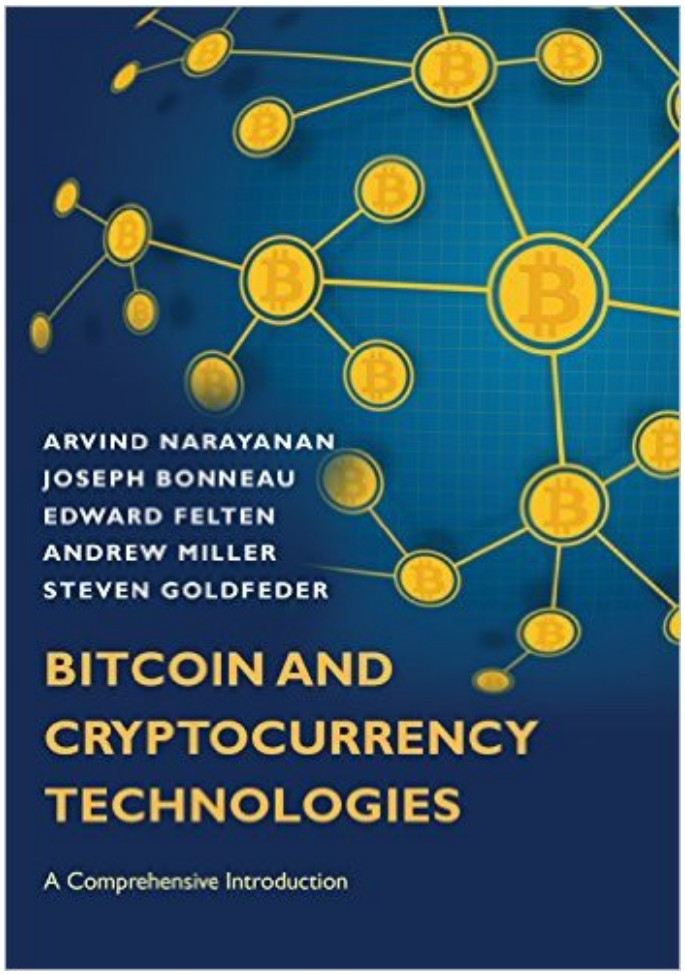
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Nhập môn An toàn thông tin

Giới thiệu sơ lược về Mật mã và Tiền mật mã

Ngày 3 tháng 6 năm 2021

Tài liệu tham khảo



Nội dung

Hàm băm mật mã

Hàm băm là một hàm thỏa mãn ba tính chất:

- Đầu vào là một xâu độ dài bất kỳ.
- Đầu ra là một xâu độ dài cố định. Ta sẽ dùng 256 bit.
- Đây là một hàm tính được một cách hiệu quả.

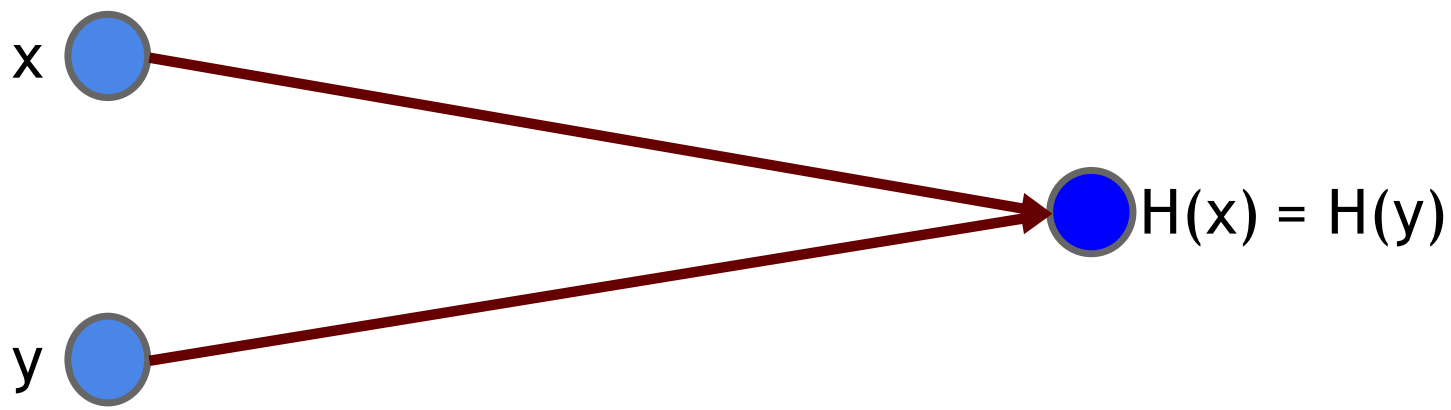
Tính an toàn

- collision-free (kháng xung đột)
- hiding
- puzzle-friendly

Tính chất 1: Kháng xung đột

Vô cùng khó để tìm hai xâu x và y sao cho

$$x \neq y \text{ và } H(x) = H(y).$$



Ứng dụng của hàm băm kháng xung đột

15.04

(Vivid Vervet): April 2015 (Supported until January 2016)

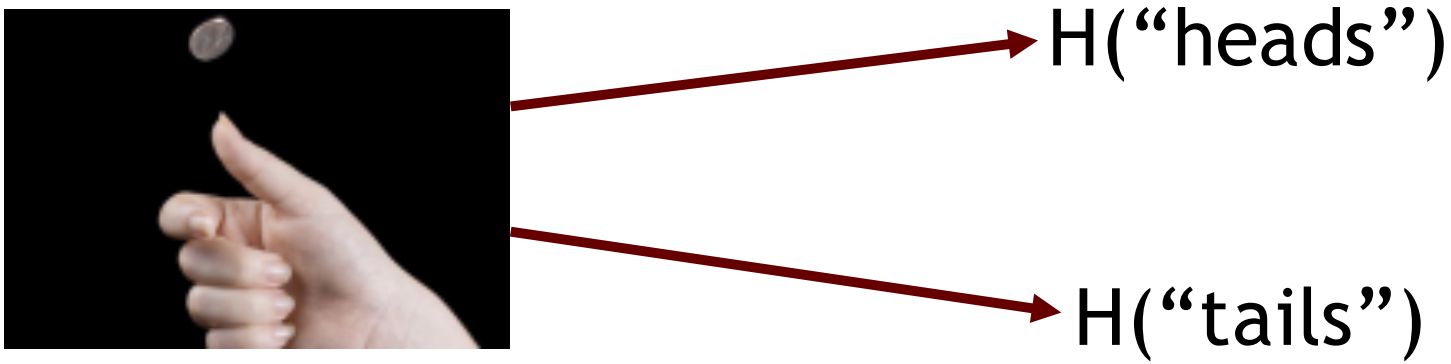
md5 Hash	*Version*
53c869eba8686007239a650d903847fd	ubuntu-15.04-desktop-amd64.iso
6ea04093b767ad6778aa245d53625612	ubuntu-15.04-desktop-i386.iso
487f4a81f22f8597503db3d51a1b502e	ubuntu-15.04-server-amd64.iso
49de7a0ed202d11456126589e2d1db22	ubuntu-15.04-server-i386.iso
fcfba8de8848944705cd200ff76c53cf	ubuntu-15.04-snappy-amd64-generic.img.xz
ef2a4951a2e889908a55c980d2bea475	ubuntu-15.04-snappy-armhf-bbb.img.xz

