TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT

KHOA CÔNG NGHỆ THÔNG TIN

----------o0o----------

BÁO CÁO SINH HOẠT HỌC THUẬT

Đề tài: Tìm hiểu công nghệ Ví trong Bitcoin

Người báo cáo : Nguyễn Thu Hằng

Đơn vị : Bộ môn Tin học Kinh tế

Hà Nội, 12/ 2023

MỤC LỤC

Contents

1 TỔNG QUAN VỀ CÔNG NGHỆ VÍ

1.1 Ví Bitcoin:

Từ "ví" được sử dụng để mô tả một vài vấn đề khác nhau trong bitcoin. Ở cấp độ cao, ví là một ứng dụng đóng vai trò là giao diện người dùng chính. Ví kiểm soát quyền truy cập vào tiền của người dùng, quản lý khóa và địa chỉ, theo dõi số dư, tạo và ký giao dịch. Hẹp hơn, từ quan điểm của một lập trình viên, từ "ví" đề cập đến cấu trúc dữ liệu được sử dụng để lưu trữ và quản lý khóa của người dùng.

Trong phần này, chúng ta sẽ xem xét ý nghĩa thứ hai, trong đó ví là nơi chứa “private key”, thường được triển khai dưới dạng tệp có cấu trúc hoặc cơ sở dữ liệu đơn giản. Trong đó, các công nghệ khác nhau được tóm tắt khi sử dụng để xây dựng ví bitcoin thân thiện với người dùng, an toàn và linh hoạt.

Một quan niệm sai lầm phổ biến về bitcoin là ví bitcoin có chứa bitcoin. Trên thực tế, ví chỉ chứa key. Các "coin" được ghi lại trong blockchain trên mạng Bitcoin. Người dùng kiểm soát các đồng tiền trên mạng bằng cách ký các giao dịch bằng các khóa trong ví của họ. Theo một nghĩa nào đó, ví bitcoin là một chuỗi khóa (keychain).

Ví Bitcoin chứa key, không phải chứa coin. Mỗi người dùng có một ví chứa key. Ví thực sự là chuỗi khóa chứa các cặp key tư / công khai ([private_public_keys]). Người dùng ký giao dịch bằng các khóa, qua đó chứng minh họ sở hữu đầu ra giao dịch (tiền của họ). Các đồng tiền được lưu trữ trên blockchain dưới dạng đầu ra giao dịch (thường được ghi nhận là vout hoặc txout).

Có hai loại ví chính, được phân biệt bằng việc các khóa mà chúng chứa có liên quan đến nhau hay không.

Loại đầu tiên là ví không xác định (nondeterministic wallet), trong đó mỗi khóa được tạo độc lập từ một số ngẫu nhiên. Các khóa liên quan đến nhau. Loại ví này còn được gọi là ví JBOK từ cụm từ "Just a Bunch Of Keys".

Loại ví thứ hai là ví xác định (deterministic wallet), trong đó tất cả các khóa có nguồn gốc từ một khóa chính (master key) duy nhất, được gọi là seed (seed). Tất cả các khóa trong loại ví này có liên quan với nhau và có thể được tạo lại nếu một khóa có seed ban đầu. Có một số phương pháp phái sinh khóa (key derivation) khác nhau được sử dụng trong ví xác định. Phương pháp phái sinh được sử dụng phổ biến nhất sử dụng cấu trúc hình cây (tree-like structure) và được gọi là ví xác định phân cấp hoặc ví HD (Hierarchical Deterministic).

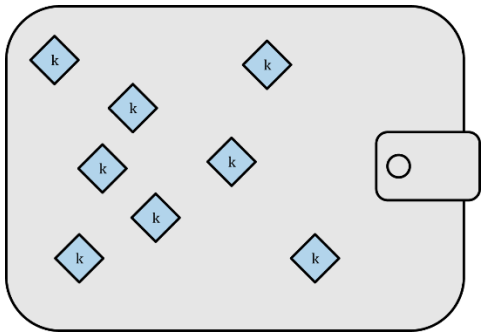
Ví xác định được khởi tạo từ một chuỗi ngẫu nhiên (entropy). Để làm cho những thứ này dễ sử dụng hơn, các chuỗi ngẫu nhiên được mã hóa dưới dạng các từ tiếng Anh, còn được gọi là các từ mã ghi nhớ (mnemonic code words).

Ví không xác định (ngẫu nhiên) (Nondeterministic (Random) Wallets)

Trong ví bitcoin đầu tiên (bây giờ được gọi là Bitcoin Core), ví là tập hợp các private key được tạo ngẫu nhiên. Ví dụ: máy khách Bitcoin Core ban đầu tạo trước 100 private key ngẫu nhiên khi mới bắt đầu và tạo nhiều khóa hơn khi cần, chỉ sử dụng mỗi khóa một lần. Những ví như vậy đang được thay thế bằng ví xác định vì chúng cồng kềnh để quản lý, sao lưu và nhập. Nhược điểm của các khóa ngẫu nhiên là nếu tạo nhiều trong số chúng thì phải giữ bản sao của tất cả chúng, có nghĩa là ví phải được sao lưu thường xuyên. Mỗi khóa phải được sao lưu hoặc số tiền mà nó kiểm soát sẽ bị mất không thể hủy ngang nếu ví không thể truy cập được. Điều này mâu thuẫn trực tiếp với nguyên tắc tránh sử dụng lại địa chỉ, bằng cách sử dụng mỗi địa chỉ Bitcoin chỉ cho một giao dịch. Việc sử dụng lại địa chỉ làm giảm quyền riêng tư bằng cách liên kết nhiều giao dịch và địa chỉ với nhau. Ví không xác định Loại 0 là một lựa chọn không tốt của ví, đặc biệt nếu muốn tránh sử dụng lại địa chỉ vì nó có nghĩa là quản lý nhiều khóa, điều này tạo ra nhu cầu sao lưu thường xuyên. Mặc dù ứng dụng khách hàng Bitcoin Core bao gồm ví Type-0, nhưng việc sử dụng ví này không được khuyến khích bởi các nhà phát triển Bitcoin Core. Ví không xác định (ngẫu nhiên) loại 0: một tập hợp các khóa được tạo ngẫu nhiên cho thấy một ví không xác định, chứa một bộ sưu tập lỏng lẻo các khóa ngẫu nhiên.

Việc sử dụng ví không xác định không được khuyến khích cho bất cứ điều gì khác ngoài các bài kiểm tra đơn giản. Chúng chỉ đơn giản là quá cồng kềnh

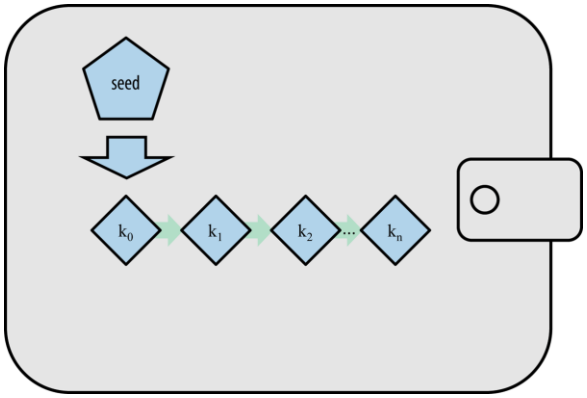
để sao lưu và sử dụng. Thay vào đó, hãy sử dụng ví HD dựa trên tiêu chuẩn ngành với chuỗi ngẫu nhiên ghi nhớ (entropy hoặc "initial seed") để sao lưu.



Hình 1. Ví không xác định (ngẫu nhiên) loại 0: tập hợp các khóa được tạo ngẫu nhiên

❖ Ví xác định (seed) - Deterministic (Seeded) Wallets

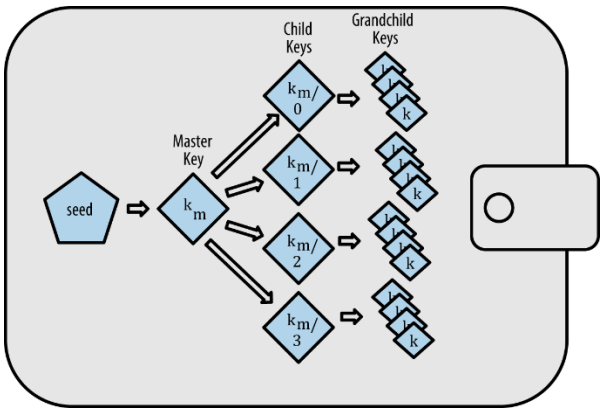
Ví xác định, hoặc "seeded", là ví chứa các private key đều có nguồn gốc từ một seed chung, thông qua việc sử dụng hàm băm một chiều (one-way hash function). Seed là một số được tạo ngẫu nhiên được kết hợp với các dữ liệu khác, chẳng hạn như số chỉ mục hoặc "mã chuỗi – chain code" (xem Ví HD (BIP-32 / BIP-44)) để lấy private key. Trong một ví xác định, seed là đủ để khôi phục tất cả các khóa dẫn xuất, và do đó một bản sao lưu duy nhất tại thời điểm tạo là đủ. Seed cũng đủ để xuất hoặc nhập ví, cho phép dễ dàng di chuyển tất cả các khóa của người dùng giữa các triển khai ví khác nhau. Ví xác định loại 1 (seed): một chuỗi các khóa xác định có nguồn gốc từ seed cho thấy sơ đồ logic của ví xác định.



