Blockchain & Bitcoin Network

CS 5158/6058 Data Security and Privacy

Spring 2018

Instructor: Boyang Wang

Trans. ID	Sender	Receiver	Amount	Time
t1	Alice	Bob	$20	03/10/2018
t2	Alice	David	$100	03/15/2018
t3	Bob	David	$50	03/21/2018

- A bank has a number of transactions
- Bank wants to ensure all the trans are correct.
  - Trans could be changed by hacker on server
  - Trans could be changed by attacker on channel

Use Hash Function

- Assume a bank has transactions: {t1, t2, t3}
  - Can use a hash function (e.g., SHA256)
  - Take all trans. as input, compute one hash value
  - E.g., H(t1, t2, t3) = h

- Bank publishes **h**, anyone can verify {t1, t2, t3}
  - Given {t1, t2, t3}, Alice checks H(t1,t2, t3) ?= h
    - If identical, all trans are correct;
    - otherwise, some trans is not correct

Trans. ID	Sender	Receiver	Amount	Time
t1	Alice	Bob	$20	03/10/2018
t2	Alice	David	$100	03/15/2018
t3	Bob	David	$50	03/21/2018
t4	David	Alice	$80	03/22/2018
t5	Charlie	David	$10	03/23/2018

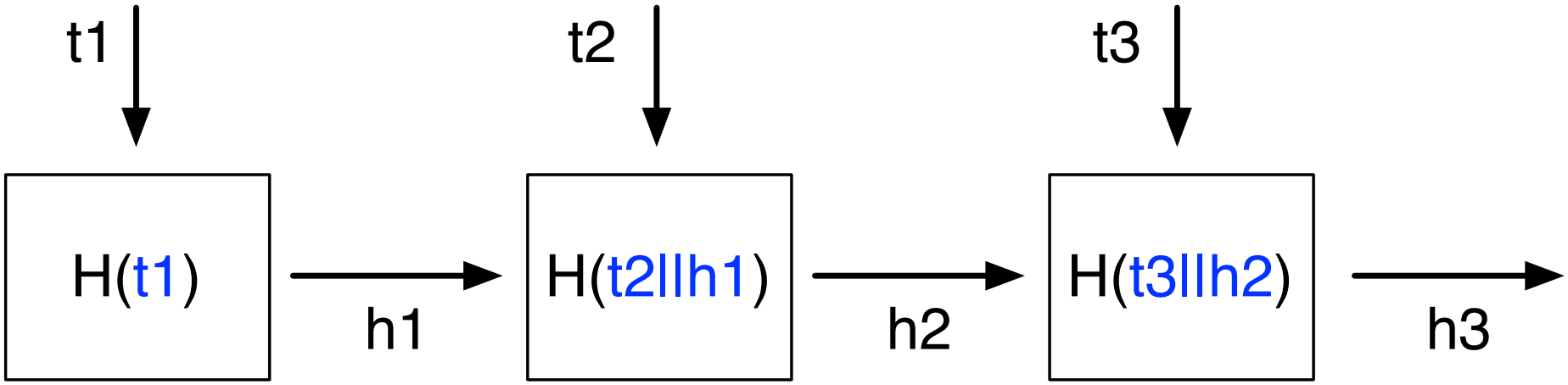
- Bank has new transactions every day
- Bank wants to ensure all the trans are correct.

Update Hash Value

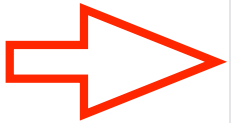
- Bank has new transactions: {t1, t2, t3, t4, t5}

- Recompute/Update a hash value
  - Take all trans. as input, compute one hash value
  - E.g., H(t1, t2, t3, t4, t5) = h
  - Bank publishes **h**, anyone can verify all trans

- Inefficient: bank needs to take all the trans as input
  - E.g., reading 1,000,000 trans. takes long time

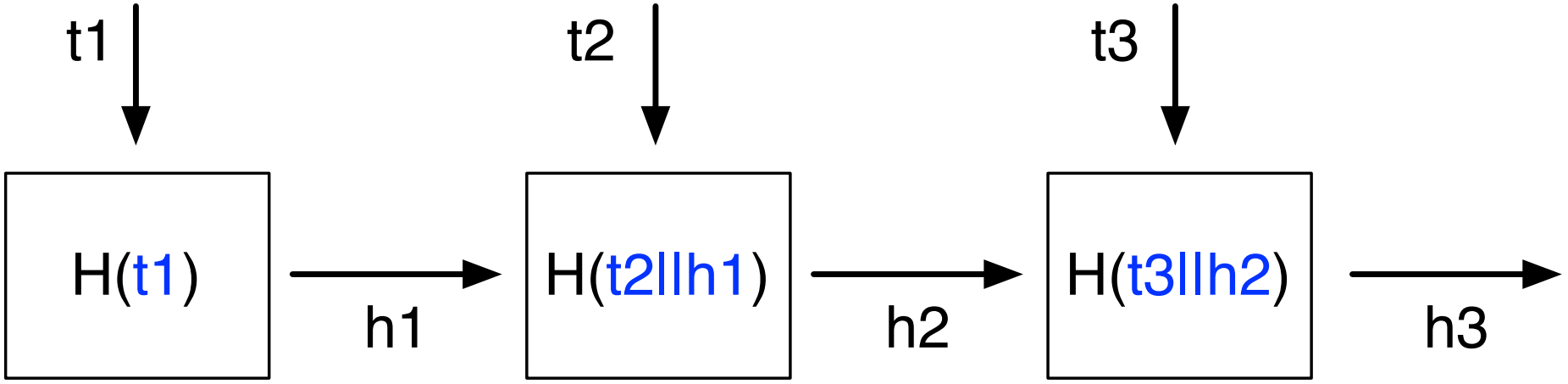


- Hash chain:
  - Given a new trans, bank computes a hash value of new trans with the current hash value
    - h1 can prove {t1} is correct
    - h2 can prove {t1, t2} are correct
    - h3 can prove {t1, t2, t3} are correct
  - Bank always updates the latest hash value

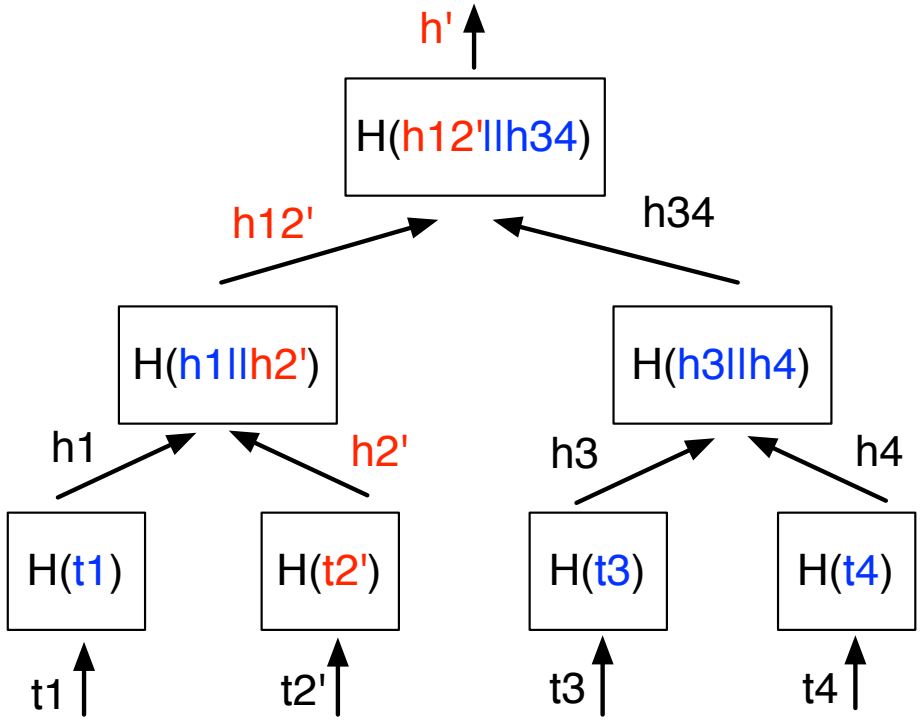


Trans. ID	Sender	Receiver	Amount	Time
t1	Alice	Bob	$20	03/10/2018
t2	Alice	David	$1000	03/15/2018
t3	Bob	David	$50	03/21/2018
t4	David	Alice	$80	03/22/2018
t5	Charlie	David	$10	03/23/2018

- Bank has new transactions every day.
- Bank updates old transactions.
- Bank wants to ensure all the trans are correct.



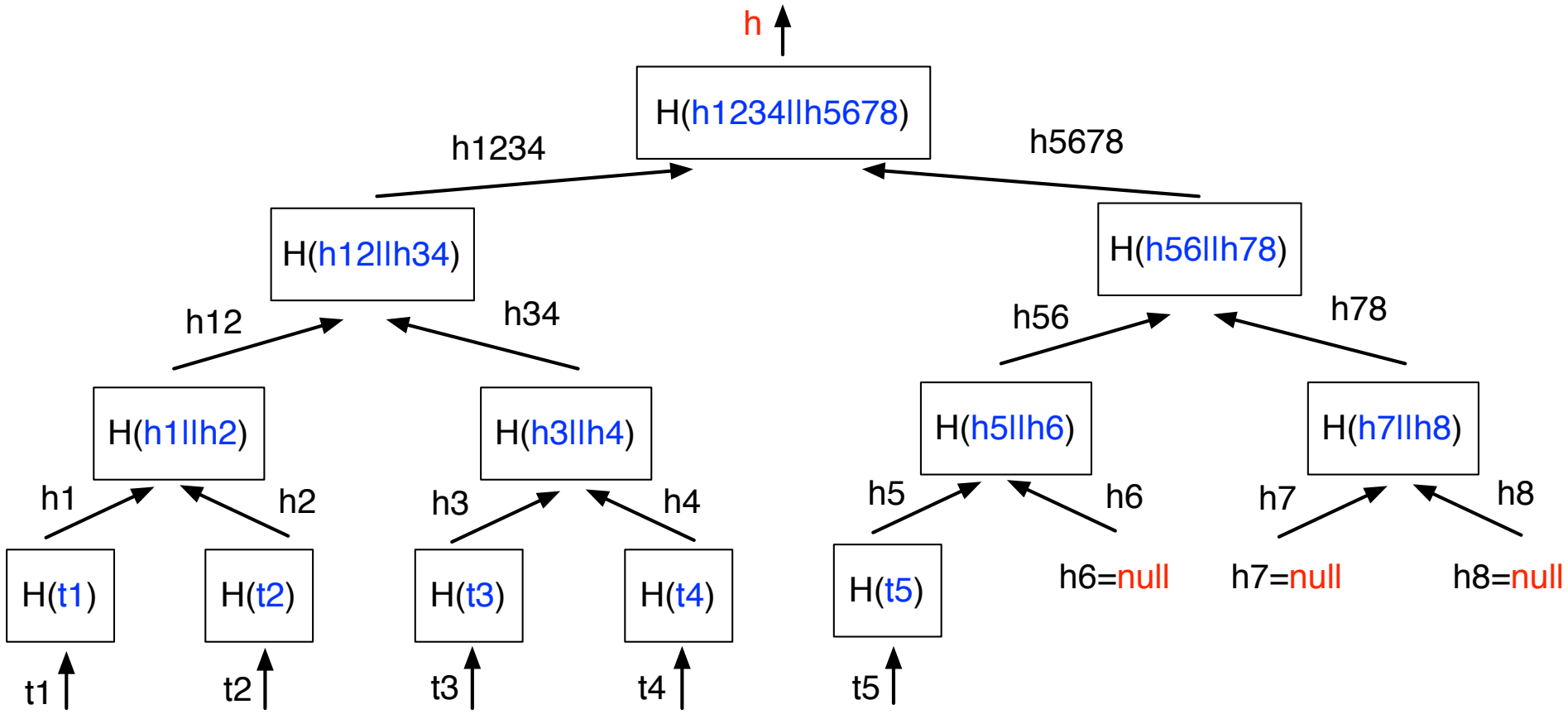
- Bank updates using Hash chain:
  - If t3 is updated to t3'
    - recompute h3, if still has a copy of h2; otherwise, recompute h1, h2, h3
  - If t1 is updated to t1',
    - recompute h1, h2, h3
- Recompute O(N) hashes, not very efficient
  - E.g., N=1,000,000



