Kẻ tấn công thực hiện:

$s_i' - s_i = (r_i + f' a_i d_i) - (r_i + f a_i d_i)$

$a_i d_i (f' - f)$.

Vậy

$d_i = \frac{s_i' - s_i}{a_i (f' - f)}$.

Mọi thừa số ở về phải của đẳng thức cuối kẻ tấn công đều biết và do đó hắn có thể trích xuất khóa cá nhân của mọi người một cách dễ dàng. Thật khó để bảo vệ trước kiểu tấn công này. Một cách là gây khó khăn (hoặc không thể) dùng và bắt đầu tiến trình ký lại. Nếu ký nhiều chữ ký bị gián đoạn thì cần phải bắt đầu lại từ bước một. Điều này khá bất tiện, nhưng cho đến khi có giải pháp mạnh hơn, đây đang là giải pháp tốt nhất mà ta có!

ỨNG DỤNG CHỮ KÝ BỘI TRONG SỔ CÁI BLOCKCHAIN

Theo truyền thống, ví điện tử được liên kết với chủ sở hữu cặp khóa công khai/cá nhân. Mật mã khóa công khai phụ thuộc vào việc giữ bí mật khóa cá nhân. Điều này gây ra khó khăn với người dùng bình thường.

Để giải quyết, giải pháp thay thế là chia sẻ trách nhiệm về khóa giữa nhiều bên, một quy trình tăng tính bảo mật và khả năng truy xuất nhưng có nguy cơ làm giảm khả năng tiếp cận. Có hai phương pháp chính: (1) chữ ký bội (**MuSig** – *multisig*) và hợp tác tính toán (**MPC** – *Multi-Party Computation*).

(**1**) **MuSig**. Chữ ký bội tạo ra nhiều khóa và yêu cầu tất cả hoặc một số người ký trên giao dịch.

(**2**) **MPC**. Hợp tác tính toán chia một khóa thành các thành phần, phân phối chúng và yêu cầu ghép chúng lại theo kiểu tri thức trị 0 (zero knowledge).

Ví với chữ ký bội

Hiện tại, hầu hết các blockchain đều triển khai cơ chế khóa bội. Tuy nhiên thiết kế này có một số nhược điểm.

Ví với chữ ký bội – MultiSig wallet, đã được hỗ trợ trong ngay từ những ngày đầu phát triển BTC và quản lý khóa nhiều bên.

MultiSig chia sẻ trách nhiệm về tính bảo mật ví bằng cách sử dụng nhiều khóa. Thay vì dùng một khóa duy nhất, MultiSig tạo ví có nhiều khóa và phân phối khóa cho các bên liên quan.

Ưu điểm.

- Độ an toàn cao vì các khóa hoạt động độc lập nhau.

Nhược điểm.

- Bảo mật. Kịch bản dễ bị lỗi.
- Hiệu quả. Nhiều chữ ký làm tăng kích thước của các giao dịch và các khoản phí liên quan đến chúng.
- Tính riêng tư. Nhiều chữ ký cho thấy địa chỉ này là MultiSig và có thể gây chú ý đến nó.

Hợp tác tính toán

Hợp tác tính toán – MPC, đề cập đến một tập hợp các kỹ thuật mã hóa cho phép nhiều bên – mỗi bên nắm giữ dữ liệu riêng tư của riêng mình – thực hiện một tính toán mà không phải tiết lộ bất kỳ dữ liệu cá nhân nào đang nắm giữ.

MPC cho phép quản lý khóa phân tán bằng cách "chia khóa" thành nhiều phần – thay vì chỉ một người giữ khóa cá nhân, các bên có thể giữ các thành phần của khóa cá nhân và một số tập con trong số đó có thể kết hợp với nhau để ký giao dịch.

Lược đồ chia sẻ bí mật của Shamir (SSSS)

Adi Shamir, 1979, phác thảo một phương pháp chia dữ liệu ẩn thành n phần trong đó cần ít nhất k phần hợp tác để tiết lộ bí mật ban đầu (trong đó k lớn hơn hơn n/2). Phương pháp này được gọi là Lược đồ chia sẻ bí mật của Shamir (SSSS). Có thể hình dung cho k điểm phân biệt, có một đa thức bậc (k-1). Chẳng hạn, qua hai điểm phân biệt (x_1, y_1) và (x_2, y_2), có đường thẳng y = ax + by qua 2 điểm đó. Hay, qua ba điểm, có đa thức bậc hai $y = ax^2 + bx + c$ đi qua.