Tính chất 2: Hiding

- Ta mong muốn tính chất:

  Cho $H(x)$, không thể tìm được x.

- Điều này không khả thi khi không gian input nhỏ!



Tính chất 2: Hiding

Một hàm băm gọi là hiding nếu

khi giữ bí mật giá trị r được chọn ngẫu nhiên, thì cho $H(r \mid x)$ ta không thể tìm được x.

Ứng dụng: Sơ đồ ủy thác

- Mô phỏng: "Giấu một giá trị trong phong bì dán kín" và "Mở phong bì" sau này.
- Ủy thác một giá trị và tiết lộ nó sau này.

Sơ đồ ủy thác

API:

- `(com, key) := commit(msg)`
- `match := verify(com, key, msg)`

Giấu `msg` trong phong bì:

- `(com, key) := commit(msg)`
- sau đó công khai `com`

Mở phong bì:

- Công khai `key`, `msg`
- Ai cũng có thể dùng hàm `verify()` để kiểm tra tính hợp lệ.

Tính an toàn của Sơ đồ ủy thác

API:

- `(com, key) := commit(msg)`
- `match := verify(com, key, msg)`

Tính an toàn:

- Hiding: Chỉ cho `com`, ta không thể tìm được `msg`.
- Binding: Ta không thể tìm được `msg` $\neq$ `msg'` sao cho `verify(commit(msg), msg') == true`

Sơ đồ ủy thác

Cài đặt:

- `commit(msg):=(H(key | msg), H(key))`
  với `key` là một giá trị ngẫu nhiên 256 bit.
- `verify(com, key, msg):=(H(key | msg) == com)`

Tính an toàn dựa trên:

- Hiding: Chỉ cho `H(key | msg)`, không thể tìm được `msg`.
- Binding: Không thể tìm được `msg` $\neq$ `msg'` such that `H(key | msg) == H(key | msg')`

Tính chất 3: Puzzle-friendly

Puzzle-friendly:

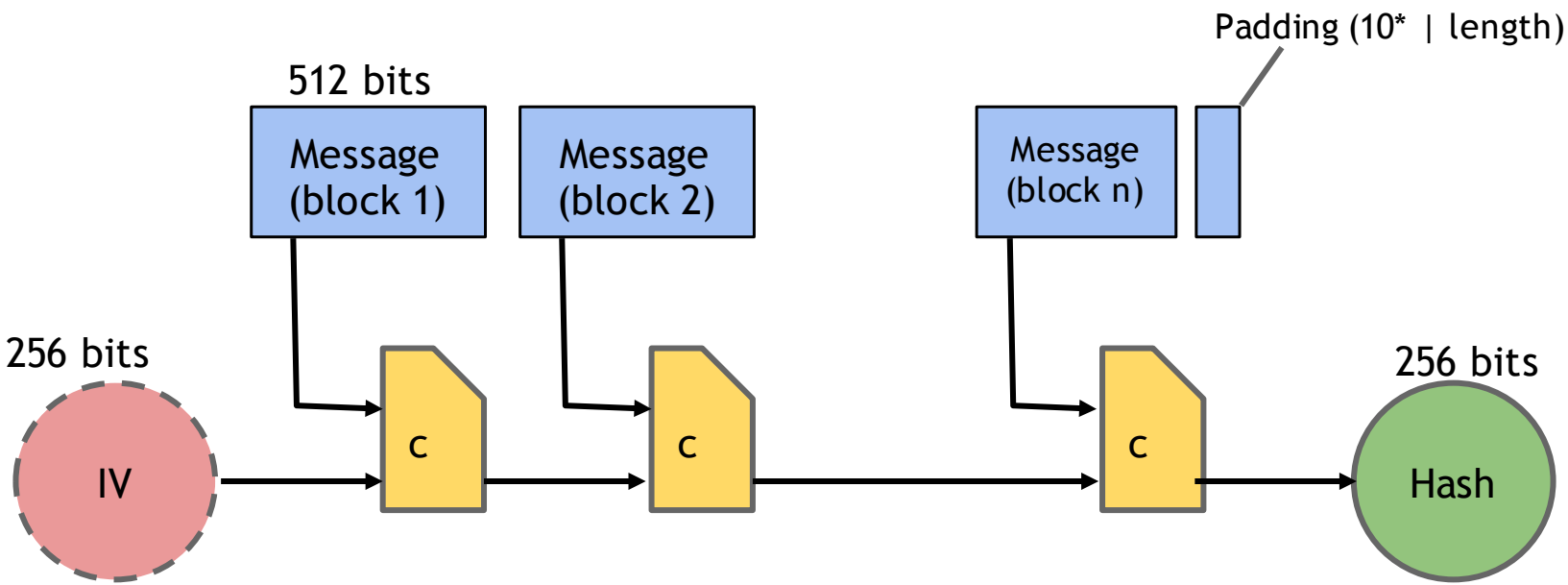
- Với mọi giá trị y cho trước và k được chọn ngẫu nhiên, thì vô cùng khó để tìm được x sao cho $H(k \mid x) = y$.

Ứng dụng: Search puzzle

- Cho một "puzzle ID" `id`, và một tập giá trị Y.
- Hãy tìm một "nghiệm" x thỏa mãn $H(\texttt{id} \mid x) \in Y$.