Hình 2. Ví xác định loại 1 (seed): một chuỗi các khóa xác định có nguồn gốc từ seed

❖ Ví HD (BIP-32/BIP-44)

Ví xác định được phát triển để giúp dễ dàng lấy được nhiều khóa từ một "seed" duy nhất. Hình thức tiên tiến nhất của ví xác định là ví HD được xác định bởi tiêu chuẩn BIP-32. Ví HD chứa các khóa có nguồn gốc từ cấu trúc cây, sao cho khóa cha có thể lấy được một chuỗi các khóa con, mỗi khóa con có thể lấy được một chuỗi các khóa cháu, v.v., đến độ sâu vô hạn. Cấu trúc cây này được minh họa trong ví HD Loại 2: một cây khóa được tạo từ một seed duy nhất.



Hình 3. Ví HD loại 2: một cây khóa được tạo từ một seed duy nhất

Ví HD cung cấp hai lợi thế lớn so với các khóa ngẫu nhiên (không xác định). Đầu tiên, cấu trúc cây có thể được sử dụng để thể hiện ý nghĩa tổ chức bổ sung, chẳng hạn như khi một nhánh cụ thể của khóa con được sử dụng để nhận các khoản thanh toán đến và một nhánh khác được sử dụng để nhận thay đổi từ các khoản thanh toán đi. Các nhánh khóa cũng có thể được sử dụng trong tổ chức công ty, phân bổ các chi nhánh khác nhau cho các phòng ban, công ty con, chức năng cụ thể hoặc danh mục kế toán.

Ưu điểm thứ hai của ví HD là người dùng có thể tạo một chuỗi khóa công khai mà không cần truy cập vào các private key tương ứng. Điều này cho phép ví HD được sử dụng trên một máy chủ không an toàn hoặc trong khả năng chỉ nhận, phát hành một khóa công khai khác nhau cho mỗi giao dịch. Các khóa công khai không cần phải được tải sẵn hoặc được tạo trước, nhưng máy chủ không có các private key để có thể giao dịch tiền.

❖ Seed và mã ghi nhớ (BIP-39)

Ví HD là một cơ chế rất mạnh mẽ để quản lý nhiều khóa và địa chỉ. Chúng thậm chí còn hữu ích hơn nếu chúng được kết hợp với một tiêu chuẩn hóa để tạo

seed từ một chuỗi các từ tiếng Anh dễ phiên âm, xuất và nhập qua ví. Điều này được gọi là ghi nhớ - mnemonic và tiêu chuẩn được xác định bởi BIP-39. Ngày nay, hầu hết các ví bitcoin (cũng như ví cho các loại tiền điện tử khác) sử dụng tiêu chuẩn này và có thể nhập và xuất seed để sao lưu và phục hồi bằng cách sử dụng các ghi nhớ có thể tương tác.

Hãy xem xét điều này từ góc độ thực tế. Seed nào sau đây dễ sao chép, ghi lại trên giấy, đọc mà không có lỗi, xuất và nhập vào ví khác?

Một seed cho một ví xác định, trong hex

0C1E24E5917779D297E14D45F14E1A1A

Một seed cho một ví xác định, từ một ghi nhớ 12 từ

army van defense carry jealous true

garbage claim echo media make crunch

1.2 Các phương pháp phổ biến nhất về ví

Khi công nghệ ví bitcoin đã phát triển, một số tiêu chuẩn công nghiệp phổ biến nhất định đã xuất hiện giúp ví bitcoin có thể tương tác rộng rãi, dễ sử dụng, an toàn và linh hoạt. Những tiêu chuẩn chung này là:

- Các từ mã ghi nhớ (Mnemonic code words), dựa trên BIP-39
- Ví HD (HD wallets), dựa trên BIP-32
- Cấu trúc ví HD đa năng (Multipurpose HD wallet structure), dựa trên BIP-43
- Ví đa tiền tệ và đa tài khoản (Multicurrency and multiaccount wallets), dựa trên BIP-44

Các tiêu chuẩn này có thể thay đổi hoặc có thể trở nên lỗi thời bởi sự phát triển trong tương lai, nhưng hiện tại chúng tạo thành một tập hợp các công nghệ lồng vào nhau đã trở thành tiêu chuẩn ví thực tế cho bitcoin.

Các tiêu chuẩn đã được áp dụng bởi một loạt các ví bitcoin phần mềm và phần cứng, làm cho tất cả các ví này có thể tương tác. Người dùng có thể xuất mnemonic được tạo trên một trong các ví này và nhập nó vào một ví khác, khôi phục tất cả các giao dịch, khóa và địa chỉ.

Một số ví dụ về ví phần mềm hỗ trợ các tiêu chuẩn này bao gồm (được liệt kê theo thứ tự bảng chữ cái) Bluewallet, Breadwallet, Copay và Multibit HD. Ví dụ về ví phần cứng hỗ trợ các tiêu chuẩn này bao gồm (được liệt kê theo thứ tự bảng chữ cái) KeepKey, Ledger và Trezor.

Các phần sau đây kiểm tra chi tiết từng công nghệ này.

Nếu đang triển khai ví bitcoin, nó nên được xây dựng dưới dạng ví HD, với seed có nguồn gốc từ và được mã hóa dưới dạng mã ghi nhớ để sao lưu, tuân theo các tiêu chuẩn BIP-32, BIP-39, BIP-43 và BIP-44, như được mô tả trong các phần sau.

❖ Sử dụng ví Bitcoin

Gabriel, một thiếu niên trẻ ở Rio de Janeiro, người đang điều hành một cửa hàng trực tuyến đơn giản bán áo phông, cốc cà phê và nhãn dán mang nhãn hiệu bitcoin. Gabriel sử dụng ví phần cứng Trezor bitcoin (Thiết bị Trezor: ví HD bitcoin trong phần cứng) để quản lý an toàn bitcoin của mình. Trezor là một thiết bị USB đơn giản với hai nút lưu trữ khóa (dưới dạng ví HD) và ký các giao dịch. Ví Trezor thực hiện tất cả các tiêu chuẩn ngành được thảo luận trong chương này, vì vậy Gabriel không phụ thuộc vào bất kỳ công nghệ độc quyền hoặc giải pháp nhà cung cấp duy nhất nào.

Hình 4. Thiết bị Trezor: ví HD bitcoin trong phần cứng

Khi Gabriel sử dụng Trezor lần đầu tiên, thiết bị đã tạo ra một chuỗi ngẫu nhiên (entropy), ghi nhớ liên quan và lấy seed từ bộ tạo số ngẫu nhiên phần cứng tích hợp. Trong giai đoạn khởi tạo này, ví hiển thị một chuỗi các từ được đánh số, từng từ một, trên màn hình (xem Trezor hiển thị một trong các mnemonic).

Hình 5. Trezor hiển thị một trong những mnemonic

Bằng cách viết ra bản ghi nhớ này, Gabriel đã tạo ra một bản sao lưu (xem bản sao lưu ghi nhớ bằng giấy của Gabriel) có thể được sử dụng để khôi phục trong trường hợp mất mát hoặc hư hỏng thiết bị Trezor. Ghi nhớ này có thể được sử dụng để khôi phục trong Trezor mới hoặc trong bất kỳ ví phần mềm hoặc phần cứng tương thích nào. Lưu ý rằng chuỗi từ rất quan trọng, vì vậy các bản sao lưu giấy ghi nhớ có khoảng trắng được đánh số cho mỗi từ. Gabriel đã phải cẩn thận ghi lại từng từ trong không gian được đánh số để bảo toàn trình tự chính xác.

Bảng 1. Bản sao lưu giấy của Gabriel về ghi nhớ

1.	*army*	7.	*garbage*
2.	*van*	8.	*claim*
3.	*defense*	9.	*echo*
4.	*carry*	10.	*media*
5.	*jealous*	11.	*make*
6.	*true*	12.	*crunch*

Ghi chú: Một ghi nhớ 12 từ được hiển thị trong bản sao lưu giấy của Gabriel về ghi nhớ, để đơn giản. Trên thực tế, hầu hết các ví phần cứng đều tạo ra một ghi nhớ 24 từ an toàn hơn. Ghi nhớ được sử dụng theo cùng một cách, bất kể độ dài.

Đối với lần triển khai đầu tiên của cửa hàng trực tuyến của mình, Gabriel sử dụng một địa chỉ Bitcoin duy nhất, được tạo trên thiết bị Trezor của mình. Địa chỉ duy nhất này được sử dụng bởi tất cả khách hàng cho tất cả các đơn đặt hàng. Như chúng ta sẽ thấy, cách tiếp cận này có một số nhược điểm và có thể được cải thiện với ví HD.

2 Chi tiết công nghệ ví

Bây giờ chúng ta hãy xem xét từng tiêu chuẩn ngành quan trọng được sử dụng chi tiết bởi nhiều ví bitcoin.

2.1 Từ mã ghi nhớ (BIP-39) - Mnemonic Code Words (BIP-39)

Các từ mã ghi nhớ là các chuỗi từ đại diện (mã hóa - encode) một số ngẫu nhiên được sử dụng làm seed để lấy ví xác định. Chuỗi từ là đủ để tạo lại seed và từ đó tạo lại ví và tất cả các khóa dẫn xuất. Một ứng dụng ví thực hiện ví xác định với các mnemonic sẽ hiển thị cho người dùng một chuỗi từ 12 đến 24 từ khi lần đầu tiên tạo ví. Chuỗi từ đó là bản sao lưu ví và có thể được sử dụng để khôi phục và tạo lại tất cả các khóa trong cùng một hoặc bất kỳ ứng dụng ví tương thích nào. Các mnemonic giúp người dùng sao lưu ví dễ dàng hơn vì chúng dễ đọc và phiên âm chính xác, so với một chuỗi số ngẫu nhiên.