Ý tưởng là xây dựng một đa thức $y = a_K + a_{K-1}x^{K-1} + \ldots + a_2x^2 + a_1x$, bậc (K – 1) với a_K là mã bí mật và các thừa số còn lại là ngẫu nhiên. Để phục hồi số bí mật a_K, cần K điểm bất kỳ trong N điểm bằng cách sử dụng đa thức nội suy Lagrange.

Ví dụ, đặt mã bí mật S = 65, N = 4, K = 2.

– Ban đầu, để mã hóa mã bí mật, xây dựng đa thức bậc (K – 1) = 1: giả sử y = a + bx, trong đó

$a_K = a$ là mã bí mật.

– Cho b là số ngẫu nhiên bất kỳ, giả sử b = 15 thì y = 65 + 15x, và tạo ra N = 4 điểm. Giả sử 4 điểm đó là (1, 80), (2, 95), (3, 110), (4, 125).

Rõ ràng, ta có thể tạo lại đa thức ban đầu từ hai điểm bất kỳ trong số 4 điểm này và đa thức thu được, số hạng không đổi a là mã bí mật bắt buộc.

Áp dụng vào mã công khai, khóa cá nhân sẽ là a_K, a = 65 trong ví dụ đang xét. Mỗi bên nhận được một trong bốn điểm trên. Khi cần chữ ký, bất kỳ hai đối tác (2 điểm) cũng đủ để xây dựng lại đa thức và ta lấy lại được a = 65.

Tạo khóa – **KeyGen**

Tạo một khóa duy nhất và được chia thành nhiều phần.

Ký tên – **Signature**

Tất cả hoặc một số phần khóa được tập hợp lại để phục hồi khóa bí mật.

<u>*Ưu điểm*</u>:

- Phí tương tự như giao dịch bình thường không thay đổi.
- Riêng tư. Không thu hút sự chú ý.

<u>*Nhược điểm*</u>:

- Sau khi khóa cá nhân được phục hồi và sử dụng để ký, nó đã bị lộ.

HÀM SONG TUYẾN TÍNH

Cặp ghép song tuyến tính

Cặp ghép song tuyến tính, gọi tắt là cặp ghép (*pairing*), là ánh xạ $f: G_1 \times G_2 \to G_T$, định nghĩa trên các nhóm G_1, G_2, G_T có bậc nguyên tố p, với các phần tử sinh lần lượt là

$g_1 =< G_1 >, g_2 =< G_2 >, và\ g_T =< G_T >$.

Nếu $G_1 \equiv G_2$, cặp ghép được gọi là đối xứng, ngược lại là bất đối xứng.

Cặp ghép có hai tính chất hữu ích cho mật mã: tính song tuyến tính và tính không suy biến.

- **Tính song tuyến tính** (*bi-linearity*). $\forall u \in G_1, v \in G_2, a, b \in \mathbb{Z}_p: f(u^a, v^b) = f(u, v)^{ab}$.
- **Tính không suy biến** (*non-degeneracy*). $f(g_1, g_2) \neq 1_{G_T}$, với 1_{G_T} là phần tử đơn vị của G_T.

Cặp ghép và giao thức thiết lập khóa Diffie-Hellman

Giả sử $<g_1> = <g_2> = G$, và Allice, Bob, Charlie có các khóa bí mật a, b, c, và trao đổi các khóa công khai g^a, g^b, g^c, thì khóa chung của Allice, Bob, và Charlie là $k = f(g, g)^{abc}$.

Cặp ghép và chữ ký BLS

Chữ lý BLS – Boneh-Lynn-Sacham, như sau:

(1) Giả sử $G_2 = <g_2>$, vah H là hàm băm H: $\{0, 1\}^* \rightarrow G_1$.

(2) Cặp khóa $(d \in \mathbb{Z}_p, g_2^e \in G_2)$.