Tính chất Puzzle-friendly chỉ ra rằng không có chiến lược nào tốt hơn việc thử một cách ngẫu nhiên giá trị cho x.

Hàm băm SHA-256



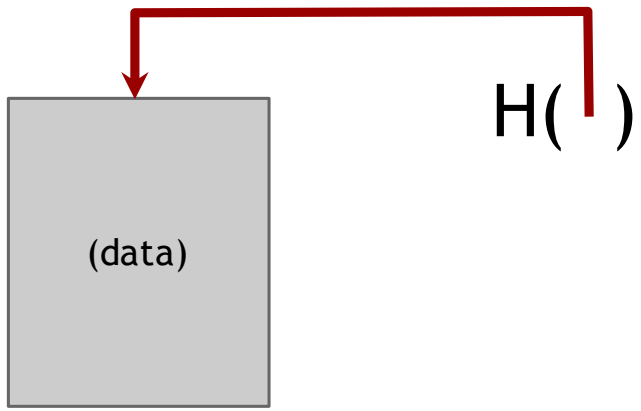
Thảo luận

Thiết kế giao thức để chơi trò chơi dân gian Oẳn tù tì qua điện thoại.

Con trỏ băm

Con trỏ băm là

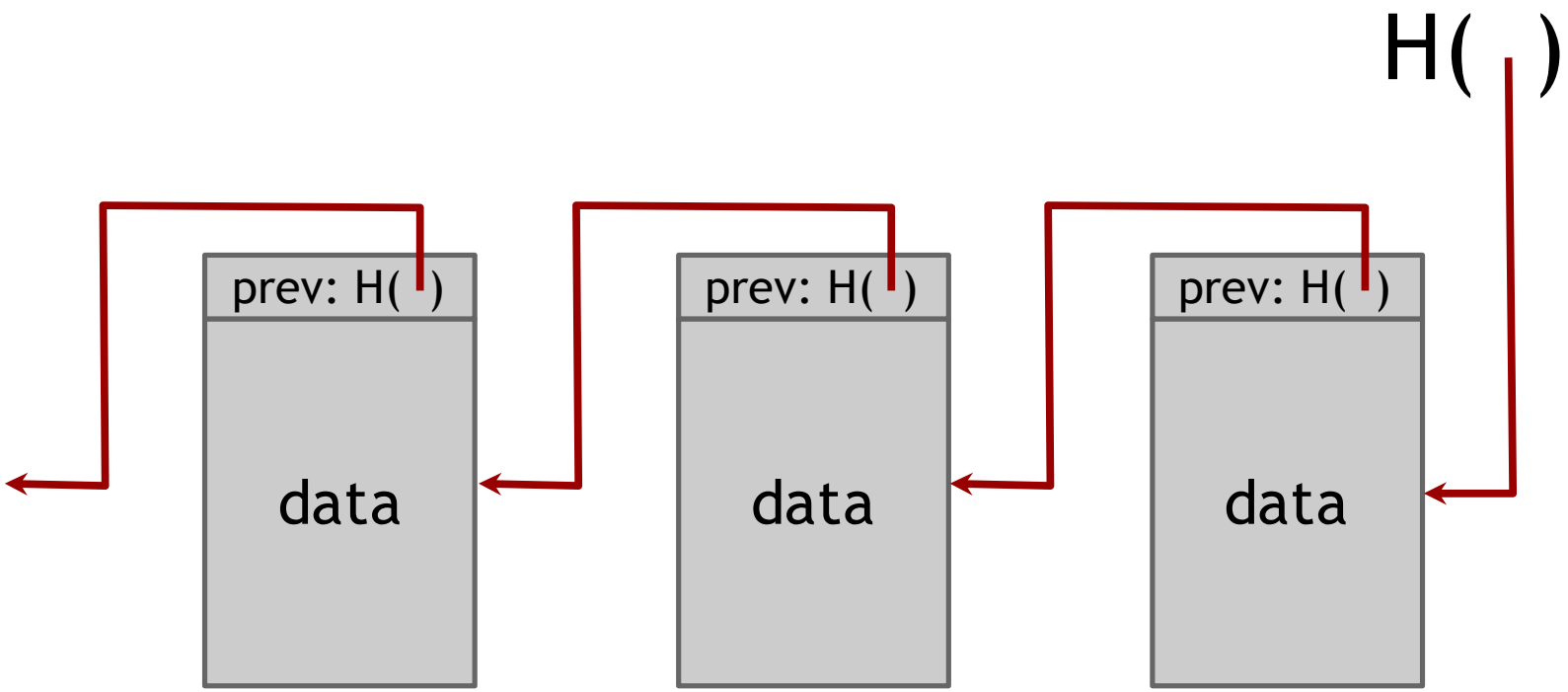
- một con trỏ tới dữ liệu, và
- mã băm của dữ liệu.