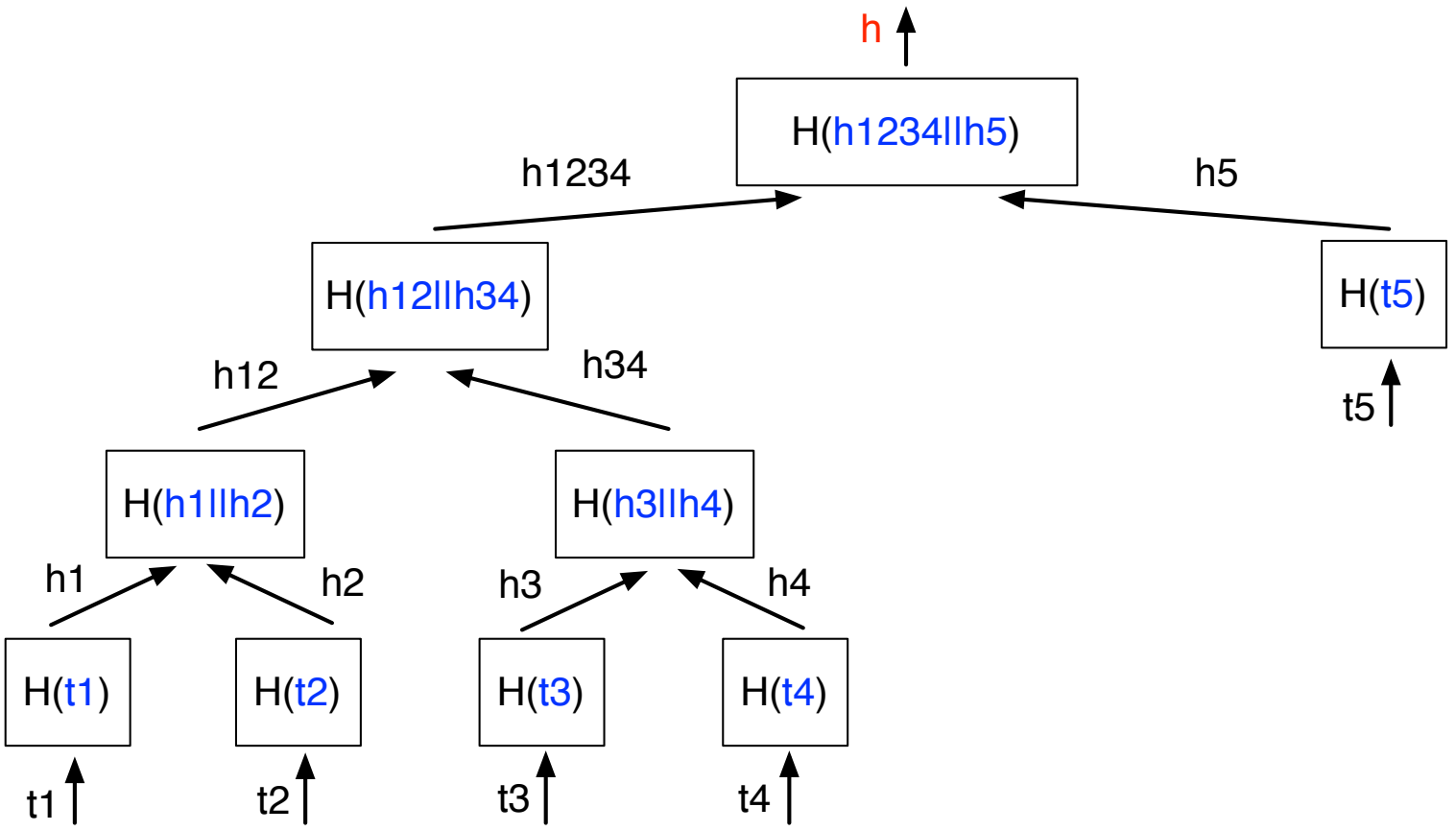
- Bank updates t2 to t2’
  - Recomputes h2’, h12’, h’ in the tree
    - Assume bank maintains all the non-leaf nodes
    - O(logN) hashes v.s. O(N) in hash chain

- Bank publishes **h’**, anyone can verify all trans

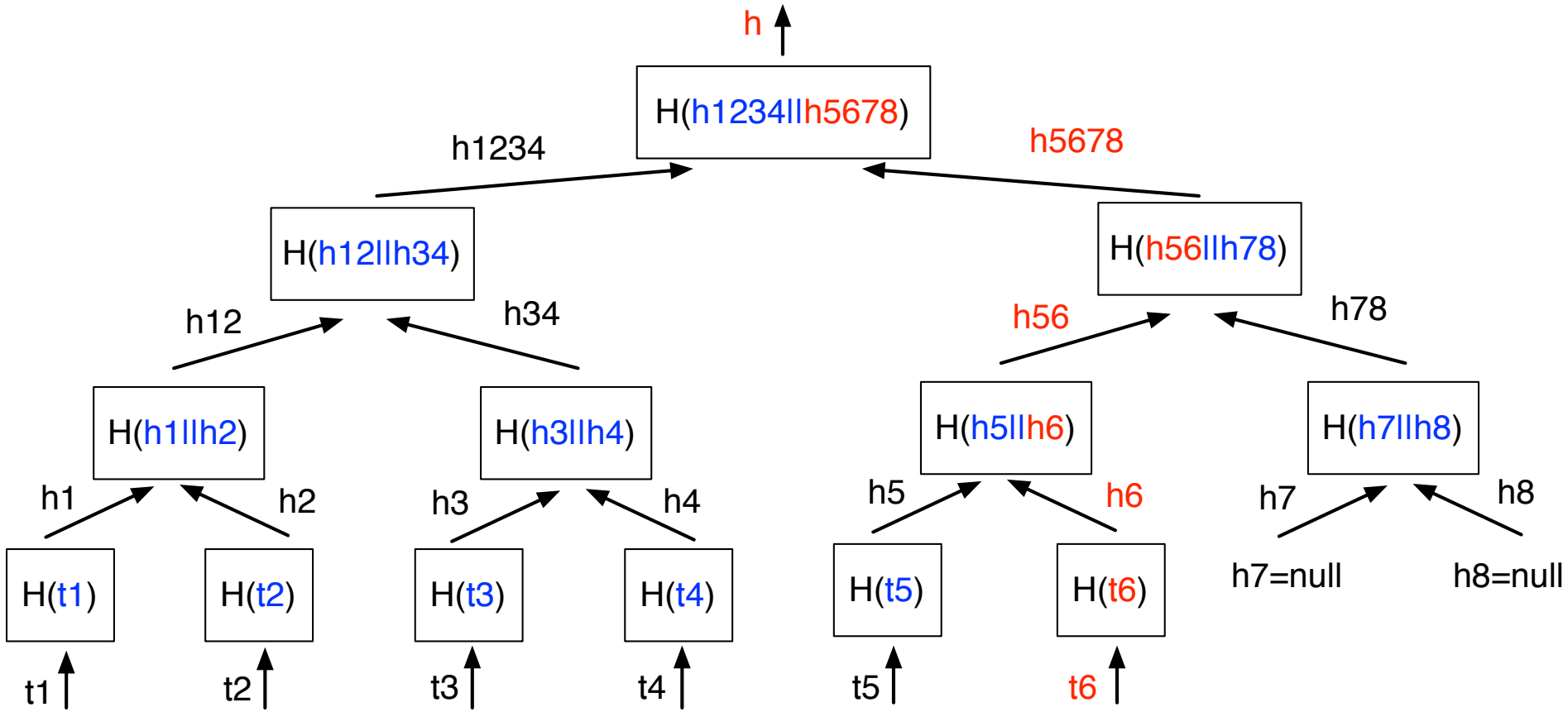
- Practice: if bank has 5 trans, {t1, t2, t3, t4, t5}
  - How to compute the root hash?
  - Leave h6, h7, h8 as null (easy to implement, all the leaf nodes are at the same level)