(3) Để ký trên thông điệp m bằng khóa cá nhân d, tính $\sigma = H(m)^d \in G_1$.

(4) Để xác minh chữ ký σ của thông điệp m bằng khóa công khai e, kiểm $f(\sigma, g_2) == f(H(m), e)$.

Mã hóa dựa trên định danh

Mã dựa trên định danh – **IBE** (*Identification-Based Encryption*) do Boneh và Franklin đề xuất dựa trên cặp ghép. Trong lược đồ IBE, ta mã hóa trực tiếp địa chỉ email (hoặc số điện thoại) người dùng, thay vì khóa công khai khó nhớ.

Để IBE hoạt động, cần một bên thứ ba đáng tin cậy (**TTP –** *Trusted-Third Party*) được gọi là bộ tạo khóa (PKG – Private Key Generator), sẽ cấp khóa bí mật cho người dùng dựa trên địa chỉ email của họ.

PKG dùng khóa bí mật của mình $msk \in \mathbb{Z}_p$, tương ứng với khóa công khai $mpk = g_2^s, với\ < g_2 > = G_2$. **mpk** được công bố và được sử dụng để mã hóa tin nhắn cho bất kỳ người dùng nào qua địa chỉ email của họ. Điều quan trọng là phải giữ bí mật **msk**.

Gọi $H_1: \{0,1\}^* \rightarrow G_1^*$; $H_T: \{0,1\}^* \rightarrow G_T$ là 2 hàm băm mật mã. Để mã hóa thông điệp m có n-bít và gửi cho người có địa chỉ email là id, thực hiện mã tựa-ElGamal (*ElGamal-like*):

(1) $g_{id} = f(H_1(id), mpk) \in G_T$.

(2) $r \overset{\$}{\leftarrow} \mathbb{Z}_p$.

(3) $c = \left(g_2^r, m \oplus H_T(g_{id}^r)\right) \in G_2 \times \{0,1\}^n$.

BLS12-381

BLS12-381, do Barreto, Lynn and Scott đề xuất (không phải chữ ký BLS của Boneh – Lynn – Schamam mô tả trên), trên 3 nhóm G_1, G_2, và G_T:

- G_1 là nhóm đường cong elliptic $E(F_q) = \{(x, y) \in F_q^2 | y^2 = x^3 + 4\}, với\ |G_1| = p$.

- G_2 trên $E'(F_{q^2}) = \{(x, y) \in F_{q^2}^2 | y^2 = x^3 + 4(1 + i)\}$, với $i^2 = -1, và\ |G_2| = p$.
- G_T là tập tất cả các căn bậc p của đơn vị trong $F_{q^k}, k = 12$.

Cặp ghép f(.,.) được mở rộng thành

$f(u, v) = f_{p,u}(v)^{\frac{q^k-1}{p}}$.

Đa cặp ghép

Bất cứ khi nào phải tính tích của n cặp ghép, trước tiên, ta có thể tính n vòng lặp và thực hiện phép lũy thừa cuối cùng thay vì n phép. Điều này làm chậm đáng kể thời gian tính toán ghép nối trong nhiều ứng dụng.

$\prod_i f(u_i, v_i) = \prod_i f_{p,u_i}(v_i)^{(q^k-1)/p} = \prod_i f_{p,v_i}(u_i)^{(q^k-1)/p}$.

TRI THỨC TRỊ-0

Hệ chứng minh không lộ tri thức

Giả sử công khai hàm $f: X \rightarrow Y$.

Cho $(x, y) \in X \times Y$, cần kiểm tra xem $y == f(x)$ hay không, chỉ bằng một ít phép tính $y' = f(x)$ và so sánh $y' == y$?

Bài toán đặt ra là mọi người muốn kiểm tra phải lặp cùng các bước tính toán $f(x)$. Nhưng điều gì xảy ra khi chi phí tính *f* quá cao và kích thước *x* lớn (có khi cả nhiều GB) hay *x* có chứa một số thông tin nhạy cảm như mật khẩu, khóa cá nhân, ngày sinh, .v.v. mà ta không muốn tiết lộ?

Câu hỏi đặt ra là liệu có thể kiểm tra đẳng thức $y == f(x)$ đúng mà không phải lặp đi lặp lại các bước tính toán của $f(x)$, hay cũng không có bất kỳ tri thức nào về *x*? Bên cạnh đó, ta cũng muốn chi phí tính toán ít.