Các mnemonic Mnemonic thường bị nhầm lẫn với " brainwallets". Chúng không giống nhau. Sự khác biệt chính là brainwallets bao gồm các từ do người dùng chọn, trong khi các Mnemonic được tạo ngẫu nhiên bởi ví và được trình bày cho người dùng. Sự khác biệt quan trọng này làm cho các mnemonic an toàn hơn nhiều, bởi vì con người là nguồn ngẫu nhiên rất kém.

Mã ghi nhớ được định nghĩa trong BIP-39. Lưu ý rằng BIP-39 là một triển khai của tiêu chuẩn mã ghi nhớ. Có một tiêu chuẩn khác, với một bộ từ khác, được sử dụng bởi ví Electrum và trước BIP-39. BIP-39 được đề xuất bởi công ty đứng sau ví phần cứng Trezor và không tương thích với việc triển khai của Electrum. Tuy nhiên, BIP-39 hiện đã đạt được sự hỗ trợ rộng rãi của ngành trên hàng chục triển khai có thể tương tác và nên được coi là tiêu chuẩn công nghiệp thực tế.

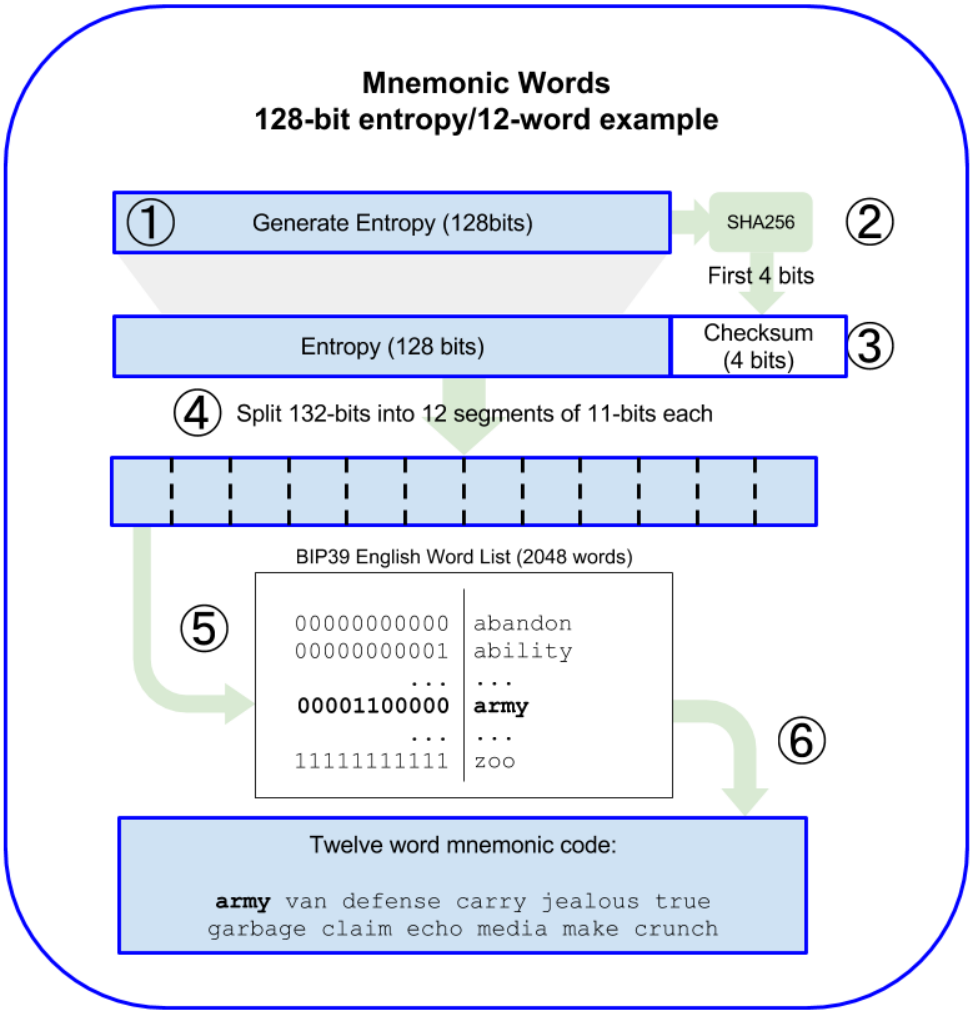
BIP-39 định nghĩa việc tạo mã ghi nhớ và seed, mà chúng tôi mô tả ở đây trong chín bước. Để rõ ràng, quá trình này được chia thành hai phần: các bước từ 1 đến 6 được hiển thị trong Tạo các mnemonic và các bước từ 7 đến 9 được hiển thị trong Mnemonic đến seed.

- *Tạo mnemonic - Generating mnemonic words*

Các mnemonic được tạo tự động bởi ví bằng cách sử dụng quy trình chuẩn hóa được xác định trong BIP-39. Ví bắt đầu từ một nguồn entropy, thêm tổng kiểm tra - checksum và sau đó ánh xạ entropy vào danh sách từ:

1. Tạo một chuỗi ngẫu nhiên (entropy) từ 128 đến 256 bit.
2. Tạo Checksum của chuỗi ngẫu nhiên bằng cách lấy các bit đầu tiên (độ dài entropy/32) của hàm băm SHA256.
3. Thêm Checksum vào cuối chuỗi ngẫu nhiên.
4. Chia kết quả thành các đoạn có độ dài 11 bit.
5. Ánh xạ mỗi giá trị 11 bit thành một từ trong từ điển được xác định trước gồm 2048 từ.
6. Mã ghi nhớ là chuỗi từ.

- Tạo entropy và mã hóa dưới dạng các mnemonic cho thấy entropy được sử dụng như thế nào để tạo ra các mnemonic.



Hình 6. Tạo entropy và mã hóa dưới dạng các mnemonic

- Mã ghi nhớ: entropy và độ dài từ cho thấy mối quan hệ giữa kích thước của dữ liệu entropy và độ dài của mã ghi nhớ trong từ.

Bảng 2. Mã ghi nhớ: entropy và độ dài từ

Entropy (bits)	Checksum (bits)	Entropy + checksum (bits)	Mnemonic length (words)
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

- *Mnemonic đến seed*

Các mnemonic đại diện cho entropy với độ dài từ 128 đến 256 bit. Entropy sau đó được sử dụng để lấy một seed dài hơn (512-bit) thông qua việc sử dụng key-stretching function PBKDF2. Seed được tạo ra sau đó được sử dụng để xây dựng một ví xác định và lấy được key của nó.

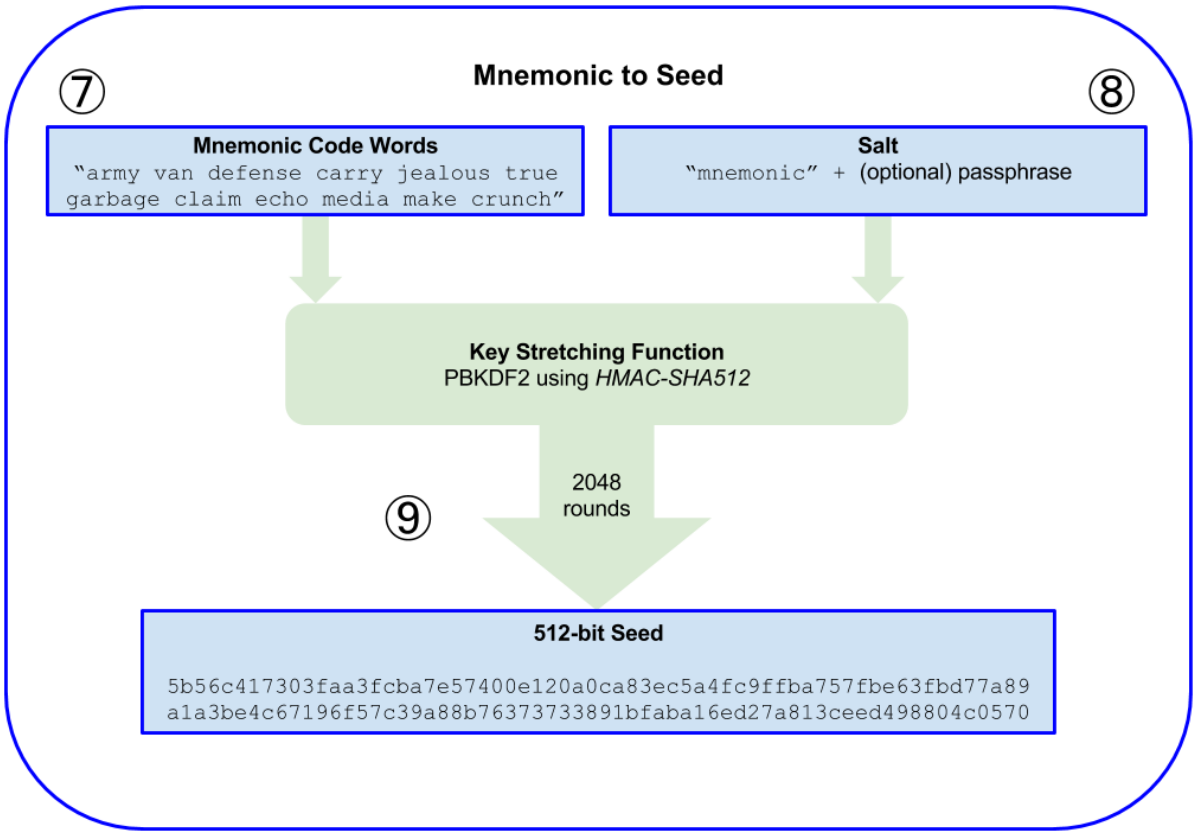
Key-stretching function có hai tham số: mnemonic và salt. Mục đích của salt trong chức năng kéo dài phím là gây khó khăn cho việc xây dựng bảng tra cứu cho phép tấn công brute-force attack. Trong tiêu chuẩn BIP-39, salt có một mục đích khác - nó cho phép giới thiệu cụm mật khẩu đóng vai trò là yếu tố bảo mật bổ sung bảo vệ seed, như chúng tôi sẽ mô tả chi tiết hơn trong Cụm mật khẩu tùy chọn trong BIP-39.