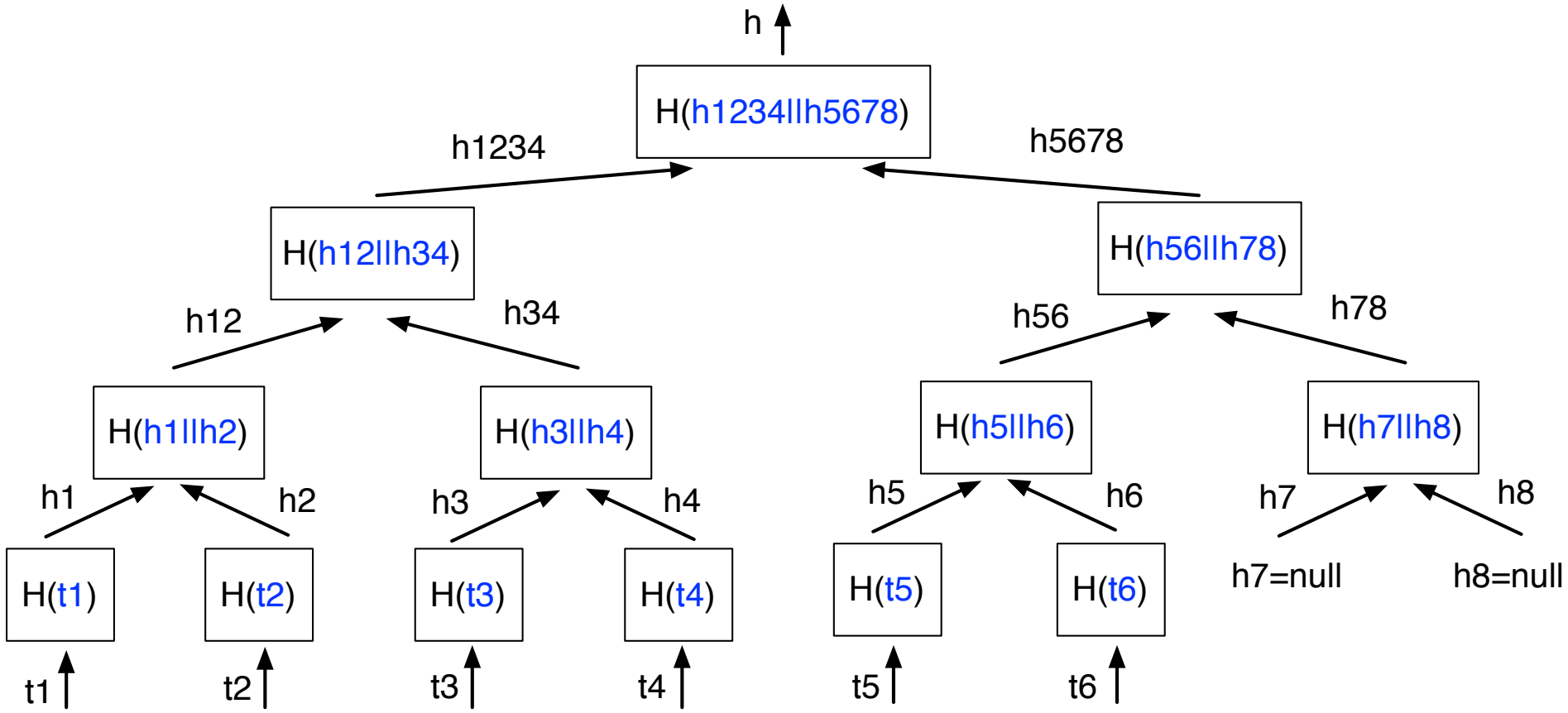
- Practice: if bank has 5 trans, {t1, t2, t3, t4, t5}
  - How to compute the root hash?
  - Another possible solution (also works, as long as verification algo. is consistent)



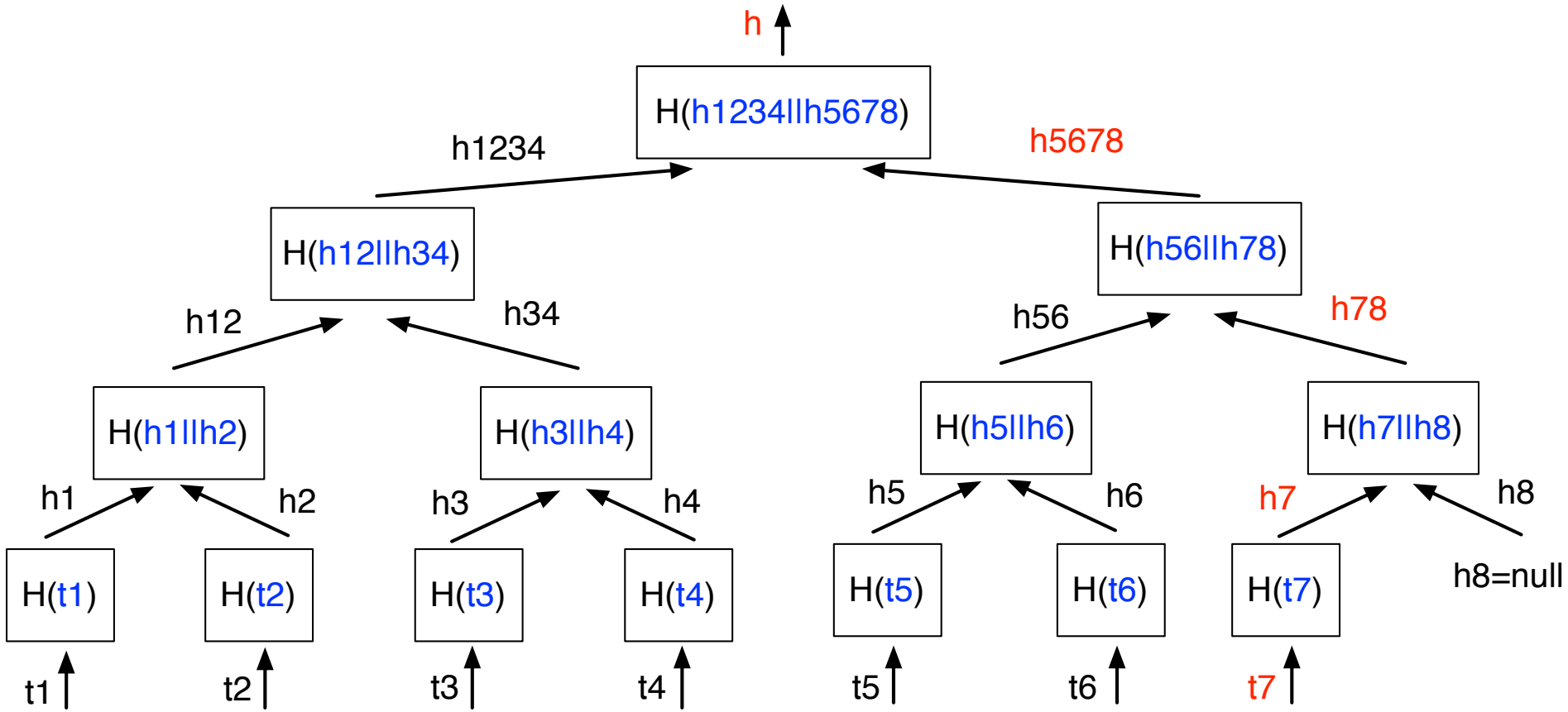
- Bank has 5 trans {t1, t2, t3, t4, t5}
  - Bank has a new trans t6
  - Bank needs to update h6, h56, h5678, h
  - O(logN) v.s. O(1) in hash chain

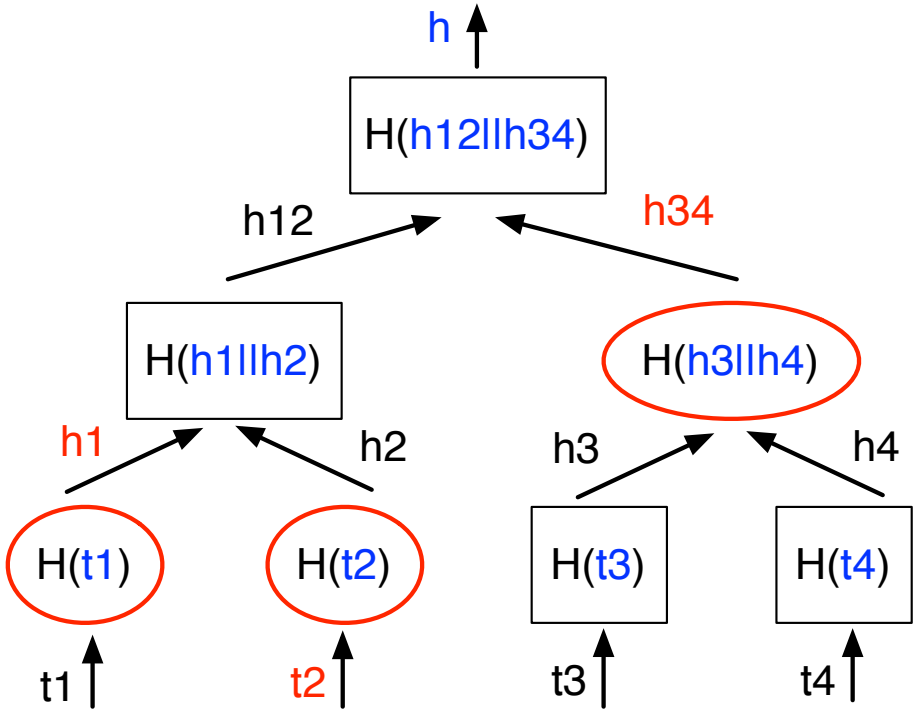


- Practice: Bank has 6 trans {t1, t2, t3, t4, t5, t6}
  - Bank has a new trans t7
  - Which hash values need to be recomputed?