Hệ ràng buộc hạng-1

Hệ ràng buộc hạng-1 (*Rank-1 Constraint System* – **R1CS**)

Xét ví dụ sau: Cho hàm số được định nghĩa bởi

```
def f(x):
  return x**3 + x + 5
```

Biểu diễn hàm f dưới dạng danh sách các chỉ thị có dạng

c = a op b,

với op kí hiệu toán tử + hay *, ta có

```
def f(x):
  sym_1 = x * x
  y = sym_1 * x
```

sym_2 = y + x

out = sym_2 + 5

return out.

Véc-tơ ánh xạ các biến: Đặt (one, x, out, sym_1, y, sym_2) là véc-tơ ánh xạ biến,

Mã lệnh	C	A	B
sym_1 = x * x	(0, 0, 0, 1, 0, 0)	(0, 1, 0, 0, 0, 0)	(0, 1, 0, 0, 0, 0)
y = sym_1 * x	(0, 0, 0, 0, 1, 0)	(0, 0, 0, 1, 0, 0)	(0, 1, 0, 0, 0, 0)
sym_2 = y + x	(0, 0, 0, 0, 0, 1)	(0, 1, 0, 0, 1, 0)	(1, 0, 0, 0, 0, 0)
out = sym_2 + 5	(0, 0, 1, 0, 0, 0)	(5, 0, 0, 0, 0, 1)	(1, 0, 0, 0, 0, 0)

Chẳng hạn, cho x = 3, ta có w = (1, 3, 25, 9, 27, 30) là nghiệm hay véc-tơ bằng chứng thỏa đẳng thức

$A * w^T . B * w^T = C * w^T$,

Trong đó,

$$A = \begin{bmatrix} 0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0 \\ 5\,0\,0\,0\,0\,1 \end{bmatrix}, B = \begin{bmatrix} 0\,1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0 \end{bmatrix}, C = \begin{bmatrix} 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,0 \end{bmatrix}.$$

Từ hệ ràng buộc hạng-1 đến chương trình số học bậc hai

Sử dụng ngôn ngữ nội suy để biến đổi các ma trận A, B, và C có kích thước mxn, thành n đa thức bậc (m – 1),

$\{A_j(x), B_j(x), C_j(x)\}_{j=1}^{n}$, trong đó $A_j(i) = A_{i,j}, i = 1, \dots, m$.

Ta cũng định nghĩa:

$T(x) = (x-1)(x-2) \dots (x-m)$.

Chẳng hạn, ta sẽ tìm $A_1(x), \dots, A_6(x) \in \mathbb{Z}_{29}[x]$ thỏa mãn công thức nội suy Lagrange:

$A_1(1) = 0, A_1(2) = 0, A_1(3) = 0, A_1(4) = 5$

$A_2(1) = 1, A_2(2) = 0, A_2(3) = 1, A_2(4) = 0$

$A_3(1) = 0, A_3(2) = 0, A_3(3) = 0, A_3(4) = 0$

$A_4(1) = 0, A_4(2) = 1, A_4(3) = 0, A_4(4) = 0$

$A_5(1) = 0, A_5(2) = 0, A_5(3) = 1, A_5(4) = 0$

$A_6(1) = 0, A_6(2) = 0, A_6(3) = 0, A_6(4) = 1$

Thi hành đoạn mã Python trên trình Sage Math Cell,

R = GF(29)['x']

A = [

[(1, 0), (2, 0), (3, 0), (4, 5)],

[(1, 1), (2, 0), (3, 0), (4, 0)],

[(1, 0), (2, 0), (3, 0), (4, 0)],

[(1, 0), (2, 0), (3, 1), (4, 0)],

[(1, 0), (2, 0), (3, 0), (4, 1)]

]

for i in range(len(A)):

Ai = R.langrange – polynomial(A[i])

Print('A{}(x) = {}''.format(i + 1, Ai)

Ta được

$A_1(x) = 25*x^3 + 24*x^2 + 14*x + 24$

$A_2(x) = 9*x^3 + 5*x^2 + 8*x + 8$

$A_3(x) = 0$

$A_4(x) = 15*x^3 + 25*x^2 + 24*x + 23$

$A_5(x) = 14*x^3 + 18*x^2 + 22*x + 4$

$A_6(x) = 5*x^3 + 28*x^2 + 26*x + 28$

Thực hiện tương tự để được các đa thức $B_i(x)$ và $C_i(x)$, i = 1, 2, 3, 4, 5, 6.