Quá trình được mô tả trong các bước từ 7 đến 9 tiếp tục từ quy trình được mô tả trước đó trong Tạo các mnemonic:

7. Tham số đầu tiên của hàm tạo khóa bí mật từ mật khẩu PBKDF2 là mnemonic được tạo ra từ bước 6.
8. Tham số thứ hai của hàm tạo khóa bí mật từ mật khẩu PBKDF2 là salt. Salt bao gồm hằng số chuỗi "mnemonic" được nối với chuỗi cụm mật khẩu tùy chọn do người dùng cung cấp.

9. PBKDF2 hàm tạo khóa bí mật từ mật khẩu các tham số ghi nhớ và salt bằng cách sử dụng 2048 vòng băm với thuật toán HMAC-SHA512, tạo ra giá trị 512 bit làm đầu ra cuối cùng. Giá trị 512 bit đó là seed.

Mnemonic đến seed cho thấy cách ghi nhớ được sử dụng để tạo seed.



Hình 7. Mnemonic đến seed

Hàm tạo khóa bí mật từ mật khẩu, với 2048 vòng băm, là một biện pháp bảo vệ rất hiệu quả chống lại các brute-force attack chống lại ghi nhớ hoặc cụm mật khẩu. Nó làm cho nó cực kỳ tốn kém (trong tính toán) để thử hơn một vài nghìn cụm mật khẩu và kết hợp ghi nhớ, trong khi số lượng tạo ra seed có thể là rất lớn (2512).

Bảng #mnemonic_128_no_pass, #mnemonic_128_w_pass và #mnemonic_256_no_pass hiển thị một số ví dụ về mã ghi nhớ và seed mà chúng tạo ra (có hoặc không có cụm mật khẩu).

Bảng 3. Mã ghi nhớ entropy 128 bit, không có cụm mật khẩu, seed kết quả

Entropy input (128 bits)	0c1e24e5917779d297e14d45f14e1a1a
Mnemonic (12 words)	army van defense carry jealous true garbage claim echo media make crunch
Passphrase	(none)
Seed (512 bits)	5b56c417303faa3fcba7e57400e120a0ca83ec5a4fc9ffba757fbe63fbd77a89a1a3be4c67196f57c39 a88b76373733891bfaba16ed27a813ceed498804c0570

Bảng 4. Mã ghi nhớ entropy 128 bit, với cụm mật khẩu, seed kết quả:

Entropy input (128 bits)	0c1e24e5917779d297e14d45f14e1a1a
Mnemonic (12 words)	army van defense carry jealous true garbage claim echo media make crunch
Passphrase	SuperDuperSecret
Seed (512 bits)	3b5df16df2157104cfdd22830162a5e170c0161653e3afe6c88defeefb0818c793dbb28ab3ab091897d0 715861dc8a18358f80b79d49acf64142ae57037d1d54

Bảng 5. Mã ghi nhớ entropy 256 bit, không có cụm mật khẩu, seed kết quả

Entropy input (256 bits)	2041546864449caff939d32d574753fe684d3c947c3346713dd8423e74abcf8c
Mnemonic (24 words)	cake apple borrow silk endorse fitness top denial coil riot stay wolf luggage oxygen faint major edit measure invite love trap field dilemma oblige
Passphrase	(none)
Seed (512 bits)	3269bce2674acbd188d4f120072b13b088a0ecf87c6e4cae41657a0bb78f5315b33b3a04356e53d062e5 5f1e0deaa082df8d487381379df848a6ad7e98798404

Nhiều ví không cho phép tạo ví với nhiều hơn một cụm mnemonic 12 từ. sẽ nhận thấy từ các bảng trên rằng mặc dù độ dài duy nhất của đầu vào entropy, kích thước seed vẫn giữ nguyên (512 bit). Từ góc độ bảo mật, lượng entropy thực sự được sử dụng để sản xuất ví HD là khoảng 128 bit, tương đương với 12 từ. Cung cấp hơn 12 từ tạo ra entropy bổ sung là không cần thiết và entropy không sử dụng này không được sử dụng cho dẫn xuất của seed theo cách mà người ta có thể nghi ngờ ban đầu. Từ góc độ khả năng sử dụng, 12 từ cũng dễ dàng hơn để viết ra, sao lưu và lưu trữ.

- *Cụm mật khẩu tùy chọn trong BIP-39*

Tiêu chuẩn BIP-39 cho phép sử dụng cụm mật khẩu tùy chọn trong dẫn xuất của seed. Nếu không có cụm mật khẩu nào được sử dụng, ghi nhớ được tạo ra bằng một salt bao gồm chuỗi không đổi "ghi nhớ", tạo ra một seed 512 bit cụ thể từ bất kỳ ghi nhớ nhất định nào. Nếu một cụm mật khẩu được sử dụng, chống lại các brute-force attack chống lại ghi nhớ hoặc cụm mật khẩu. Nó làm cho nó cực kỳ tốn kém (trong tính toán) để thử hơn một vài nghìn cụm mật khẩu và kết hợp ghi nhớ, trong khi số lượng tạo ra seed có thể là rất lớn (2512) sẽ tạo ra một seed khác với cùng một ghi nhớ đó. Trên thực tế, với một ghi nhớ duy nhất, mọi cụm mật khẩu có thể dẫn đến một seed khác nhau. Về cơ bản, không có cụm mật khẩu "sai". Tất cả các cụm mật khẩu đều hợp lệ và tất cả chúng đều dẫn đến các seed khác nhau, tạo thành một tập hợp lớn các ví chưa được khởi tạo có thể. Tập hợp các ví có thể quá lớn (2512) đến nỗi không có khả năng thực tế để tấn công hoặc vô tình đoán được một ví đang được sử dụng.

Không có cụm mật khẩu "sai" trong BIP-39. Mỗi cụm mật khẩu dẫn đến một số ví, trừ khi được sử dụng trước đó sẽ trống.

❖ Cụm mật khẩu tùy chọn tạo ra hai tính năng quan trọng:

- Yếu tố thứ hai (một cái gì đó được ghi nhớ) làm cho việc ghi nhớ trở nên vô dụng, bảo vệ các bản sao lưu ghi nhớ khỏi sự xâm phạm của kẻ trộm.

- Hình thức từ chối hợp lý hoặc "ví cưỡng bức", trong đó một cụm mật khẩu được chọn dẫn đến một ví có một số tiền nhỏ được sử dụng để đánh lạc hướng kẻ tấn công khỏi ví "thực" chứa phần lớn tiền.

Tuy nhiên, điều quan trọng cần lưu ý là việc sử dụng cụm mật khẩu cũng gây ra rủi ro mất mát:

- Nếu chủ sở hữu ví mất khả năng hoặc đã chết và không ai khác biết cụm mật khẩu, mã ghi nhớ là vô dụng và tất cả số tiền được lưu trữ trong ví sẽ bị mất vĩnh viễn.

- Ngược lại, nếu chủ sở hữu sao lưu cụm mật khẩu ở cùng một nơi với mã ghi nhớ, nó sẽ đánh bại mục đích của yếu tố thứ hai.

Mặc dù cụm mật khẩu rất hữu ích, nhưng chúng chỉ nên được sử dụng kết hợp với quy trình được lên kế hoạch cẩn thận để sao lưu và phục hồi, xem xét

khả năng sống sót của chủ sở hữu và cho phép gia đình họ khôi phục tài sản tiền điện tử.

❖ Làm việc với mã ghi nhớ: BIP-39 được triển khai như một thư viện bằng nhiều ngôn ngữ lập trình khác nhau:

- *Python- mnemonic*: Việc triển khai tham chiếu tiêu chuẩn của nhóm SatoshiLabs đã đề xuất BIP-39, bằng Python.

- *BitcoinJS / BIP39*: Triển khai BIP-39, như một phần của khung bitcoinJS phổ biến, trong JavaScript.