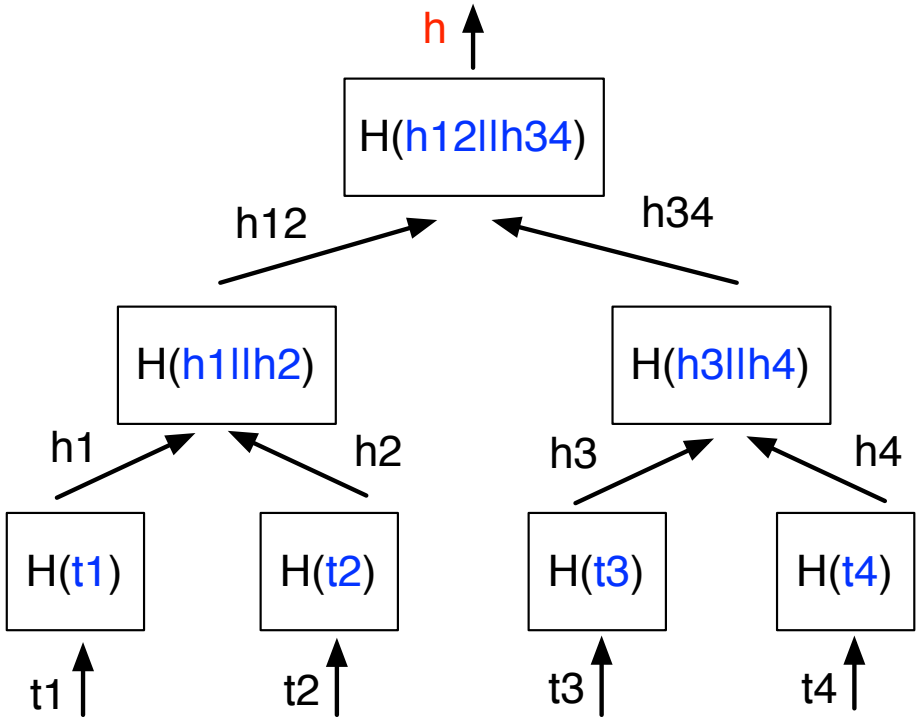
- Practice: Bank has 6 trans {t1, t2, t3, t4, t5, t6}
  - Bank has a new trans t7
  - Update h7, h78, h5678, h

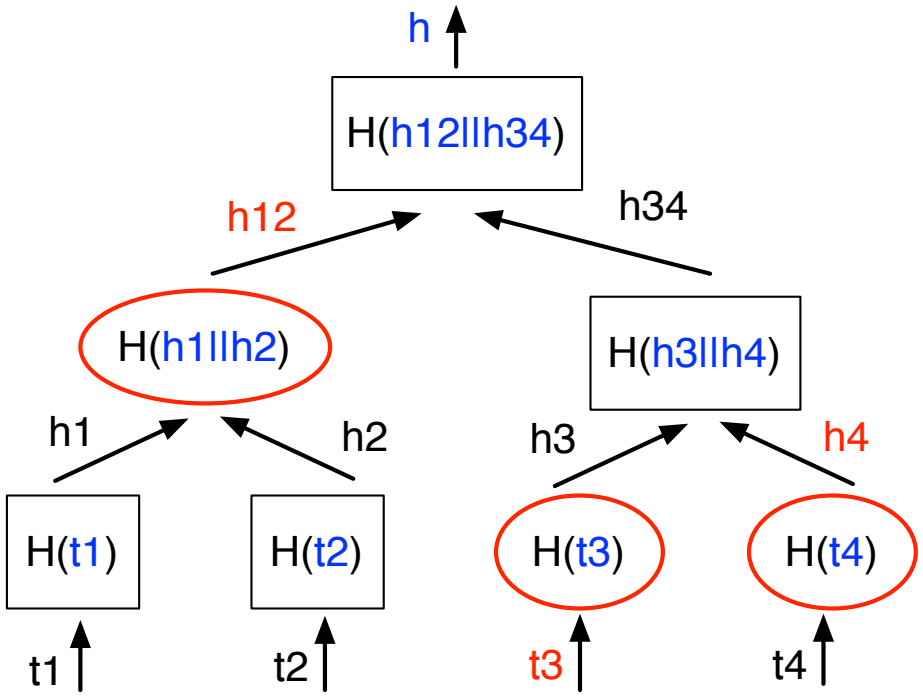




- Membership testing
- Alice wants to prove t2 is a member of all trans without knowing other trans.

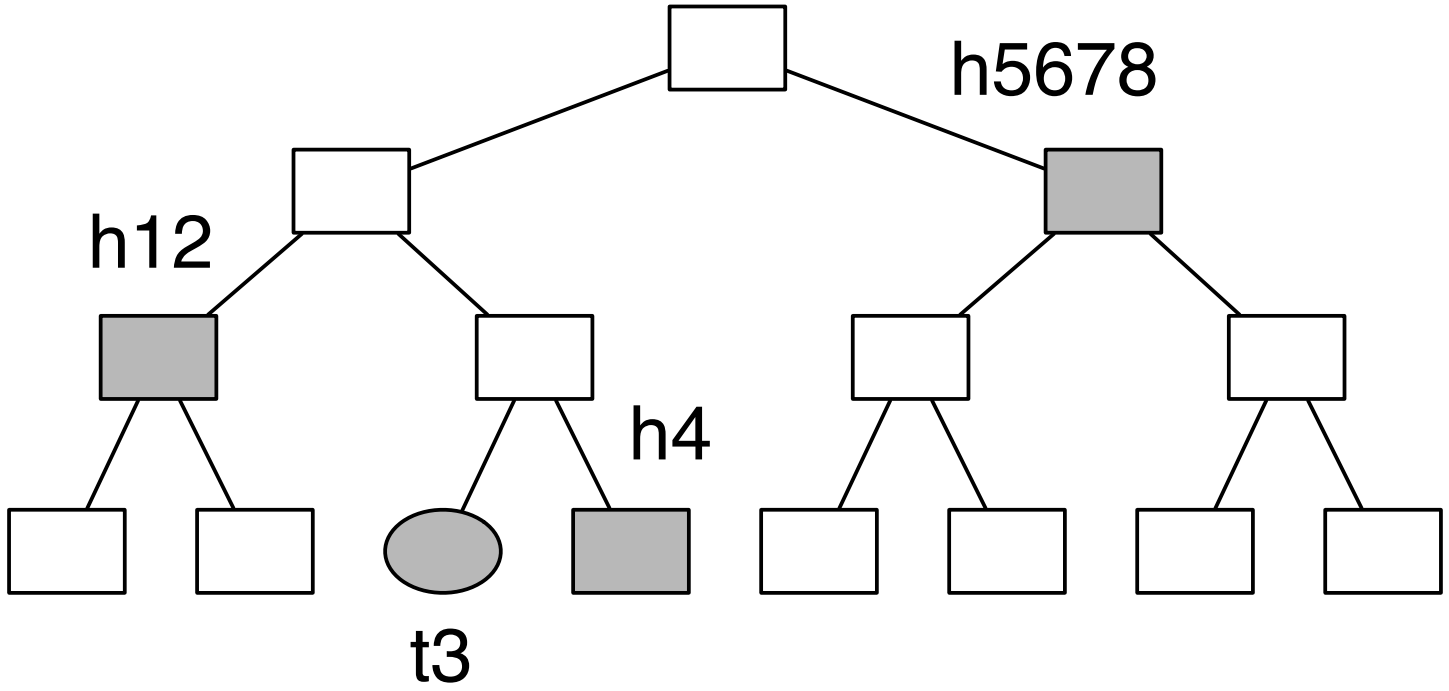
- Alice gets t2, h1, h34 from bank, recompute h
  - 1 trans + 2 hashes, in total O(logN)





- Alice wants to prove t3 is a member of all trans without knowing other trans.
- Practice: what information does Alice need?

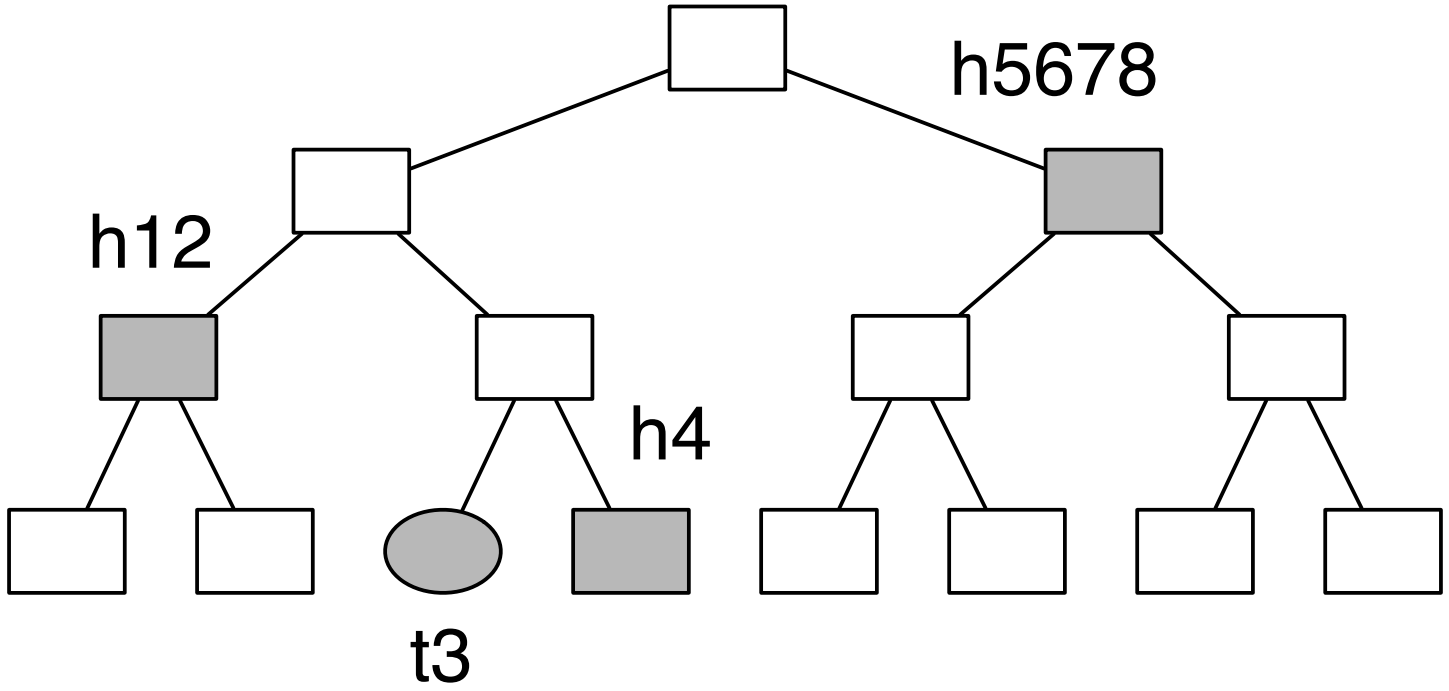
- Alice gets t3, h4, h12 from bank, recompute h