Trình số học bậc 2

Một trình bậc 2 – **QAP** (*Quadratic Arithmetic Program*) được định nghĩa bởi

$R = \{F < m, \{A_i(x), B_i(x), C_i(x) \in F[x]\}_{i=0}^{m}, T(x) \in F[x]\}.$

Ta nói $w = (w_0, \ldots, w_m) \in F^{m+1}$ thỏa một QAP nếu

$A_w(x)B_w(x) \equiv C_w(x) \pmod{T(x)}$

hay

$A_w(x)B_w(x) \equiv C_w(x) + D_w(x)T(x))$

Hay

$A_w(x)B_w(x) \equiv C_w(x) + D_w(x)T(x)$

với

$A_w(x) = \sum_{i=0}^{m} w_i A_i(x)\ ; B_w(x) = \sum_{i=1}^{m} w_i B_i(x)\ ; C_w(x) = \sum_{i=1}^{m} w_i\, C_i(x),$

phụ thuộc bằng chứng $w = (w_{[:l]}, w_{[l+1:]}) \in F^{m+1}$,

trong đó, $w_{[:l]}$ và $w_{[l:]}$ lần lượt ký hiệu đầu vào (công khai cho mọi người) và đầu vào (cho người chứng minh).

Tri thức trị 0 cho trình số học bậc hai

Từ các đa thức $A_w(x)$, $B_w(x)$, và $C_w(x)$, ta định nghĩa các đa thức sau:

$A(x) = \alpha + A_w(x) + r\delta,$

$B(x) = \beta + B_w(x) + s\delta,$

$C(x) = A(x)B(x) = \alpha B_w(x) + \beta A_w(x) + A_w(x)B_w(x) + R(x)$

= ($\alpha B_w(x) + \beta A_w(x)\, C_w(x)$) + $D_w(x)T(x) + R(x)$,

Đặt,

$L_w(x) = \sum_{i=0}^{m} L_i(x) = \sum_{i=0}^{m} w_i(\alpha B_i(x) + \beta A_i(x) + C_i(x)$,

thì

$C(x) = L_w(x) + D_w(x)T(x) + R(x)$,

Trong đó

$R(x) = \alpha\beta + \delta(s(\alpha + A_w(x)) + r(B_w(x) + \beta) + rs\delta)$

$= \alpha\beta + \delta(s(A(x) - r\delta) + r(B(x) - s\delta) + rs\delta)$

$= \alpha\beta + \delta(sA(x) + rB(x) - rs\delta)$.

Ta thấy, w thỏa

$A_w(x)B_w(x) + D_w(x)T(x)$,

nếu chỉ nếu

$A(x) = \alpha + A_w(x) + r\delta$,

$B(x) = \beta + B_w(x) + s\delta$,

$C(x) = A(x)B(x) + L_w(x) + D_w(x)T(x) + \delta(sA(x) + rB(x) - rs\delta)$

$= \alpha\beta + \delta C_p(x) + \gamma C_V(x)$,

Trong đó,

$C_p(x) = \delta^{-1}\left(D_w(x)T(x) + L_{w[l+1:]}(x)\right) + sA(x) + rB(x) - rs\delta$,

$C_V(x) = \gamma^{-1} L_{w[:l]}(x)$,

$L_w(x) = \sum_{i=0}^{m} w_i(B_i(x) + \beta A_i(x) + C_i(x))$

$= L_{w[:l]}(x) + L_{w[l+1:]}(x)$.

với $L_{w[:l]}(x)$ ký hiệu đầu vào công khai cho mọi người, và $L_{w[l+1:]}(x)$ là đầu vào cho người chứng minh.

Ta có một số lưu ý sau:

– Vai trò của α và β nhằm đảm bảo các đa thức A(x), B(x), và C(x) sử dụng cùng một véc-tơ w.