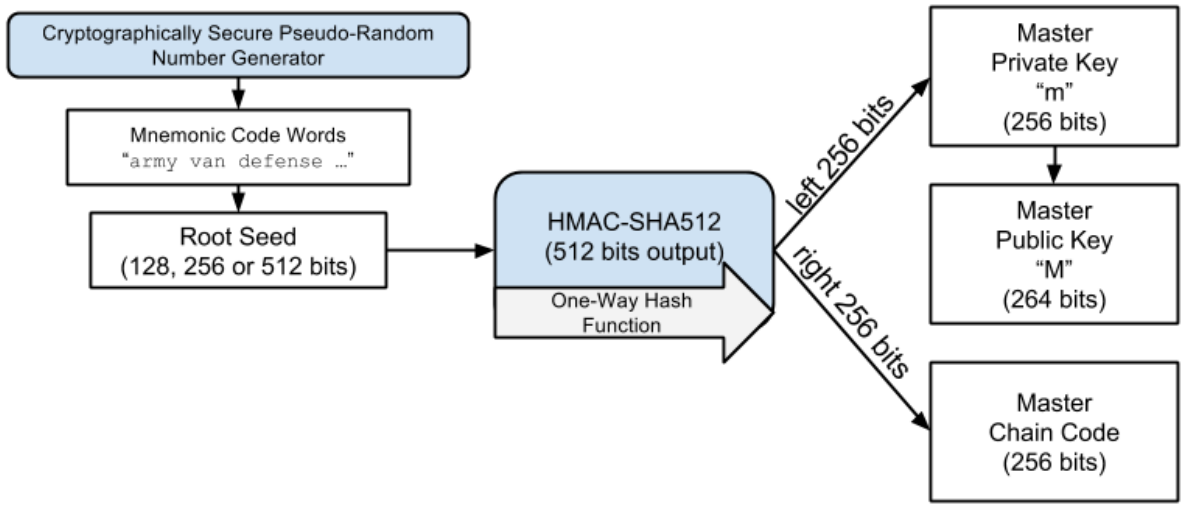
- *libbitcoin/ghi nhớ*: Triển khai BIP-39, như một phần của khung Libbitcoin phổ biến, trong C ++.

2.2 Tạo Ví HD từ Seed

Ví HD được tạo từ một seed gốc duy nhất, là một số ngẫu nhiên 128, 256 hoặc 512 bit. Thông thường nhất, seed này được tạo ra từ một ghi nhớ như chi tiết trong phần trước.

Mỗi khóa trong ví HD đều có nguồn gốc xác định từ seed gốc này, giúp có thể tạo lại toàn bộ ví HD từ seed đó trong bất kỳ ví HD tương thích nào. Điều này giúp dễ dàng sao lưu, khôi phục, xuất và nhập ví HD chứa hàng nghìn hoặc thậm chí hàng triệu khóa bằng cách chỉ cần chuyển ghi nhớ mà seed gốc có nguồn gốc.

Quá trình tạo khóa chính và mã chuỗi chính cho ví HD được thể hiện trong Tạo khóa chính và mã chuỗi từ seed gốc.



Hình 8. Tạo khóa chính và mã chuỗi từ seed gốc

Seed gốc được nhập vào thuật toán HMAC-SHA512 và hàm băm kết quả được sử dụng để tạo private key chính (m) và mã chuỗi chính (c).

Private key chính (m) sau đó tạo khóa công khai chính (M) tương ứng bằng cách sử dụng quá trình nhân đường cong elip bình thường m * G mà chúng ta đã thấy trong [pubkey].

Mã chuỗi (c) được sử dụng để giới thiệu entropy trong hàm tạo khóa con từ khóa cha, như chúng ta sẽ thấy trong phần tiếp theo.

❖ Dẫn xuất khóa con riêng tư

Ví HD sử dụng chức năng phái sinh khóa con (CKD) để lấy khóa con từ khóa cha. Các hàm phái sinh khóa con dựa trên hàm băm một chiều kết hợp:

- Private key hoặc khóa công khai cha (khóa nén ECDSA)
- Một seed được gọi là mã chuỗi (256 bit)
- Số chỉ mục (32 bit)

Mã chuỗi được sử dụng để giới thiệu dữ liệu ngẫu nhiên xác định vào quy trình, do đó việc biết chỉ mục và khóa con là không đủ để lấy các khóa con khác. Biết khóa con không giúp có thể tìm thấy anh chị em của nó, trừ khi cũng có mã chuỗi. Seed mã chuỗi ban đầu (ở gốc cây) được tạo ra từ seed, trong khi các mã chuỗi con tiếp theo có nguồn gốc từ mỗi mã chuỗi cha.

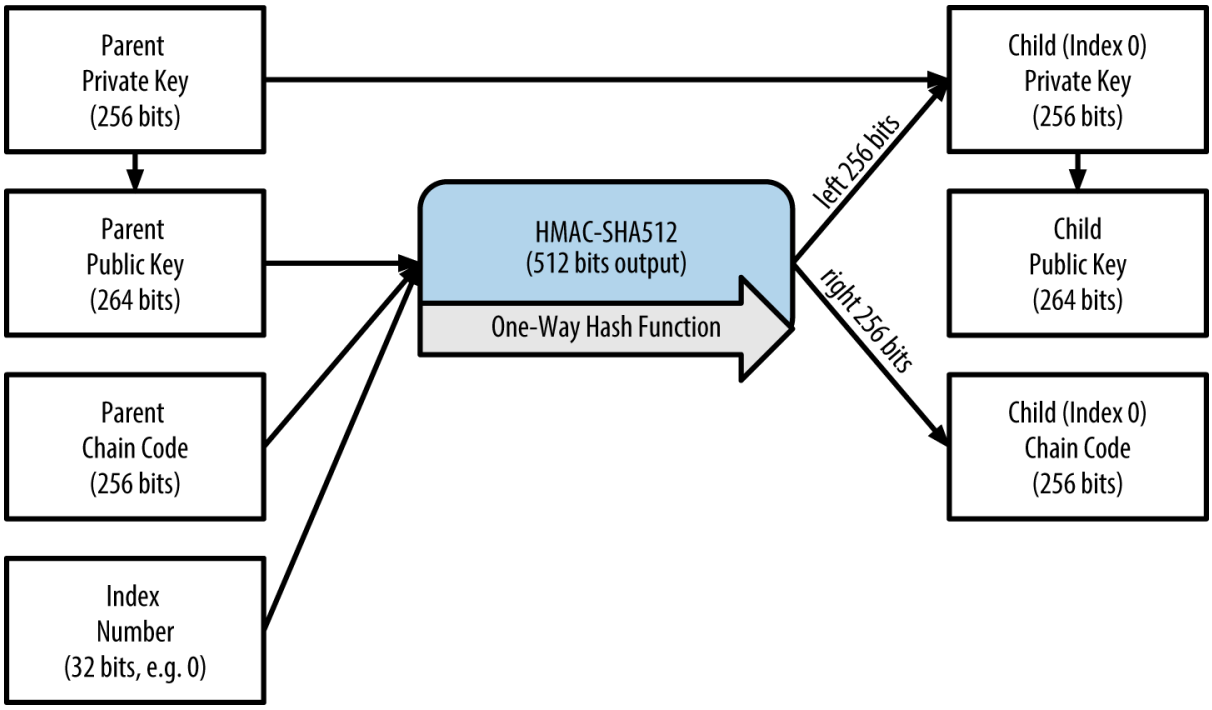
Ba mục này (khóa cha, mã chuỗi và chỉ mục) được kết hợp và băm để tạo khóa con, như sau.

Khóa công khai mẹ, mã chuỗi và số chỉ mục được kết hợp và băm với thuật toán HMAC-SHA512 để tạo ra hàm băm 512 bit. Hàm băm 512 bit này được chia thành hai nửa 256 bit. Nửa bên phải 256 bit của đầu ra băm trở thành mã chuỗi cho khóa con. Nửa bên trái 256 bit của hàm băm được thêm vào khóa cha để tạo private key con. Trong mở rộng private key cha để tạo private key con, chúng ta thấy điều này được minh họa bằng chỉ mục được đặt thành 0 để tạo ra con "không" (đầu tiên theo chỉ mục) của cha mẹ.

Thay đổi chỉ mục cho phép mở rộng chỉ mục khóa cha mẹ và tạo các khóa con khác theo chuỗi, ví dụ: Trẻ 0, Trẻ 1, Trẻ 2, v.v. Mỗi khóa cha có thể có

2.147.483.647 (231) con (231 là một nửa của toàn bộ phạm vi 232 có sẵn vì nửa còn lại được dành riêng cho một loại phái sinh đặc biệt mà chúng ta sẽ nói đến sau trong chương này).

Hình 9. Mở rộng private key của cha mẹ để tạo private key con



Lặp lại quá trình một cấp xuống cây, mỗi khóa con có thể lần lượt trở thành khóa cha mẹ và tạo ra các khóa con của riêng mình, trong vô số thế hệ.

❖ Sử dụng khóa con có nguồn gốc

Private key của con không thể phân biệt được với các khóa không xác định (ngẫu nhiên). Vì hàm phái sinh là hàm một chiều, khóa con không thể được sử dụng để tìm khóa cha. Khóa con cũng không thể được sử dụng để tìm bất kỳ anh chị em nào. Nếu có con thứ n, không thể tìm thấy anh chị em của nó, chẳng hạn như con n-1 hoặc con n + 1, hoặc bất kỳ con nào khác là một phần của chuỗi. Chỉ có khóa cha và mã chuỗi mới có thể lấy được tất cả các khóa con. Nếu không có mã chuỗi con, khóa con cũng không thể được sử dụng để lấy bất kỳ cháu nào. cần cả private key của con và mã chuỗi con để bắt đầu một nhánh mới và lấy cháu.