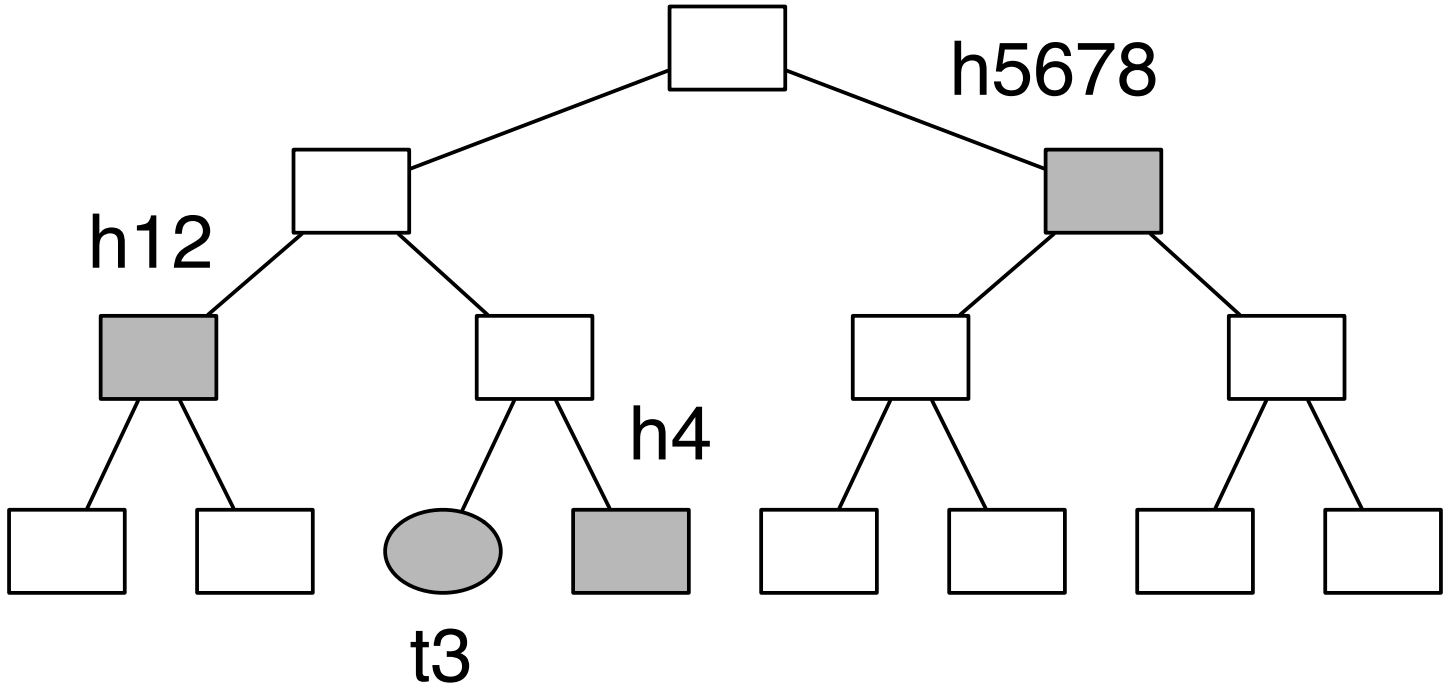
- Proving t3 is a member of all trans with Merkle tree
  - only need to get all the gray nodes O(logN)
    - 1 trans + 3 hash values
- Hash chain needs to get all trans O(N)
  - need t1, t2, t3, t4, t5, t6, t7, t8
    - 8 trans

Comparison

	Hash Chain	Merkle Tree
Compute Proof	N hashes	2N-1 hashes
Size of Proof	1 hash value	1 hash value
Append	O(1)	O(logN)
Update	O(N)	O(logN)
Proof of Membership	O(N)	O(logN)



- Use Merkle tree in P2P networks, e.g., BitTorrent
  - Alice downloads t3, a piece of movie
  - Alice needs to check whether this t3 is correct while other pieces have not been downloaded
  - Alice retrieves all the hashes of gray nodes
  - Alice computes root hash h', compare with h

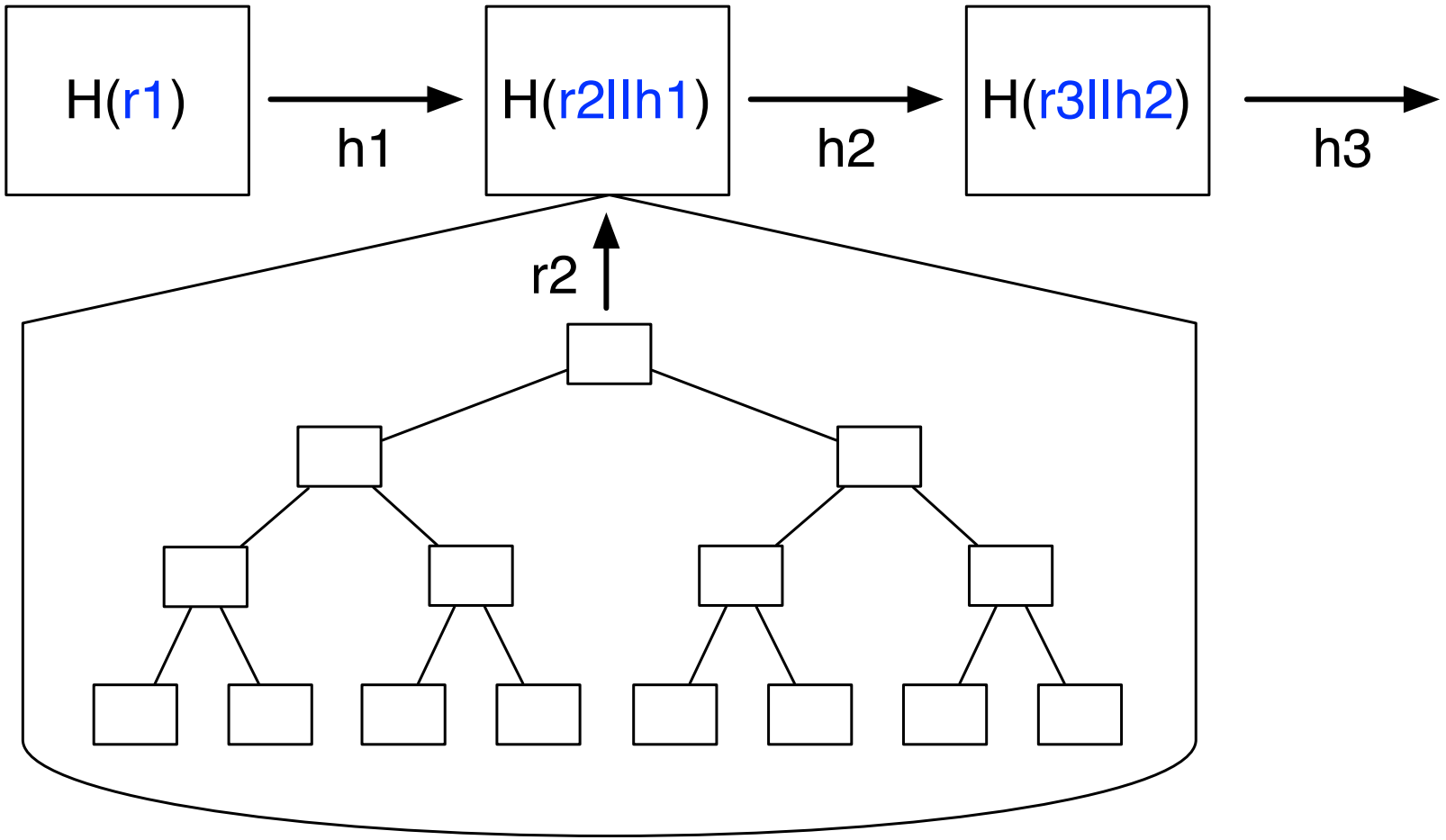


- Without using Merkle tree
  - Alice needs to wait the entire movie is downloaded (e.g., several hours later)
  - Waste bandwidth if the movie data (e.g., a small piece of the entire data) is not correct.

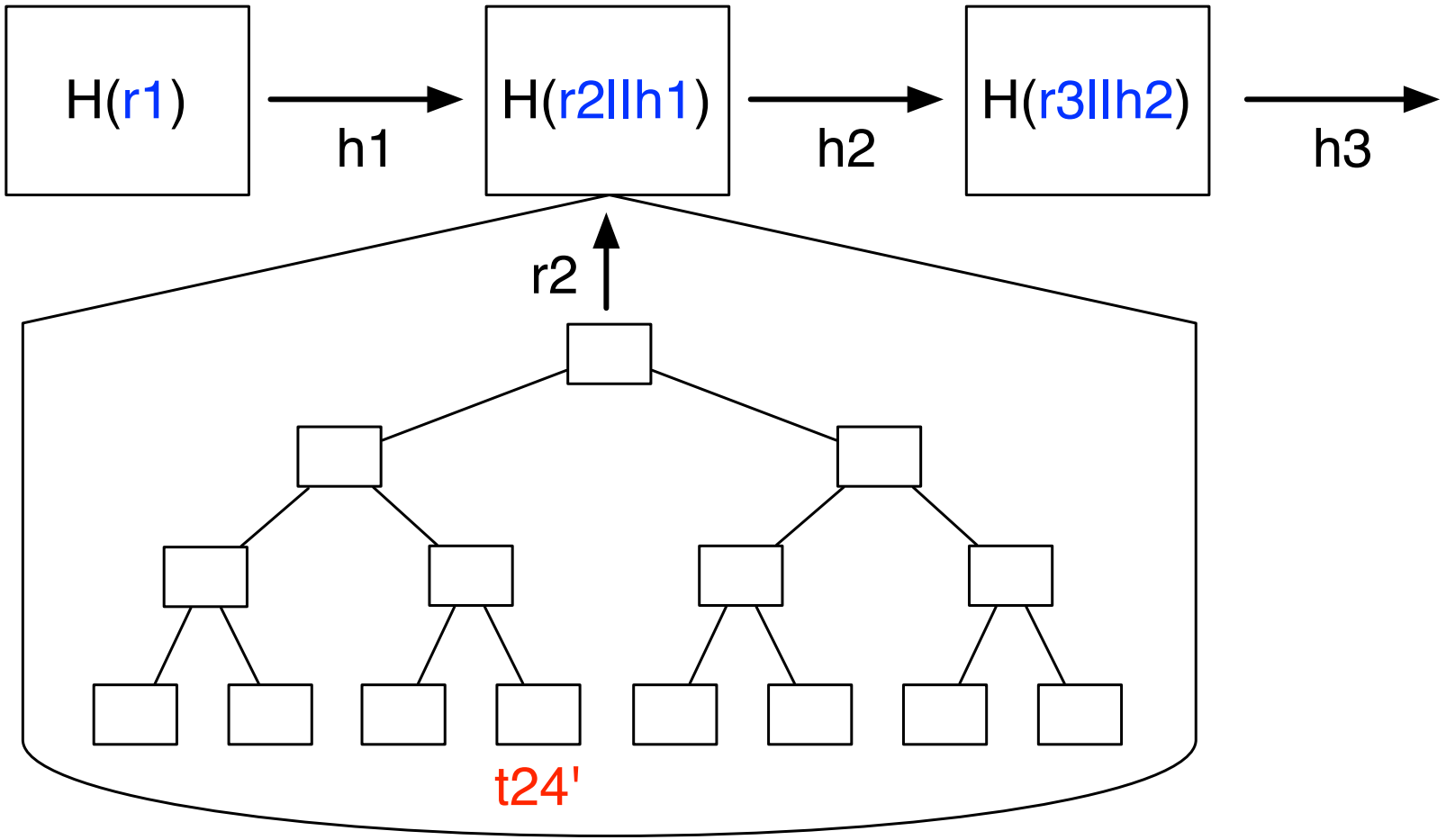
The Blockchain

- Bitcoin network has a large number of transactions.
  - About 196,000 trans per day
  - Bitcoin uses the blockchain to maintain integrity