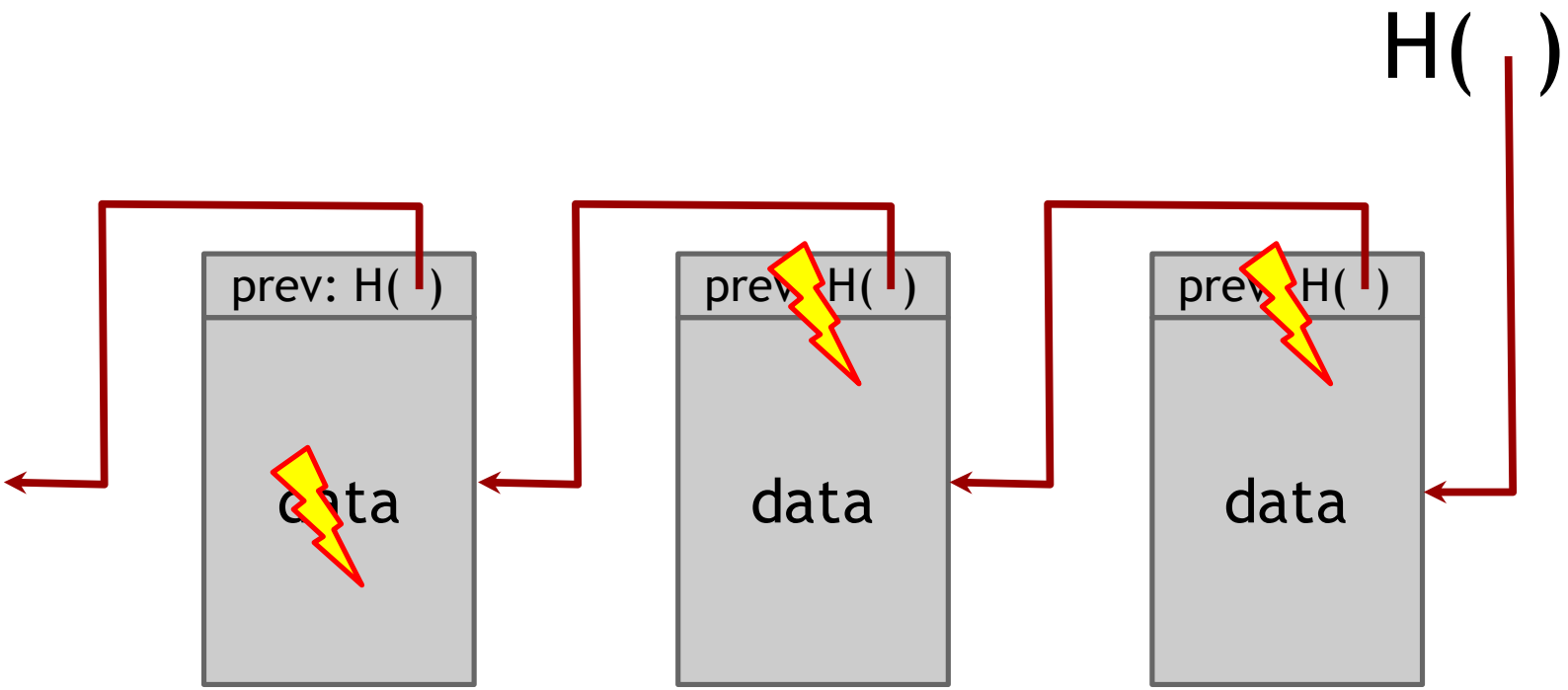
Nếu ta có một con trỏ băm, ta có thể

- lấy dữ liệu, và
- kiểm tra xem liệu dữ liệu có bị sửa đổi.

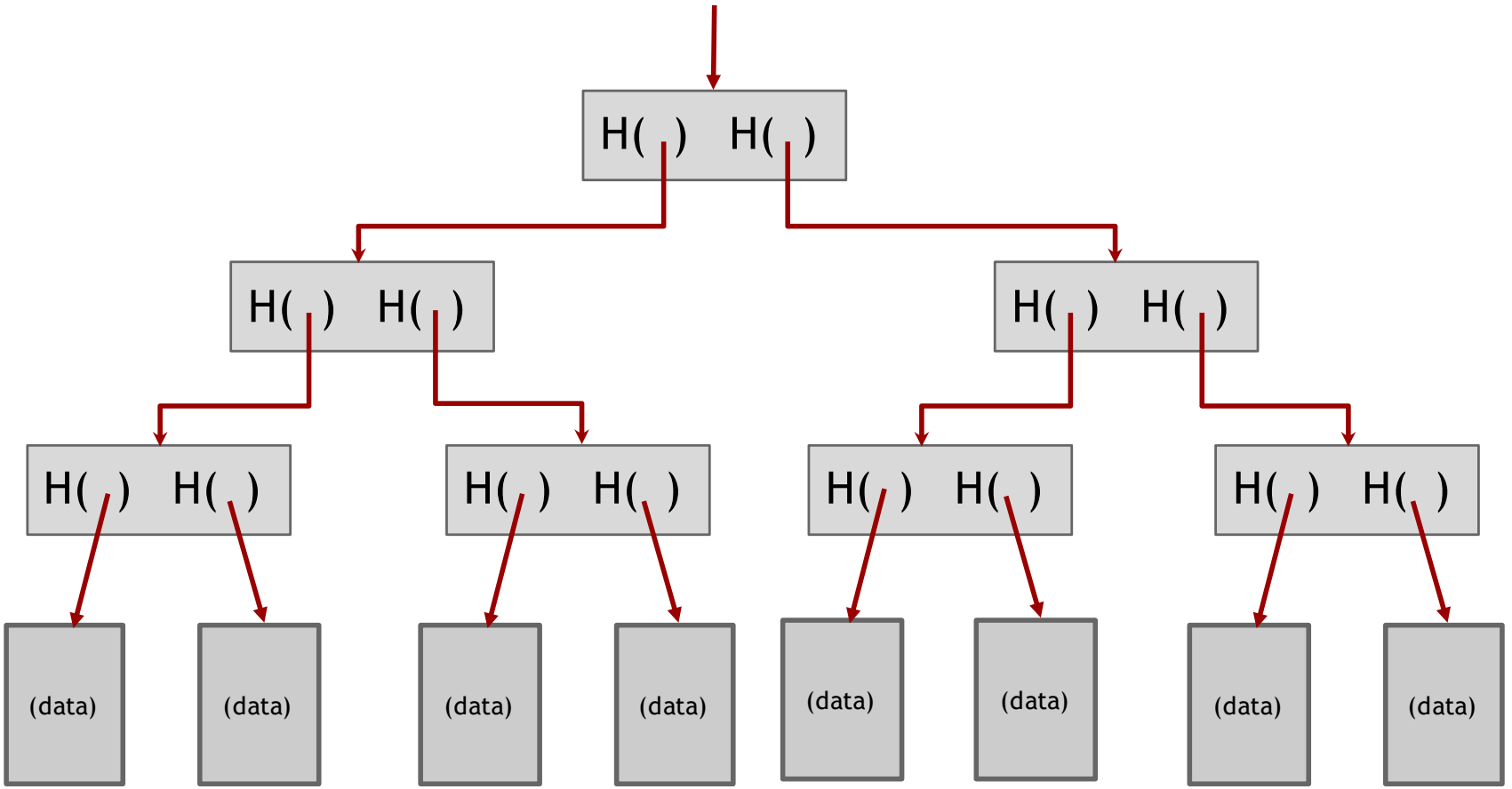
Danh sách liên kết với con trỏ băm = "blockchain"