Vậy private key của con có thể được sử dụng để làm gì? Nó có thể được sử dụng để tạo khóa công khai và địa chỉ Bitcoin. Sau đó, nó có thể được sử dụng để ký các giao dịch để chi tiêu bất cứ thứ gì được trả cho địa chỉ đó.

Chú ý: Private key con, khóa công khai tương ứng và địa chỉ Bitcoin đều không thể phân biệt được với các khóa và địa chỉ được tạo ngẫu nhiên. Thực tế là chúng là một phần của một chuỗi không hiển thị bên ngoài chức năng ví HD đã tạo ra chúng. Sau khi được tạo, chúng hoạt động chính xác như các phím "bình thường".

❖ Phím mở rộng

Như chúng ta đã thấy trước đó, hàm phái sinh khóa có thể được sử dụng để tạo con ở bất kỳ cấp độ nào của cây, dựa trên ba đầu vào: khóa, mã chuỗi và chỉ mục của con mong muốn. Hai thành phần thiết yếu là mã khóa và mã chuỗi, và kết hợp chúng được gọi là khóa mở rộng. Thuật ngữ "khóa mở rộng" cũng có thể được coi là "khóa mở rộng" vì một khóa như vậy có thể được sử dụng để lấy con.

Các khóa mở rộng được lưu trữ và biểu diễn đơn giản dưới dạng nối của khóa 256 bit và mã chuỗi 256 bit thành chuỗi 512 bit. Có hai loại phím mở rộng. Private key mở rộng là sự kết hợp của private key và mã chuỗi và có thể được sử dụng để lấy private key con (và từ chúng, khóa công khai con). Khóa công khai mở rộng là khóa công khai và mã chuỗi, có thể được sử dụng để tạo khóa công khai con (chỉ công khai), như được mô tả trong [public_key_derivation].

Một khóa mở rộng như gốc của một nhánh trong cấu trúc cây của ví HD. Với gốc của nhánh, có thể lấy phần còn lại của nhánh. Private key mở rộng có thể tạo một nhánh hoàn chỉnh, trong khi khóa công khai mở rộng chỉ có thể tạo một nhánh khóa công khai.

Mẹo: Khóa mở rộng bao gồm private key hoặc khóa công khai và mã chuỗi. Một khóa mở rộng có thể tạo ra con, tạo ra nhánh riêng của nó trong cấu trúc cây. Chia sẻ khóa mở rộng cho phép truy cập vào toàn bộ chi nhánh.

Các khóa mở rộng được mã hóa bằng Base58Check, để dễ dàng xuất và nhập giữa các ví tương thích BIP-32 khác nhau. Mã hóa Base58Check cho các khóa mở rộng sử dụng số phiên bản đặc biệt dẫn đến tiền tố "xprv" và "xpub" khi được mã hóa bằng các ký tự Base58 để làm cho chúng dễ nhận biết. Bởi vì

khóa mở rộng là 512 hoặc 513 bit, nó cũng dài hơn nhiều so với các chuỗi được mã hóa Base58Check khác mà chúng ta đã thấy trước đây.

Dưới đây là ví dụ về private key mở rộng , được mã hóa trong Base58Check:

xprv9tyUQV64JT5qs3RSTJkXCWKMyUgoQp7F3hA1xzG6ZGu6u6Q9V
MNjGr67Lctvy5P8oyaYAL9CAWrUE9i6GoNMKUga5biW6Hx4tws2six3b9c

Đây là khóa công khai mở rộng tương ứng , được mã hóa trong Base58Check:

xpub67xpozcx8pe95XVuZLHXZeG6XWXHpGq6Qv5cmNfi7cS5mtjJ2tgy
peQbBs2UAR6KECeeMVKZBPLrtJunSDMstweyLXhRgPxdp14sk9tJPW9

❖ Dẫn xuất khóa con công khai

Như đã đề cập trước đây, một đặc điểm rất hữu ích của ví HD là khả năng lấy khóa con công khai từ khóa cha mẹ công khai mà không cần có private key. Điều này cho chúng ta hai cách để lấy khóa công khai con: từ private key của con hoặc trực tiếp từ khóa công khai cha.

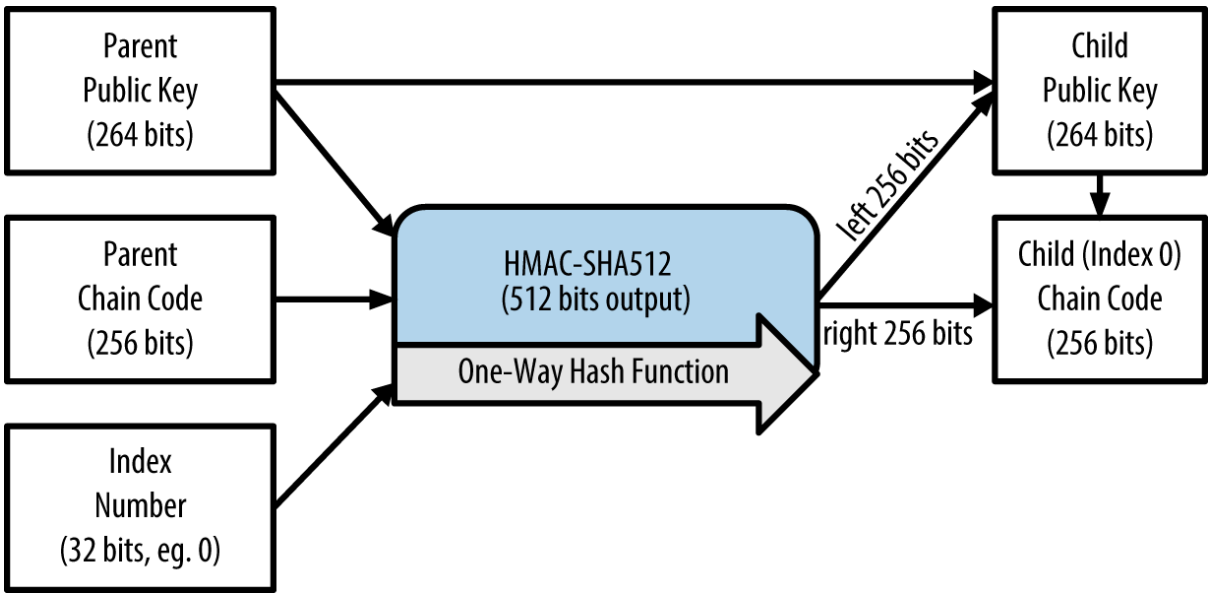
Do đó, khóa công khai mở rộng có thể được sử dụng để lấy tất cả các khóa công khai (và chỉ các khóa công khai) trong nhánh đó của cấu trúc ví HD.

Phím tắt này có thể được sử dụng để tạo các triển khai khóa công khai rất an toàn trong đó máy chủ hoặc ứng dụng có bản sao của khóa công khai mở rộng và không có private key tư nào. Kiểu triển khai đó có thể tạo ra vô số khóa công khai và địa chỉ Bitcoin, nhưng không thể chi tiêu bất kỳ khoản tiền nào được gửi đến các địa chỉ đó. Trong khi đó, trên một máy chủ khác, an toàn hơn, private key mở rộng có thể lấy được tất cả các private key tương ứng để ký giao dịch và tiêu tiền.

Một ứng dụng phổ biến của giải pháp này là cài đặt khóa công khai mở rộng trên máy chủ web phục vụ ứng dụng thương mại điện tử. Máy chủ web có thể sử dụng chức năng phái sinh khóa công khai để tạo địa chỉ Bitcoin mới cho mọi giao dịch (ví dụ: cho giỏ hàng của khách hàng). Máy chủ web sẽ không có bất kỳ private key nào dễ bị đánh cắp. Nếu không có ví HD, cách duy nhất để làm điều này là tạo hàng nghìn địa chỉ Bitcoin trên một máy chủ an toàn riêng biệt

và sau đó tải trước chúng trên máy chủ thương mại điện tử. Cách tiếp cận đó rất cồng kềnh và yêu cầu bảo trì liên tục để đảm bảo rằng máy chủ thương mại điện tử không "hết" địa chỉ.

Một ứng dụng phổ biến khác của giải pháp này là cho ví lạnh hoặc ví cứng. Trong trường hợp đó, private key mở rộng có thể được lưu trữ trên ví giấy hoặc thiết bị phần cứng (chẳng hạn như ví phần cứng Trezor), trong khi khóa công khai mở rộng có thể được giữ trực tuyến. Người dùng có thể tạo địa chỉ "nhận" theo ý muốn, trong khi các private key được lưu trữ ngoại tuyến một cách an toàn. Để chi tiêu tiền, người dùng có thể sử dụng private key mở rộng trên ứng dụng khách Bitcoin ký ngoại tuyến hoặc ký các giao dịch trên thiết bị ví phần cứng (ví dụ: Trezor). Mở rộng khóa công khai cha để tạo khóa công khai con minh họa cơ chế mở rộng khóa công khai cha để lấy khóa công khai con.