- Blockchain includes Hash chain + Merkle tree
  - Each block has 1MB limit (1000~2000 trans)
  - Trans in one block form one Merkle tree
  - All the root hash values form a hash chain
  - Total blocks: 512,169 (as of 3/5/2018)



- Example: Assume each block has 8 transactions
  - 3 blocks, 24 transactions in total
  - Hash value **h3** proves all the 24 transactions



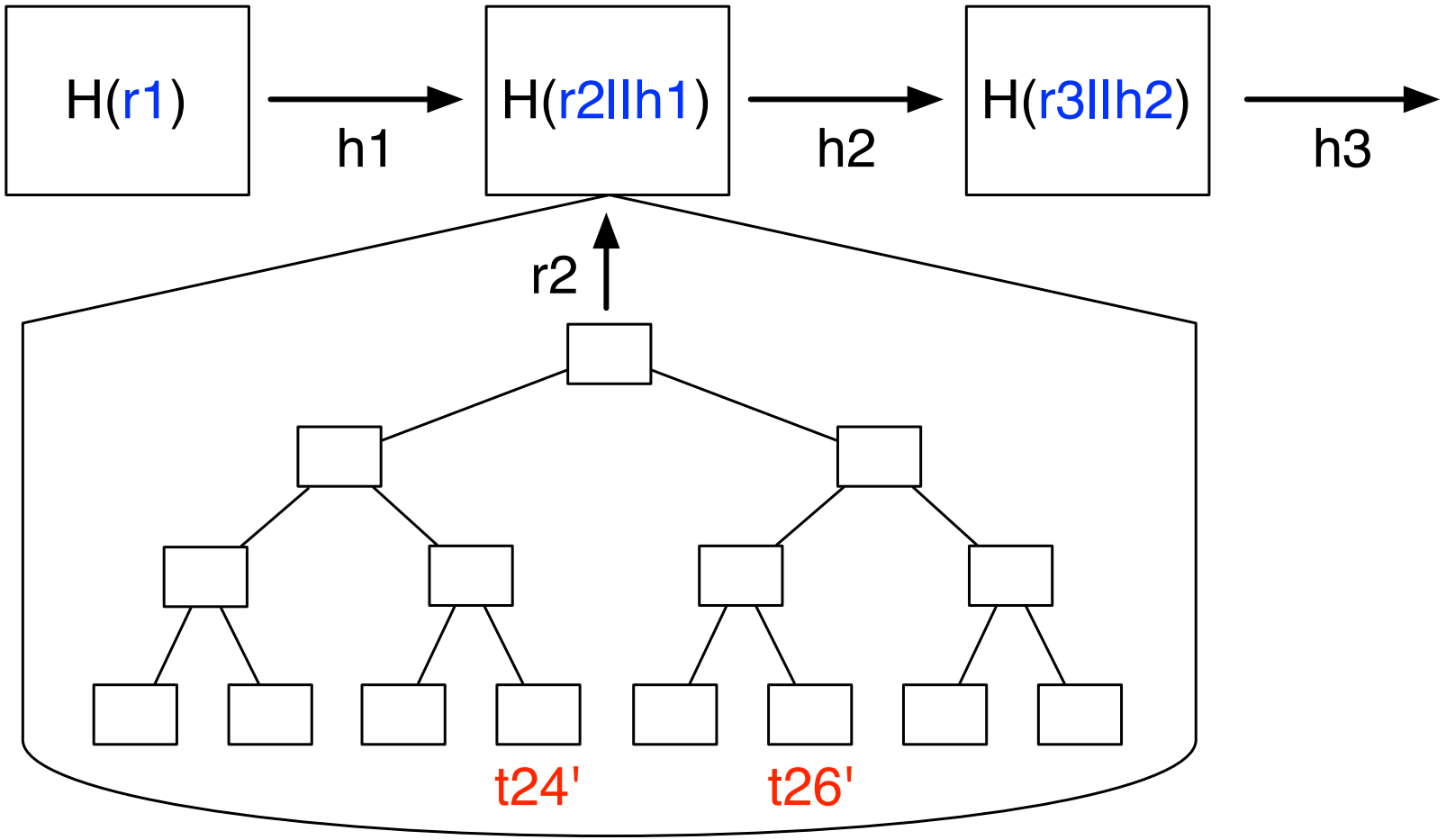
- Attacker wants to change 1 trans. (t24 to t24’)
  - Needs to find a collision s.t. H(t24) == H(t24’)
  - Happens with a negligible probability

Changes in Blockchain

- t24: Alice pays 2 bitcoins to Bob

- There is a later trans related to t24
  - t26: Bob pays (the) 2 bitcoins to Charlie
  - t26 also includes info of t24

- Attacker changes t24 to t24’
- Attacker also needs to change t26 to t26’



- Attacker wants to change 1 trans. (t24 to t24’)
  - t26 is associated with t24, needs to be changed
  - Needs to find 2 collisions s.t.
    - H(t24) == H(t24’) AND H(t26) == H(t26’)

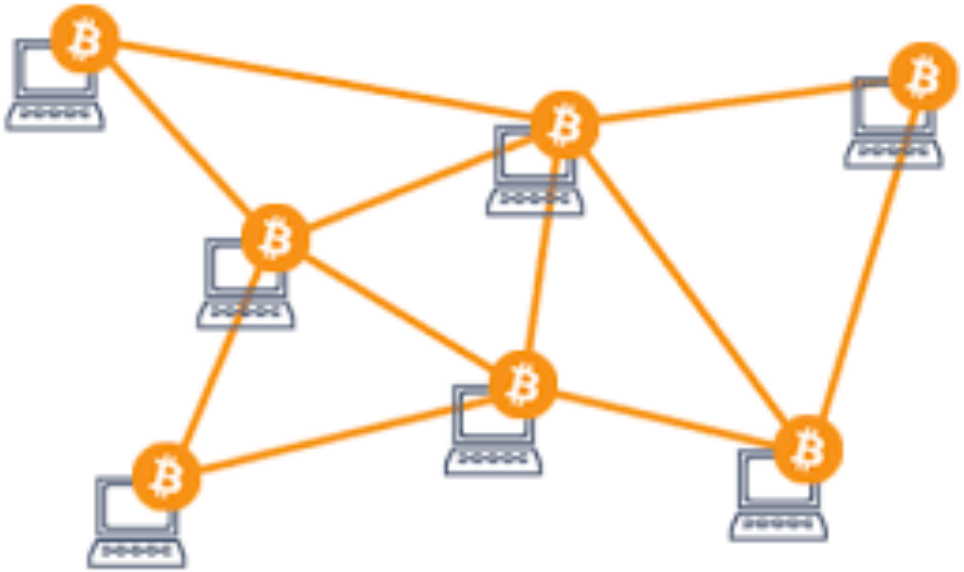
Changes in Blockchain

- Attacker changes t24 to t24'
  - If there are n-1 later trans associated with t24
  - Assume the prob. of finding a collision is p
    - p is negligible
  - Total prob. of success: p^n
  - A trans is more secure if it is older.

- Blockchain can be used distributed scenario
  - E.g., Supply chain, Internet of Things

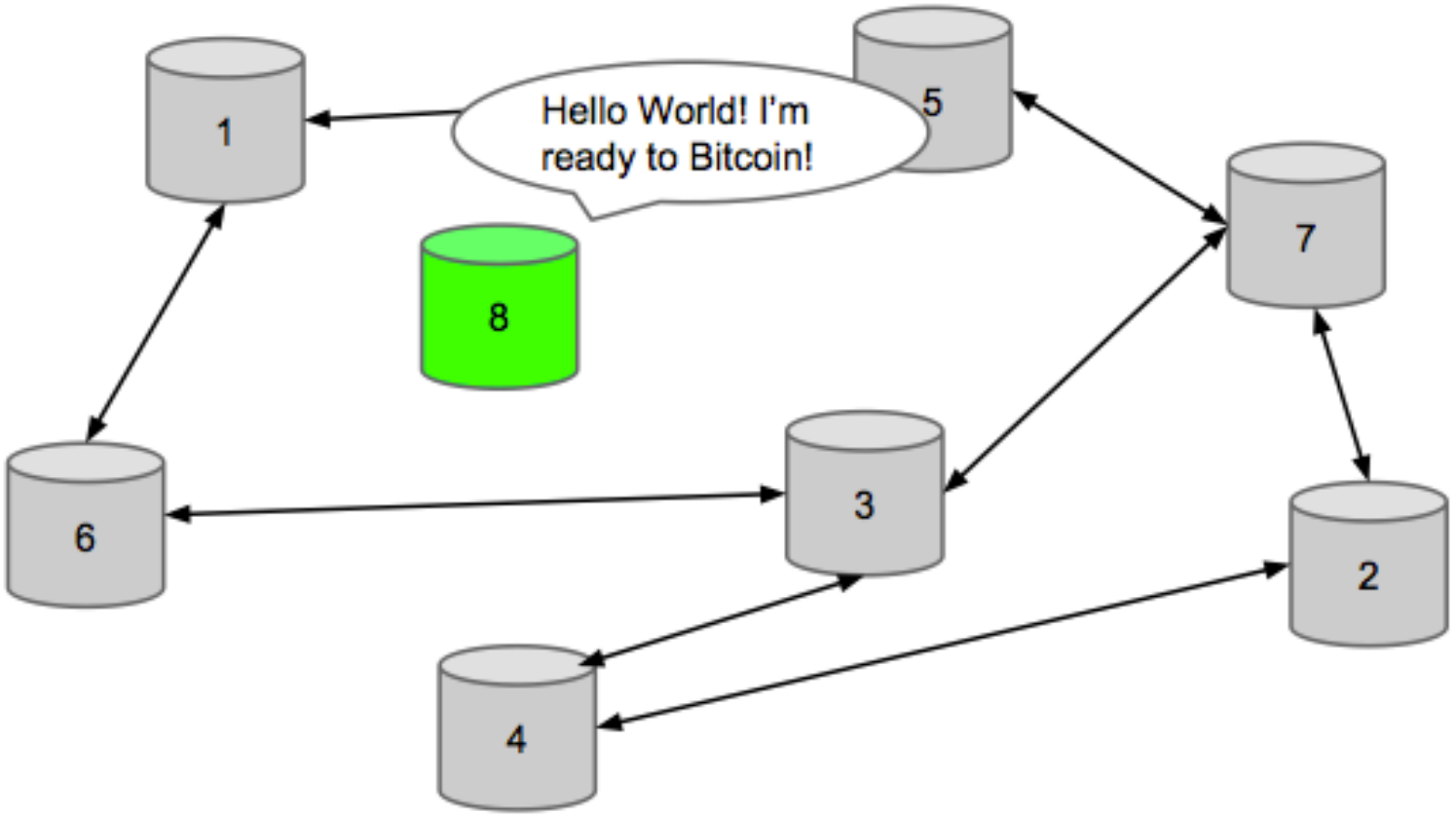
Bitcoin Network

- Bitcoin network is a peer-to-peer network
  - No central authority
  - All nodes are equal
  - Forget non-responding nodes after 3 hrs