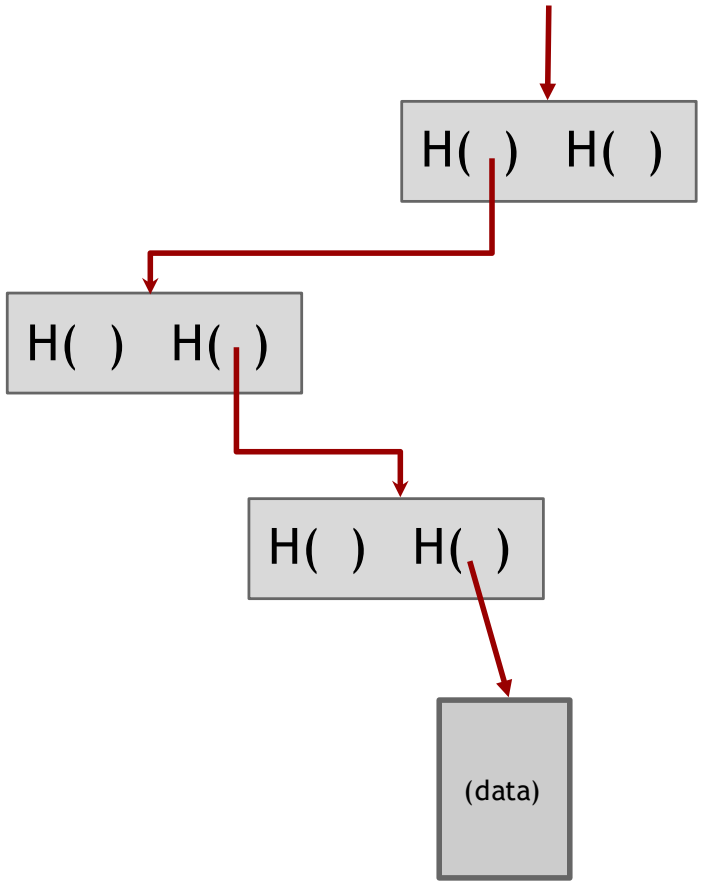
Phát hiện sửa đổi



Cây nhị phân với con trỏ băm = "Merkle tree"



Là thành viên của cây?



Trong O(log n)

Chữ ký số

Ý tưởng:

- Chỉ bạn có thể ký, nhưng ai cũng có thể kiểm tra
- Chữ ký được gắn với một tài liệu cụ thể:
  không thể copy và paste cho tài liệu khác.

API:

- `(sk, pk) := generateKeys(keysize)`
- `sig := sign(sk, message)`
- `isValid := verify(pk, message, sig)`

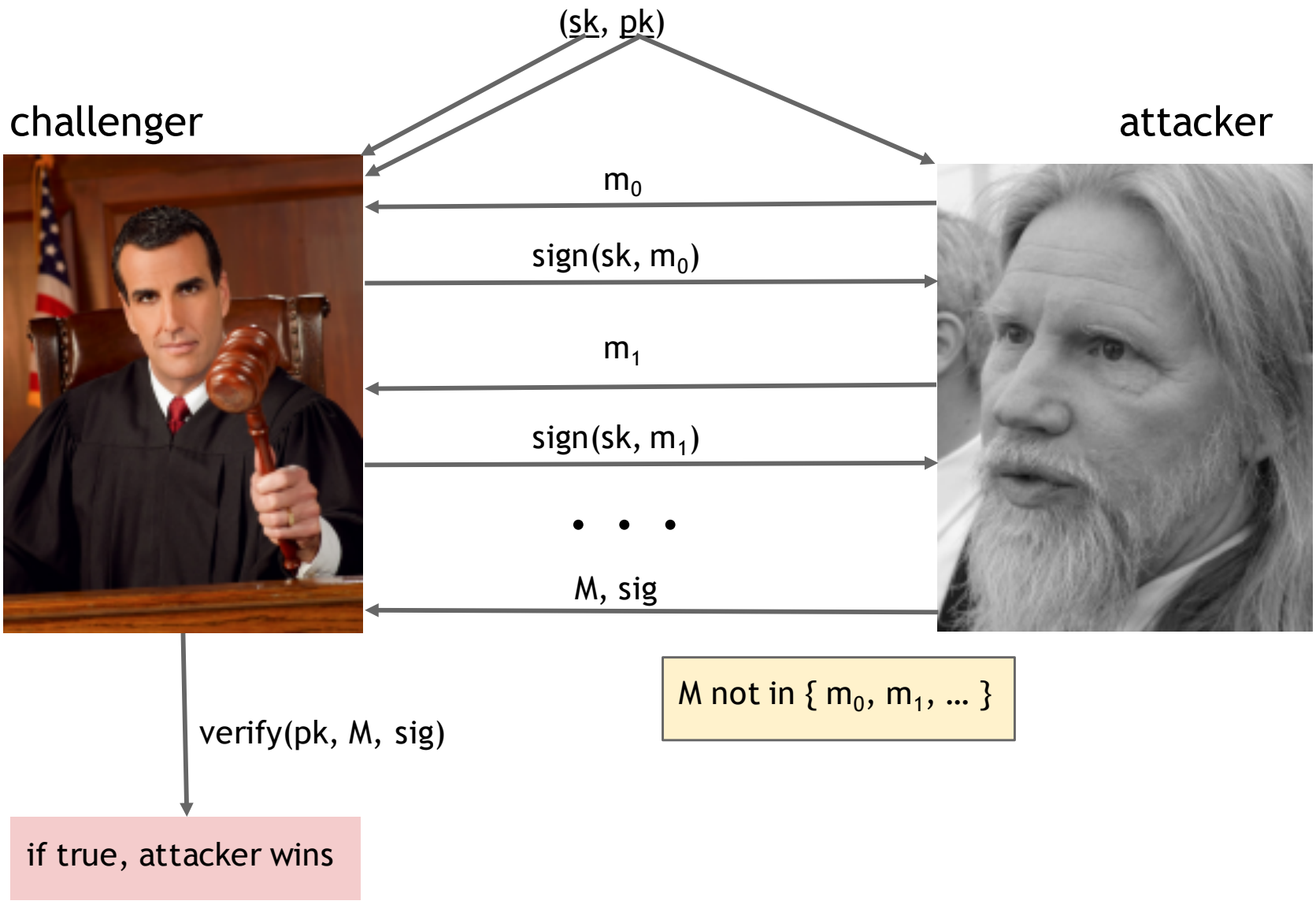
Tính an toàn của chữ ký số

Kiểm tra đúng chữ ký hợp lệ

$$\texttt{verify}(\texttt{pk}, \texttt{message}, \texttt{sign}(\texttt{sk}, \texttt{message})) == \texttt{true}$$

Không thể giả mạo chữ ký:

- Dù kẻ tấn công Evil biết `pk` và có được chữ ký hợp lệ của một số tài liệu (Evil muốn)
- Evil cũng không thể tạo ra chữ ký hợp lệ cho tài liệu mới.