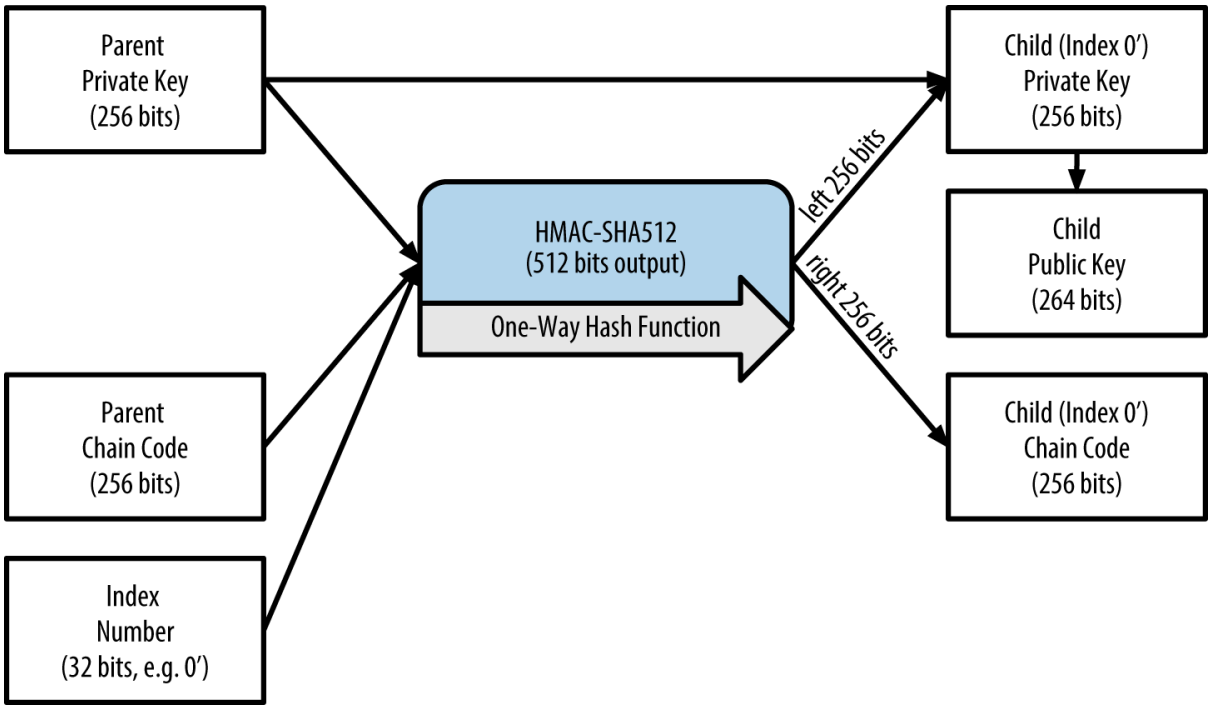
Hình 10. Mở rộng khóa công khai cha để tạo khóa công khai con

❖ Dẫn xuất khóa con cứng

Khả năng lấy một nhánh khóa công khai từ xpub rất hữu ích, nhưng nó đi kèm với rủi ro tiềm ẩn. Quyền truy cập vào xpub không cấp quyền truy cập vào private key tư của con. Tuy nhiên, vì xpub chứa mã chuỗi, nếu một private key của con được biết hoặc bằng cách nào đó bị rò rỉ, nó có thể được sử dụng với mã chuỗi để lấy tất cả các private key con khác. Một private key tư con bị rò rỉ, cùng với mã chuỗi cha, tiết lộ tất cả các private key của tất cả con. Tệ hơn nữa,

private key con cùng với mã chuỗi cha có thể được sử dụng để suy ra private key của cha mẹ.

Để chống lại rủi ro này, ví HD sử dụng một chức năng phái sinh thay thế được gọi là phái sinh cứng, "phá vỡ" mối quan hệ giữa khóa công khai mẹ và mã chuỗi con. Hàm phái sinh cứng sử dụng private key cha để lấy mã chuỗi con, thay vì khóa công khai cha. Điều này tạo ra một "tường lửa" trong chuỗi cha / con, với mã chuỗi không thể được sử dụng để thỏa hiệp private key của cha mẹ hoặc anh chị em. Hàm dẫn xuất cứng trông gần giống với dẫn xuất private key con thông thường, ngoại trừ private key cha được sử dụng làm đầu vào cho hàm băm, thay vì khóa công khai cha, như thể hiện trong sơ đồ trong dẫn xuất cứng của khóa con; bỏ qua khóa công khai cha.



Hình 11. Dẫn xuất cứng của một khóa con; bỏ qua khóa công khai cha

Khi hàm phái sinh riêng cứng được sử dụng, private key con và mã chuỗi kết quả hoàn toàn khác với kết quả của hàm phái sinh thông thường. Kết quả là "nhánh" của các khóa có thể được sử dụng để tạo ra các khóa công khai mở rộng không dễ bị tấn công, bởi vì mã chuỗi mà chúng chứa không thể bị khai thác để tiết lộ bất kỳ private key nào. Do đó, đạo hàm cứng được sử dụng để tạo ra một "khoảng trống" trong cây trên mức sử dụng khóa công khai mở rộng.

Nói một cách đơn giản, nếu muốn sử dụng sự tiện lợi của xpub để lấy các nhánh khóa công khai, mà không phải đối mặt với nguy cơ mã chuỗi bị rò rỉ, nên lấy nó từ khóa cha cứng, thay vì khóa cha bình thường (không cứng). Như một phương pháp hay nhất, con cấp 1 của các khóa chính luôn có nguồn gốc thông qua đạo hàm cứng, để ngăn chặn sự xâm phạm của các khóa chính.

❖ Số chỉ mục cho đạo hàm bình thường và cứng

Số chỉ mục được sử dụng trong hàm đạo hàm là số nguyên 32 bit. Để dễ dàng phân biệt giữa các khóa có nguồn gốc thông qua hàm đạo hàm thông thường so với các khóa có nguồn gốc thông qua đạo hàm cứng, số chỉ mục này được chia thành hai phạm vi. Các số chỉ số từ 0 đến 231–1 (0x0 đến 0x7FFFFFFF) chỉ được sử dụng cho đạo hàm thông thường. Số chỉ số từ 231 đến 232–1 (0x80000000 đến 0xFFFFFFFF) chỉ được sử dụng cho đạo hàm cứng. Do đó, nếu số chỉ số nhỏ hơn 231, con là bình thường, trong khi nếu số chỉ số bằng hoặc trên 231, con bị cứng.

Để làm cho số chỉ mục dễ đọc và hiển thị hơn, số chỉ mục cho con cứng được hiển thị bắt đầu từ số không, nhưng với ký hiệu số nguyên tố. Do đó, khóa con bình thường đầu tiên được hiển thị là 0, trong khi khóa con cứng đầu tiên (chỉ số 0x80000000) được hiển thị là 0'. Theo trình tự sau đó, khóa cứng thứ hai sẽ có 0x80000001 chỉ mục và sẽ được hiển thị dưới dạng 1', v.v. Khi thấy chỉ số ví HD i', điều đó có nghĩa là 231+i.

❖ Mã định danh khóa ví HD (đường dẫn)

Các khóa trong ví HD được xác định bằng cách sử dụng quy ước đặt tên "đường dẫn", với mỗi cấp độ của cây được phân tách bằng ký tự dấu gạch chéo (/) (xem ví dụ về đường dẫn ví HD). Private key có nguồn gốc từ private key chính bắt đầu bằng "m". Khóa công khai bắt nguồn từ khóa công khai chính bắt đầu bằng "M". Do đó, private key con đầu tiên của private key chính là m / 0. Khóa công khai con đầu tiên là M / 0. Cháu thứ hai của đứa con đầu tiên là m / 0/1, v.v.

"Tổ tiên" của một khóa được đọc từ phải sang trái, cho đến khi đạt đến khóa chính mà nó được bắt nguồn. Ví dụ: mã định danh m / x / y / z mô tả

private key là con thứ z của private key mẹ m / x / y, là con thứ y của private key mẹ m / x, là con thứ x của private key chính mẹ m.

Bảng 6. Ví dụ về đường dẫn ví HD

HD path	Key described
m/0	The first (0) child private key from the master private key (m)
m/0/0	The first (0) child private key from the first child (m/0)
m/0'/0	The first (0) normal child from the first *hardened* child (m/0')
m/1/0	The first (0) child private key from the second child (m/1)
M/23/17/0/0	The first (0) child public key from the first child (M/23/17/0) from the 18th child (M/23/17) from the 24th child (M/23)

❖ Điều hướng cấu trúc cây ví HD

Cấu trúc cây ví HD mang lại sự linh hoạt rất lớn. Mỗi phụ huynh có thể có 4 tỷ con: 2 tỷ trẻ bình thường và 2 tỷ con khó khăn. Mỗi con trong số đó có thể có thêm 4 tỷ con, v.v. Cây có thể sâu như muốn, với vô số thế hệ. Tuy nhiên, với tất cả sự linh hoạt đó, việc điều hướng cái cây vô hạn này trở nên khá khó khăn. Đặc biệt khó chuyển ví HD giữa các lần triển khai, bởi vì khả năng tổ chức nội bộ thành các chi nhánh và chi nhánh phụ là vô tận.

Hai BIP cung cấp một giải pháp cho sự phức tạp này bằng cách tạo ra một số tiêu chuẩn được đề xuất cho cấu trúc của cây ví HD. BIP-43 đề xuất sử dụng chỉ số con cứng đầu tiên như một định danh đặc biệt biểu thị "mục đích" của cấu trúc cây. Dựa trên BIP-43, ví HD chỉ nên sử dụng một nhánh cấp 1 của cây, với số chỉ mục xác định cấu trúc và không gian tên của phần còn lại của cây bằng cách xác định mục đích của nó. Ví dụ: ví HD chỉ sử dụng nhánh m / i &#x27 / nhằm biểu thị một mục đích cụ thể và mục đích đó được xác định bằng số chỉ mục "i".