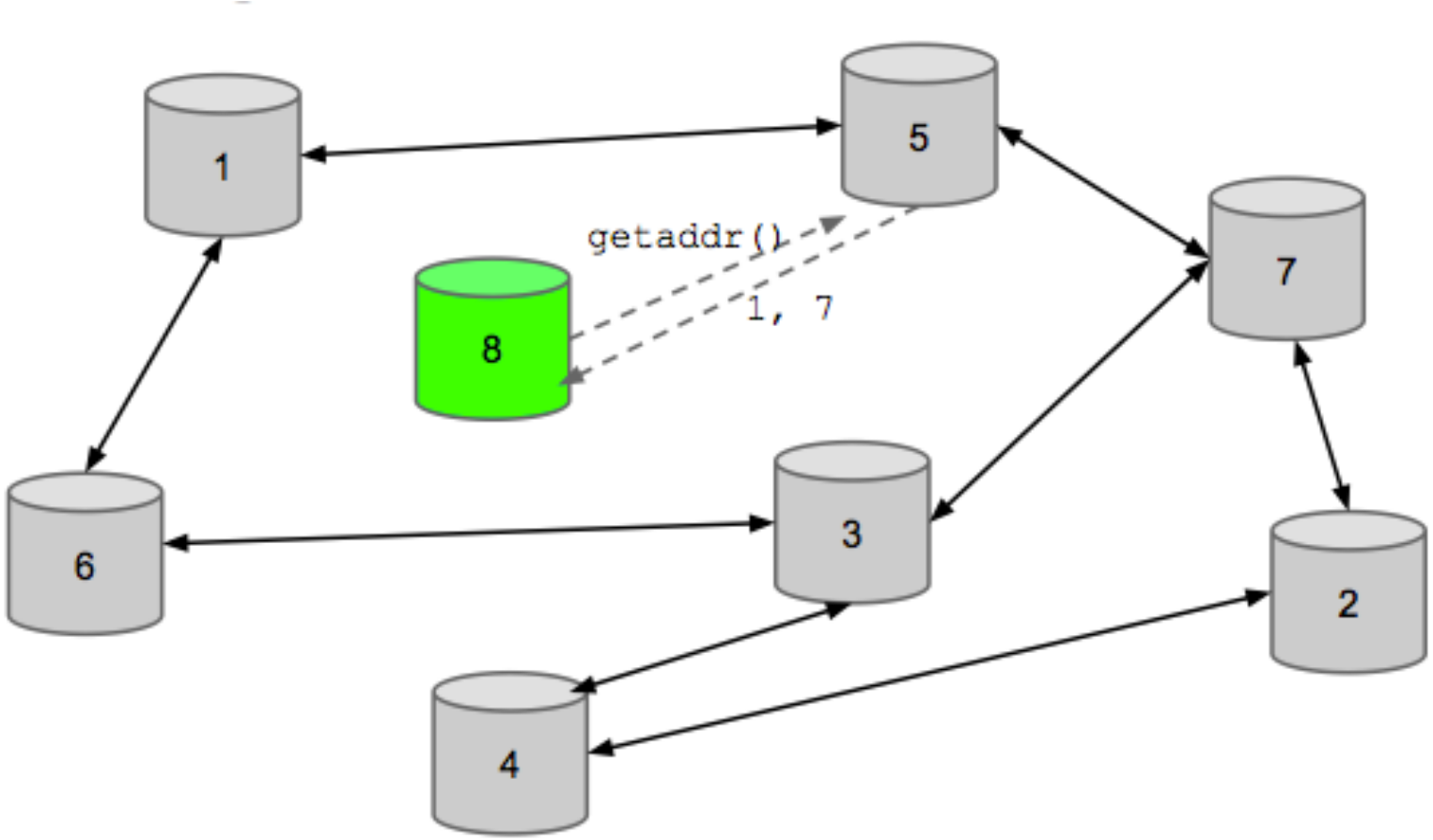
Join Bitcoin Network

- Can join anytime



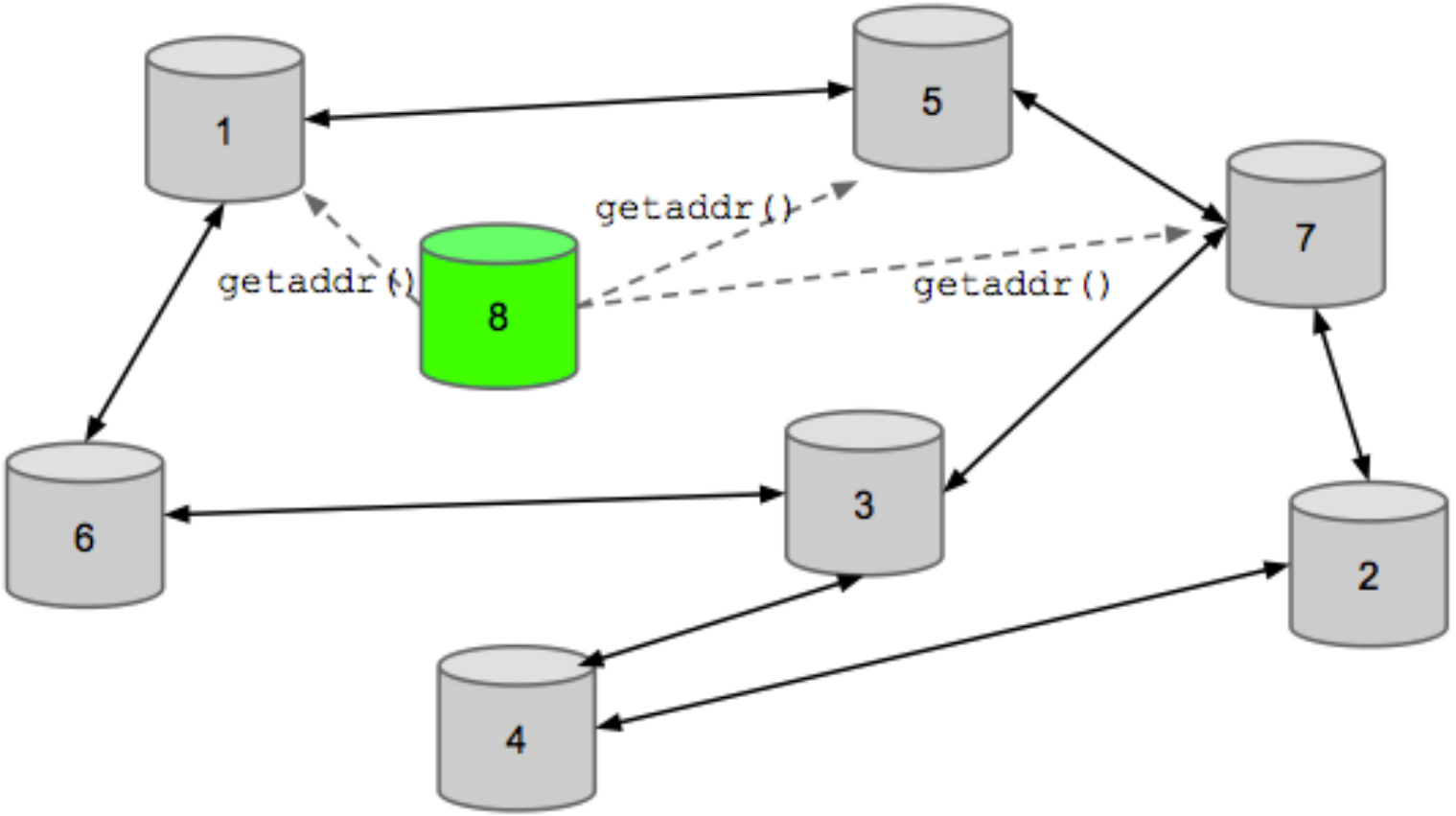
Join Bitcoin Network

- Find one node, and get its neighbors



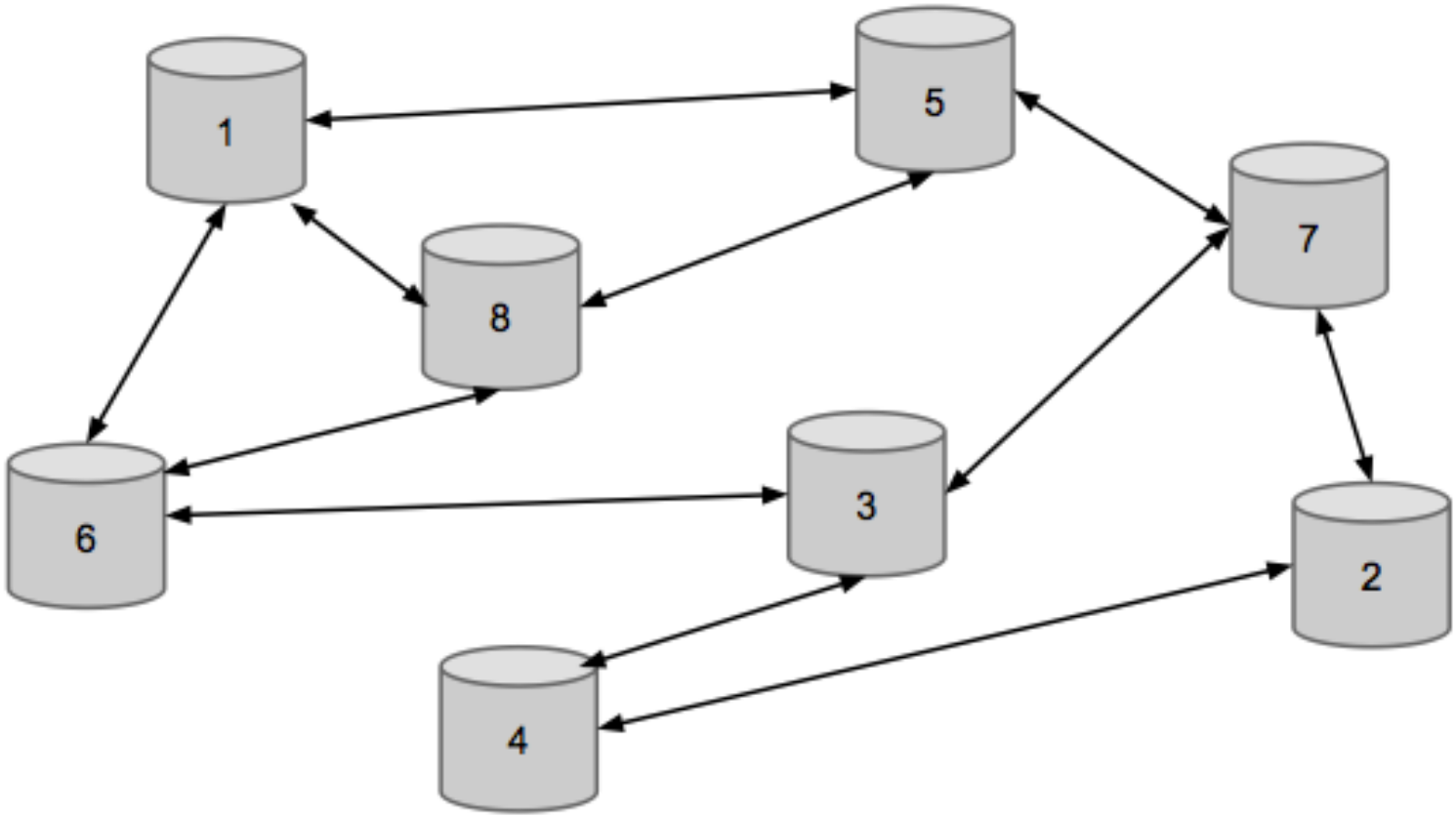
Join Bitcoin Network

- Based on neighbors, find neighbors' neighbors



Join Bitcoin Network

- Connect to nodes (some response, some not)



Join Bitcoin Network

- Two types of nodes
  - Bitcoin users (use Bitcoins)
  - Miners (use & generate Bitcoins)
- Each node has a public key and a private key
- Use hash value of public key as account number (e.g., routing number on your check)
  - SHA256: H(pk), then convert to base56