(sk, pk)
challenger
attacker
m_0
sign(sk, m_0)
m_1
sign(sk, m_1)
• • •
M, sig
M not in { m_0, m_1, … }
verify(pk, M, sig)
if true, attacker wins

Thực tế

- Bitcoin sử dụng sơ đồ chữ ký số ECDSA (Elliptic Curve Digital Signature Algorithm)
- Cụ thể, Bitcoin sử dụng đường cong chuẩn `secp256k1`: có mức an toàn 128 bit
- Kích thước khóa và chữ ký:

Khóa bí mật:	256 bit
Khóa công khai, chưa nén:	512 bit
Khóa công khai, đã nén:	257 bit
Thông điệp được ký:	256 bit
Chữ ký:	512 bit

Dùng khóa công khai như định danh

- Nếu có `sig` thỏa mãn

$$\texttt{verify}(\texttt{pk}, \texttt{msg}, \texttt{sig}) == \texttt{true}$$

  thì ta có thể xem như

$$\texttt{pk}\ \text{"thông báo"}\ \texttt{msg}$$

- Để có thể thông báo trong vai trò `pk`, ta phải biết khóa bí mật `sk`.

Làm thế nào để tạo ra một định danh mới?

Sinh ra một cặp khóa ngẫu nhiên `sk,pk`:

$$(\texttt{sk}, \texttt{pk}) := \texttt{generateKeys}(\texttt{keysize})$$

- `pk` là tên "công khai" mà bạn có thể dùng. Hoặc tốt hơn nếu dùng `hash(pk)` làm tên công khai.
- `sk` cho phép tạo ra thông báo.

Bạn hoàn toàn kiểm soát định danh của bạn vì chỉ có bạn biết `sk`.

Hệ thống quản lý định danh phi tập trung

- Ai cũng có thể tạo ra định danh mới: bất cứ lúc nào và bao nhiêu cũng được!
- Không cần có hệ thống quản lý trung tâm.
- Trong Bitcoin, định danh được gọi là "địa chỉ".

Tính riêng tư

- Các địa chỉ không trực tiếp liên kết với định danh trong thế giới thực.
- Nhưng nếu quan sát theo thời gian ta có thể liên kết các hoạt động từ các địa chỉ, và từ đó suy ra mối liên hệ giữa các địa chỉ với đối tượng trong thế giới thực.

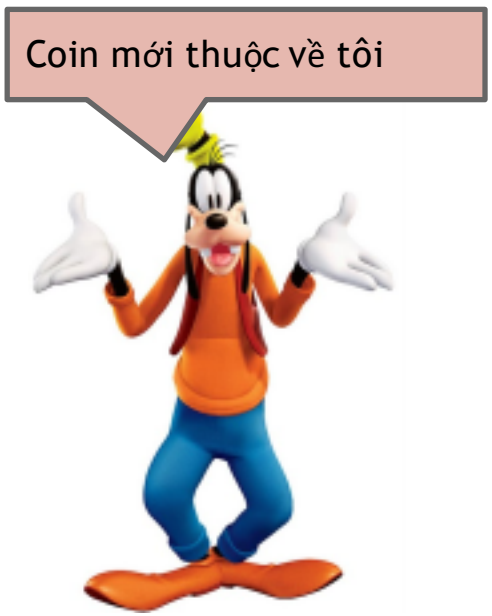
Nội dung

GoofyCoin

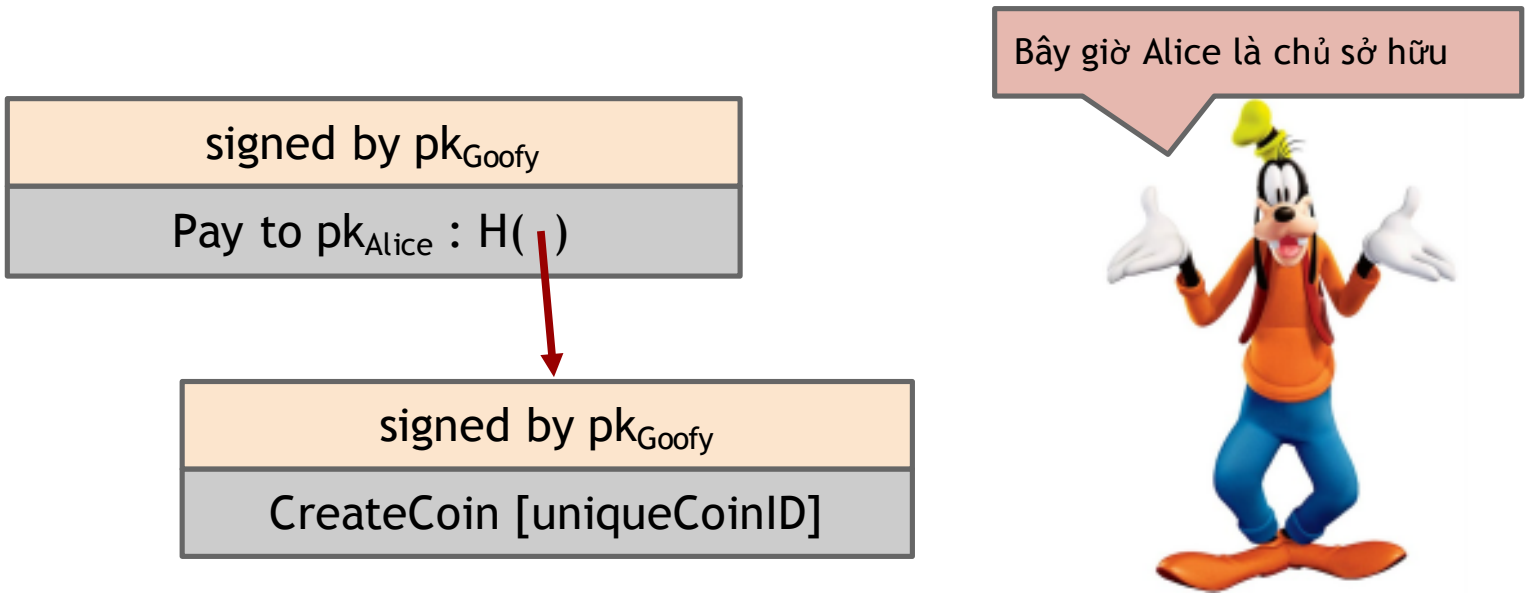
Goofy có thể tạo ra coin mới

signed by pk_{Goofy}

CreateCoin [uniqueCoinID]