Mở rộng đặc điểm kỹ thuật đó, BIP-44 đề xuất cấu trúc đa tài khoản là "mục đích" số 44 'theo BIP-43. Tất cả các ví HD theo cấu trúc BIP-44 được xác định bởi thực tế là chúng chỉ sử dụng một nhánh của cây: m / 44 '/.

BIP-44 chỉ định cấu trúc bao gồm năm cấp độ cây được xác định trước:

m / mục đích' / coin_type' / tài khoản' / thay đổi / address_index

"Mục đích" cấp độ đầu tiên luôn được đặt thành 44'. "coin_type" cấp hai chỉ định loại tiền điện tử, cho phép ví HD đa tiền tệ trong đó mỗi loại tiền tệ có cây con riêng ở cấp độ thứ hai. Hiện tại có ba loại tiền tệ được xác định: Bitcoin là m / 44 '/ 0', Bitcoin Testnet là m / 44 &#x27 / 1 ', và Litecoin là m / 44' / 2'.

Cấp độ thứ ba của cây là "tài khoản", cho phép người dùng chia nhỏ ví của họ thành các tài khoản phụ logic riêng biệt, cho mục đích kế toán hoặc tổ chức. Ví dụ: ví HD có thể chứa hai "tài khoản" bitcoin: m / 44 &#x27 / 0 #x27; / 0 ' và m/44'/0'/1'. Mỗi tài khoản là gốc của cây con riêng của nó.

Ở cấp độ thứ tư, "thay đổi", ví HD có hai cây con, một để tạo địa chỉ nhận và một để tạo địa chỉ thay đổi. Lưu ý rằng trong khi các mức trước sử dụng đạo hàm cứng, mức này sử dụng đạo hàm bình thường. Điều này là để cho phép mức này của cây xuất các khóa công khai mở rộng để sử dụng trong môi trường không an toàn. Địa chỉ có thể sử dụng có nguồn gốc từ ví HD khi còn là con của cấp bốn, làm cho cấp độ thứ năm của cây trở thành "address_index". Ví dụ: địa chỉ nhận thứ ba cho các khoản thanh toán bitcoin trong tài khoản chính sẽ là M / 44 &#x27 / 0 ' / 0 #x27; / 0/2. Ví dụ về cấu trúc ví BIP-44 HD cho thấy một vài ví dụ khác.

Bảng 7. Ví dụ về cấu trúc ví BIP-44 HD

HD path	Key described
M/44'/0'/0'/0/2	The third receiving public key for the primary bitcoin account
M/44'/0'/3'/1/14	The fifteenth change-address public key for the fourth bitcoin account
m/44'/2'/0'/0/1	The second private key in the Litecoin main account, for signing transactions

2.3 Sử dụng khóa công khai mở rộng trên cửa hàng trực tuyến

Ví HD được sử dụng cửa hàng trực tuyến của Gabriel.

Gabriel lần đầu tiên thiết lập cửa hàng trực tuyến của mình như một sở thích, dựa trên một trang Wordpress được lưu trữ đơn giản. Cửa hàng của anh khá cơ bản chỉ với một vài trang và một mẫu đơn đặt hàng với một địa chỉ Bitcoin duy nhất.

Gabriel đã sử dụng địa chỉ Bitcoin đầu tiên được tạo bởi thiết bị Trezor của mình làm địa chỉ Bitcoin chính cho cửa hàng của mình. Bằng cách này, tất cả các khoản thanh toán đến sẽ được thanh toán đến một địa chỉ được kiểm soát bởi ví phần cứng Trezor của anh ấy.

Khách hàng sẽ gửi đơn đặt hàng bằng cách sử dụng biểu mẫu và gửi thanh toán đến địa chỉ Bitcoin đã công bố của Gabriel, kích hoạt email với chi tiết đơn đặt hàng để Gabriel xử lý. Chỉ với một vài đơn đặt hàng mỗi tuần, hệ thống này hoạt động đủ tốt.

Tuy nhiên, cửa hàng trực tuyến nhỏ đã trở nên khá thành công và thu hút nhiều đơn đặt hàng từ cộng đồng địa phương. Ngay sau đó, Gabriel đã bị choáng ngợp. Với tất cả các đơn đặt hàng thanh toán cùng một địa chỉ, việc khớp chính xác các đơn đặt hàng và giao dịch trở nên khó khăn, đặc biệt là khi nhiều đơn đặt hàng cho cùng một số tiền đến gần nhau.

Ví HD của Gabriel cung cấp một giải pháp tốt hơn nhiều thông qua khả năng lấy khóa con công khai mà không cần biết private key tư. Gabriel có thể tải khóa công khai mở rộng (xpub) trên trang web của mình, có thể được sử dụng để lấy một địa chỉ duy nhất cho mỗi đơn đặt hàng của khách hàng. Gabriel có thể chi tiêu tiền từ Trezor của mình, nhưng xpub được tải trên trang web chỉ có thể tạo địa chỉ và nhận tiền. Tính năng này của ví HD là một tính năng bảo mật tuyệt vời. Trang web của Gabriel không chứa bất kỳ private key tư nào và do đó không cần mức độ bảo mật cao.

Để xuất xpub, Gabriel sử dụng ứng dụng Trezor Suite dành cho máy tính để bàn kết hợp với ví phần cứng Trezor. Thiết bị Trezor phải được cắm vào để xuất khóa công khai. Lưu ý rằng ví phần cứng sẽ không bao giờ xuất private key tư — những khóa đó luôn tồn tại trên thiết bị. Xuất xpub từ ví cứng Trezor cho thấy những gì Gabriel thấy trong Trezor Suite khi xuất xpub.

Hình 12. Xuất xpub từ ví cứng Trezor

Gabriel sao chép xpub vào phần mềm cửa hàng bitcoin của cửa hàng trực tuyến của mình. Anh ấy sử dụng Máy chủ BTCPay, một cửa hàng trực tuyến mã nguồn mở cho nhiều nền tảng nội dung và lưu trữ web. Máy chủ BTCPay sử dụng xpub để tạo một địa chỉ duy nhất cho mỗi giao dịch mua.

Khám phá và quản lý tài khoản

Công việc kinh doanh của Gabriel đang phát triển mạnh mẽ. Anh ấy đã cung cấp khóa công khai mở rộng (xpub) của mình cho Máy chủ BTCPay, nơi đang tạo địa chỉ duy nhất cho khách hàng đến trang web của anh ấy. Mỗi khi khách hàng vào trang web của Gabriel nhấp vào nút "Thanh toán" với một phương thức thanh toán được chỉ định (trong trường hợp này là bitcoin), Máy chủ BTCPay sẽ tạo một địa chỉ mới cho khách hàng đó. Cụ thể hơn, BTCPay Server lặp lại trên cây address_index để tạo một địa chỉ mới để hiển thị cho khách hàng, như được xác định bởi BIP-44. Nếu khách hàng quyết định chuyển đổi phương thức thanh toán hoặc từ bỏ hoàn toàn giao dịch, địa chỉ Bitcoin này sẽ không được sử dụng và sẽ không được sử dụng cho khách hàng khác ngay lập tức.

Tại một thời điểm duy nhất, trang web của Gabriel có thể có một khối lượng lớn các địa chỉ nổi bật cho khách hàng mua hàng, một số trong đó có thể không được sử dụng và cuối cùng hết hạn. Khi các địa chỉ này hết hạn, Máy chủ BTCPay sẽ quay lại sử dụng lại các địa chỉ này để lấp đầy khoảng trống trong

address index, nhưng rõ ràng làm thế nào có thể có khoảng trống giữa các lá address_index của cây xác định phân cấp nơi tiền thực sự được đặt.

Giả sử Gabriel quan tâm đến việc xem tổng số bitcoin kiếm được trên ví chỉ xem (ví cho phép xem lịch sử giao dịch, nhưng không chi tiêu tiền) tách biệt với Máy chủ BTCPay nhưng cũng phù hợp với tiêu chuẩn BIP-44. Ví riêng biệt này nên tìm kiếm tiền trong cây phân cấp rộng lớn này như thế nào và khi nào nó nên ngừng tìm kiếm? Hầu hết các ví thường sẽ tuân theo một quy trình lặp đi lặp lại sử dụng giới hạn được xác định trước, được gọi là giới hạn khoảng cách. Nếu, trong khi tìm kiếm các địa chỉ đã sử dụng, ví không tìm thấy các địa chỉ được sử dụng liên tiếp vượt quá số giới hạn này, nó sẽ ngừng tìm kiếm chuỗi địa chỉ. Giới hạn khoảng cách mặc định thường được đặt thành 20. Điều này được trình bày chi tiết trong BIP-44.

Chú ý: Giới hạn khoảng cách giải thích hiện tượng theo đó việc nhập ví có thể hiển thị số dư không chính xác hoặc bằng không. Các khoản tiền không bị mất, mà thay vào đó, chức năng nhập ví đã không đi qua đủ lá để phát hiện đầy đủ tiền. Nhiều ví cho phép thay đổi giới hạn khoảng cách mặc định này và Gabriel có thể cần tăng giới hạn này để cho phép ví của mình nhập đầy đủ lịch sử giao dịch của mình.