– Vai trò của γ và δ để chống các tấn công trộn và đối sánh bằng cách tạo $C_P(x)$ và $C_V(x)$ độc lập tuyến tính với tích $\alpha\beta$.

– r và s là những thừa số do người chứng minh tạo ngẫu nhiên cho mỗi bằng chứng tri thức trị 0, gọi là các thừa số mù.

– Phát biểu

$R = \left(p, G_1, G_2, G_3, e, l, \{A_i(x), B_i(x), C_i(x)\}_{i=0}^{m}, T(x)\right)$,

với

$e: G_1 \times G_2 \to G_3$

là cặp ghép song tuyến tính, nghĩa là $e(G^x, H^y) = g^{xy}$.

- Thiết lập

$Setup(R) \to \sigma: G \leftarrow G_1, H \leftarrow G_2, \alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p^*,$

tính

$\sigma = \left(G_\alpha, G_\beta, H_\beta, H_\gamma, G_\delta, H_\delta, g_{\alpha\beta}, \boldsymbol{G}_\tau, \boldsymbol{H}_\tau, \boldsymbol{G}_\gamma, \boldsymbol{G}_\delta, \boldsymbol{G}_T\right)$

$\left(G^\alpha, G^\beta, H^\beta, H^\gamma, G^\delta, H^\delta, e\left(G^\alpha, H^\beta\right), \left(G^{\tau^i}\right)_{i=0}^{n-1}, \left(H^{\tau^i}\right)_{i=0}^{n-1},\right.$

$\left.\left(G^{\gamma^{-1}L_i(\tau)}\right)_{i=0}^{l}, \left(G^{\delta^{-1}L_i(\tau)}\right)_{i=l+1}^{m}, \left(G^{\delta^{-1}\tau^i T(\tau)}\right)_{i=0}^{n-2}\right).$

- Chứng minh

$Prove(R, \sigma, w) \to \pi: r, s \leftarrow \mathbb{Z}_p,$

tính

$\pi = A, B, C,$

Trong đó,

$A = G_\alpha \boldsymbol{G}_T^{A_w} G_\delta^r = G^{\alpha + A_w(\tau) + r\delta} \in G_1,$

$B = H_\beta \boldsymbol{H}_\tau^{B_w} H_\delta^s = H^{\beta + B_w(\tau) + s\delta} \in G_2,$

$C = \boldsymbol{G}_\delta^{w[l+1:]} G_\alpha^s \boldsymbol{G}_\tau^{A_w} G_\beta^r \boldsymbol{G}_\tau^{B_w} \boldsymbol{G}_T^{D_w} G_\delta^{rs}$

$= G^{\delta^{-1}\left(L_w[l+1:] + D_w(\tau)T(\tau)\right) + s\left(\alpha + A_w(\tau)\right) + r\left(\beta + B_w(\tau)\right) + rs\delta}$

$= G^{C_P(\tau)} \in G_1.$

- Xác minh

$Verify\left(\delta, \pi, w_{[l:]}\right) \to g_{\alpha\beta} \,.\, e\left(\boldsymbol{G}_\gamma^{W_{[:l]}}, H_\gamma\right).\, e(C, H_\delta).$

Ta thấy, quá trình chứng minh,

Vế trái: $LHS = e(A, B) = e\left(G^{A(\tau)}, H^{B(\tau)}\right) = g^{A(\tau)B(\tau)} \in G_3.$

Vế phải: $RHS = g_{\alpha\beta}.\, e\left(\boldsymbol{G}_\tau^{W_{[:l]}}, H_\gamma\right).\, e(C, H_\delta)$

$= e\left(G^\alpha, H^\beta\right).\, e\left(G^{\gamma^{-1}L_{w[:l]}(\tau)}, H^\gamma\right).\, e\left(G^{C_P(\tau)}, H^\delta\right)$

$= g^{\alpha\beta + \gamma C_V(\tau) + \delta C_P(\tau)}$

$= g^{C(\tau)} \in G_3.$

Ta thấy vế trái bằng với vế phải, LHS = RHS, suy ra $A(\tau)B(\tau) = C(\tau)$. Bằng chứng được chấp nhận.