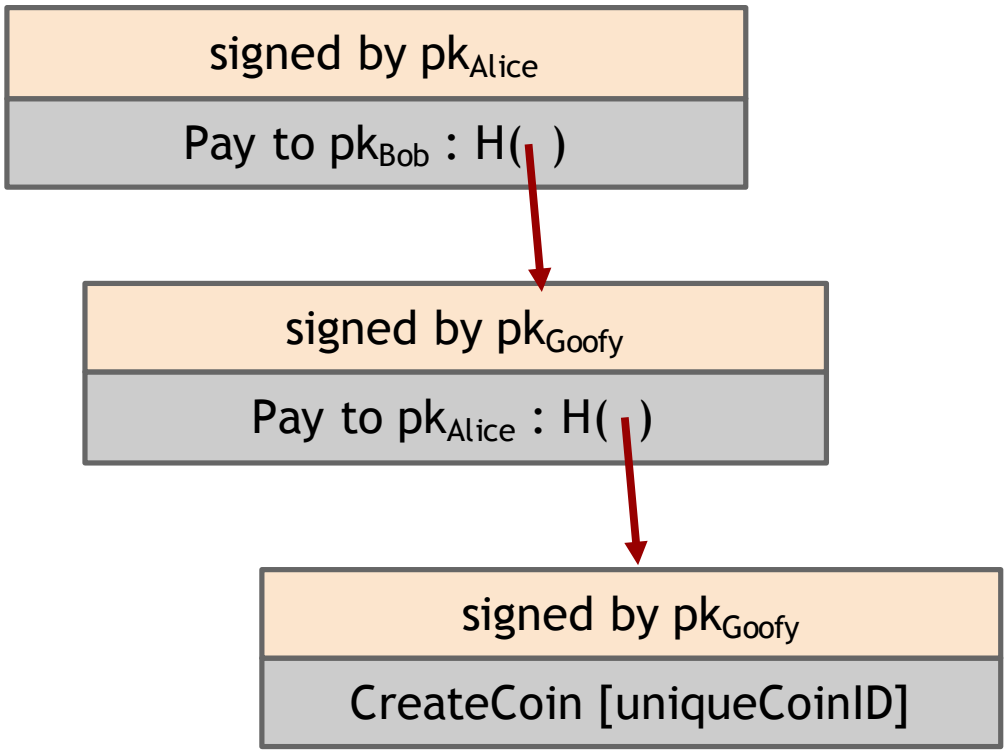


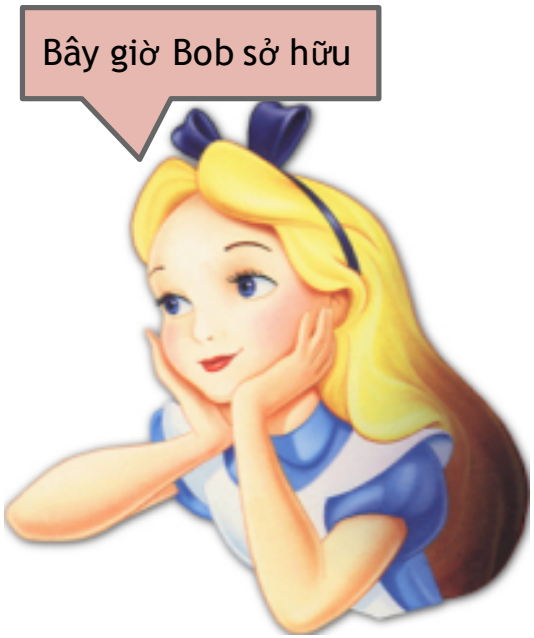
Goofy chuyển coin cho Alice



Hình: Alice có thể chứng minh mình sở hữu coin bằng cách đưa ra dữ liệu.

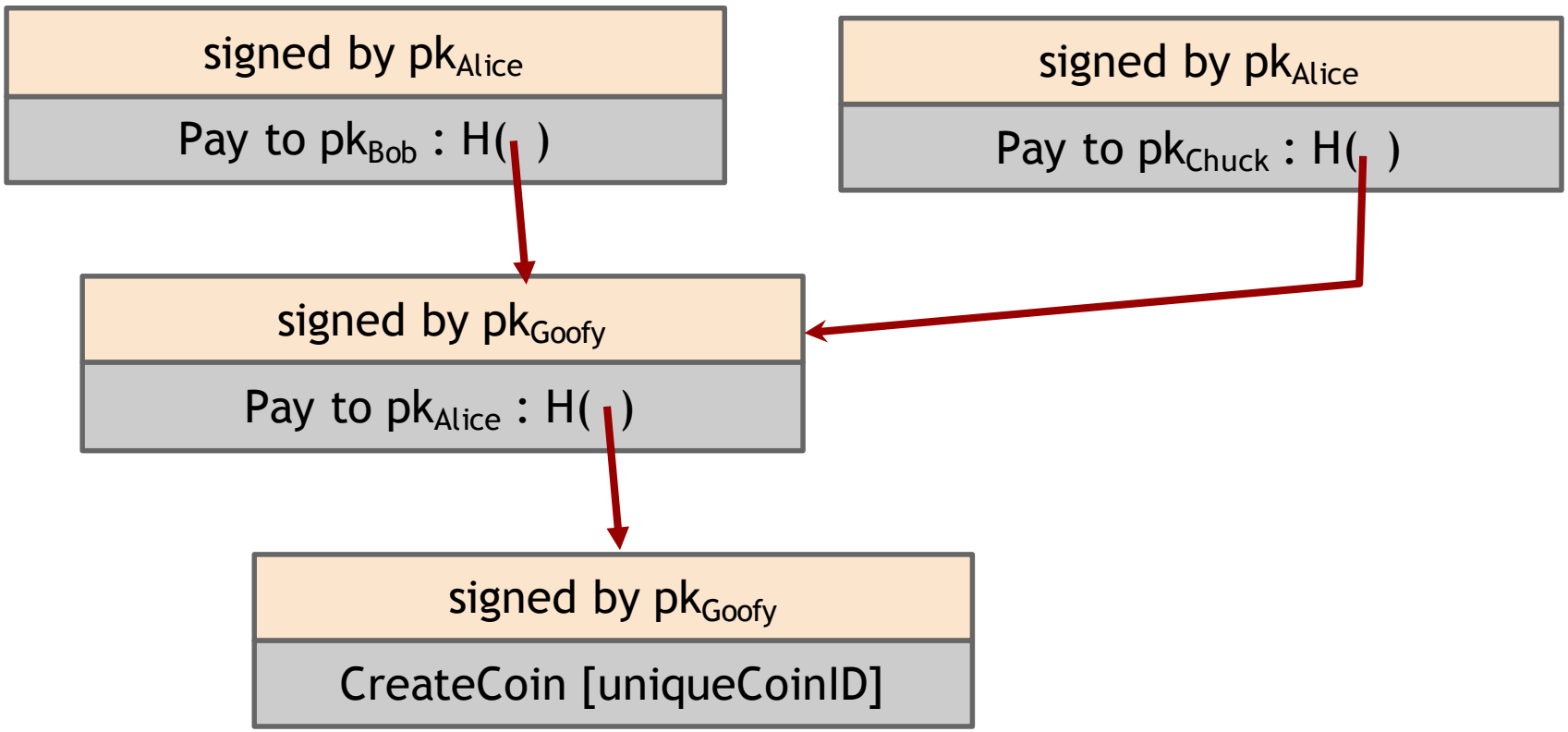
Và Alice chuyển coin cho Bob





Hình: Bob có thể chứng minh mình sở hữu coin bằng cách đưa ra dữ liệu.

Vấn đề double-spending

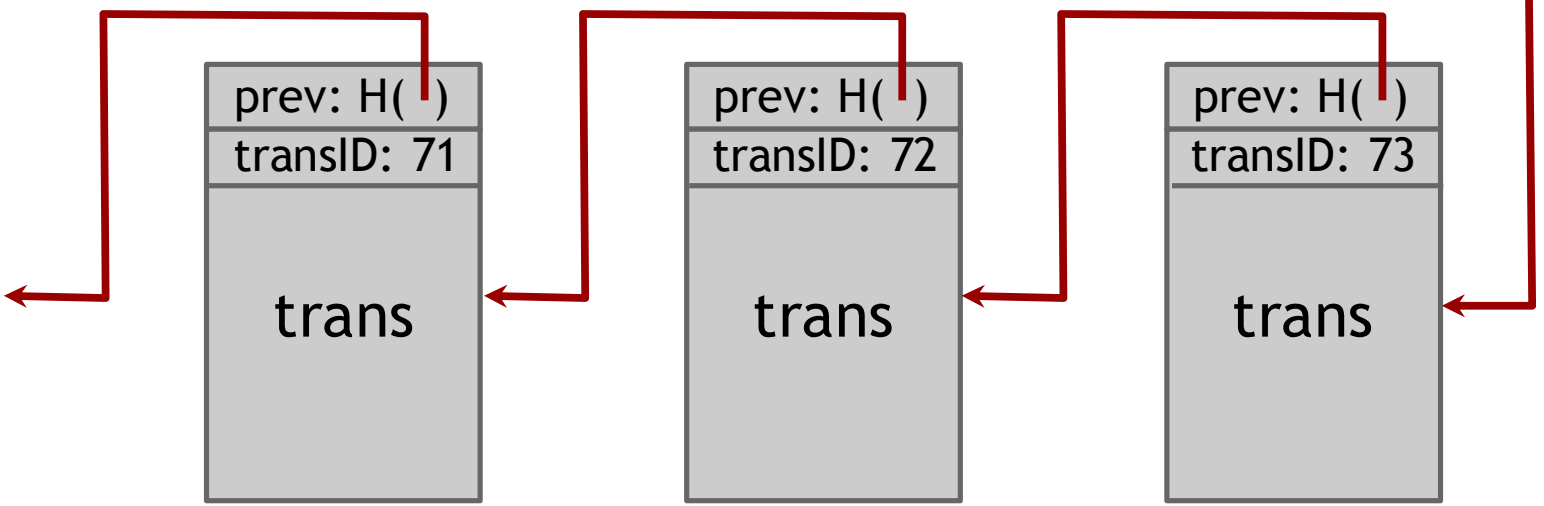


Hình: Alice tiêu một coin hai lần.

ScroogeCoin

Scrooge công khai lịch sử của mọi giao dịch
(một block chain, được ký bởi Scrooge)

H( )



Giao dịch CreateCoins tạo ra coin mới

transID: 73	type:CreateCoins	
coins created		
num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

coinID 73(0)

coinID 73(1)

coinID 73(2)

Đúng, vì tôi tạo ra.

Giao dịch PayCoin

Tiêu và phá hủy Coin

transID: 73 type:PayCoins		
consumed coinIDs: 68(1), 42(0), 72(3)		
coins created		
num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
signatures		

Hợp lệ nếu:

- coin được tiêu là hợp lệ,
- vẫn chưa được tiêu,
- tổng giá trị out = tổng giá trị in, và
- được ký bởi người sở hữu các coin được tiêu

Vấn đề của ScroogeCoin

- ScroogeCoin đảm bảo chống được double-spending vì mọi người đều có thể nhìn vào blockchain và đảm bảo các giao dịch là hợp lệ.
- Scrooge không thể tạo ra giao dịch giả vì anh ta không thể giả mạo chữ ký của người khác.
- Nhưng Scrooge có quá nhiều ảnh hưởng:
  - Có thể từ chối công khai một giao dịch nào đó,
  - tạo nhiều coin, hoặc
  - dừng cập nhật blockchain.

Thảo luận

- Hệ thống ScroogeCoin là hệ thống tập trung. Nó dựa hoàn toàn vào một bên trung gian tin cậy (Scrooge)!
- Xây dựng tiền mật mã hoạt động giống ScroogeCoin nhưng không cần bên trung gian tin cậy.