1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2

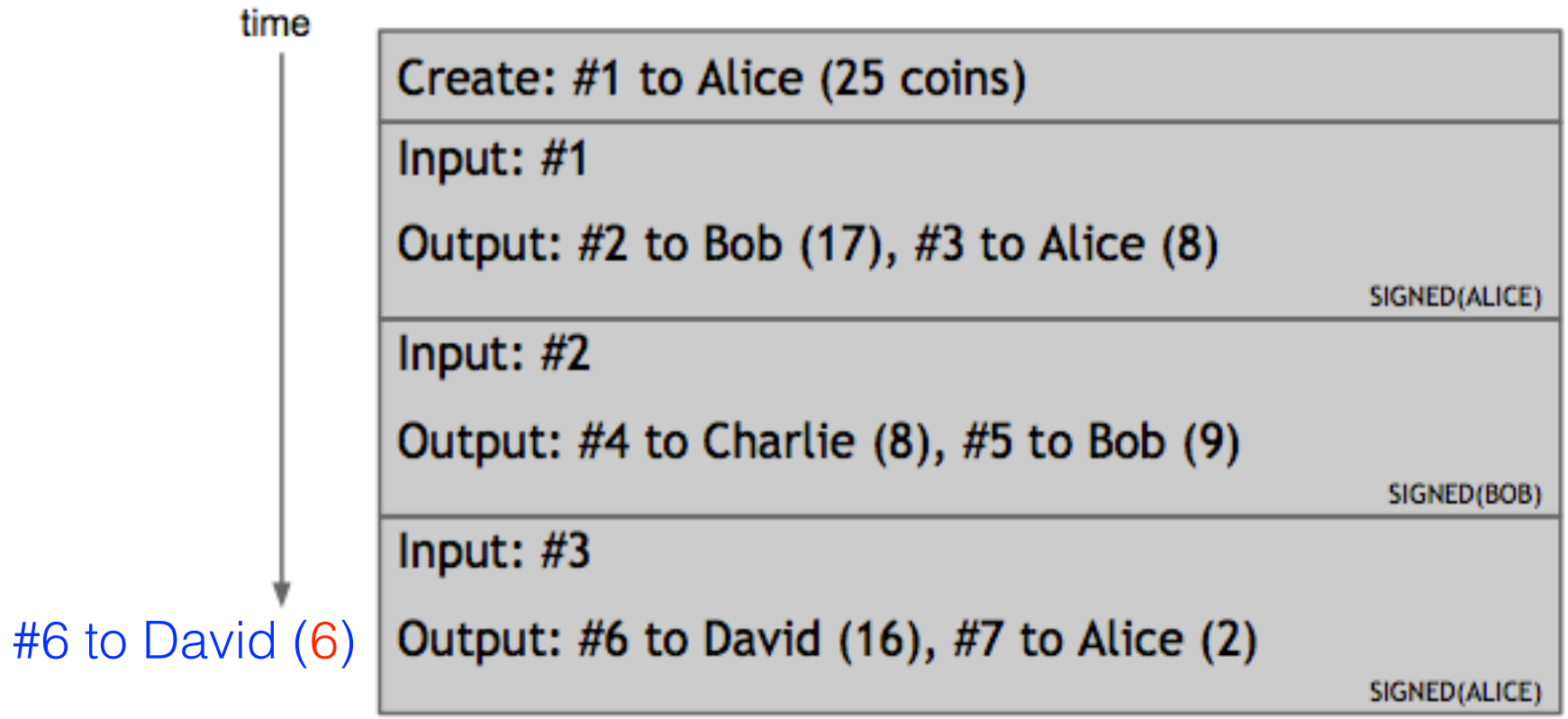
1JBonneauruSSoYm6rH7XFZc6Hcy98zRZz

Bitcoin Transaction

- Alice has 25 bitcoins
  - Transaction: Alice pays Bob 17 bitcoins
  - Alice signs this trans with her private key
  - Others can verify this trans with Alice's public key
- Transactions are coin-based
  - 1 trans destroys old "coins" and create new ones
  - E.g., Alice destroys its old coin with 25 bitcoins, create a new coin with 17 bitcoins for Bob, and create a new coin with 8 bitcoins for herself

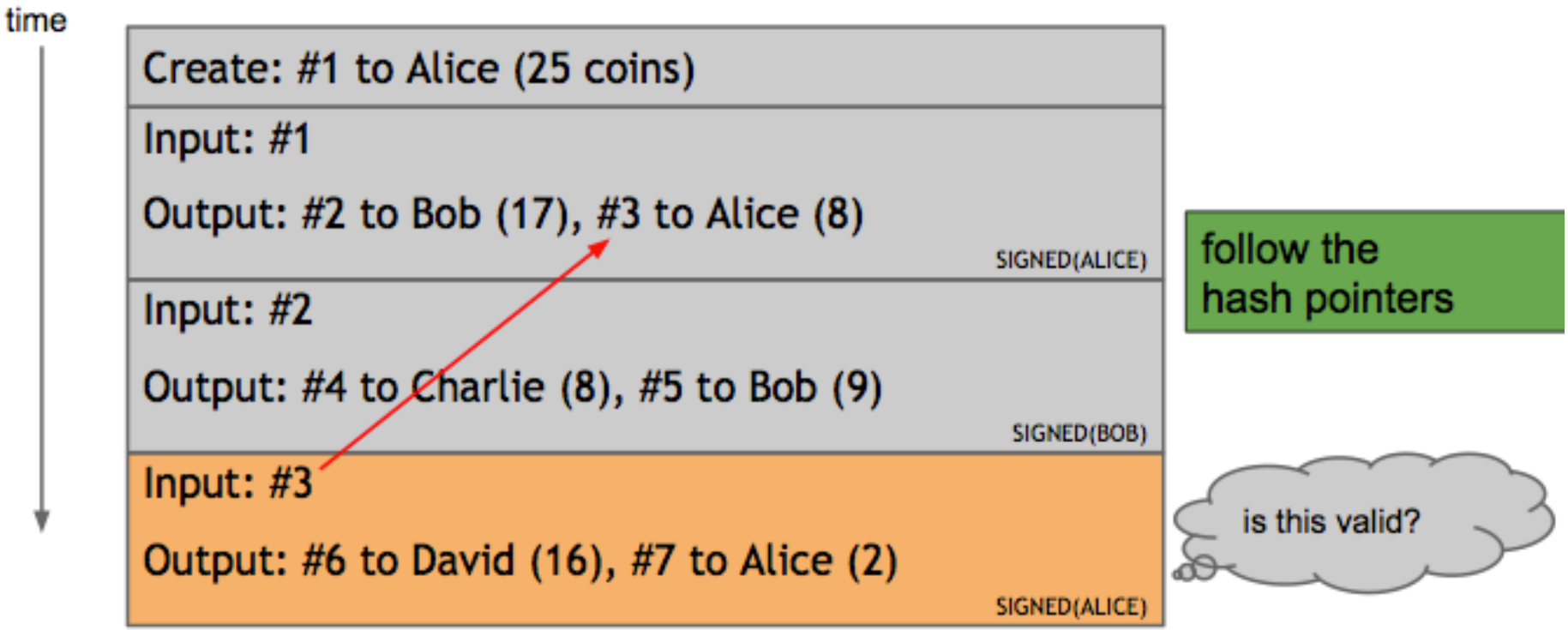
Bitcoin Transaction

- Transactions are coin-based
  - E.g., coin #1 —> coin #2 & coin #3



Bitcoin Transaction

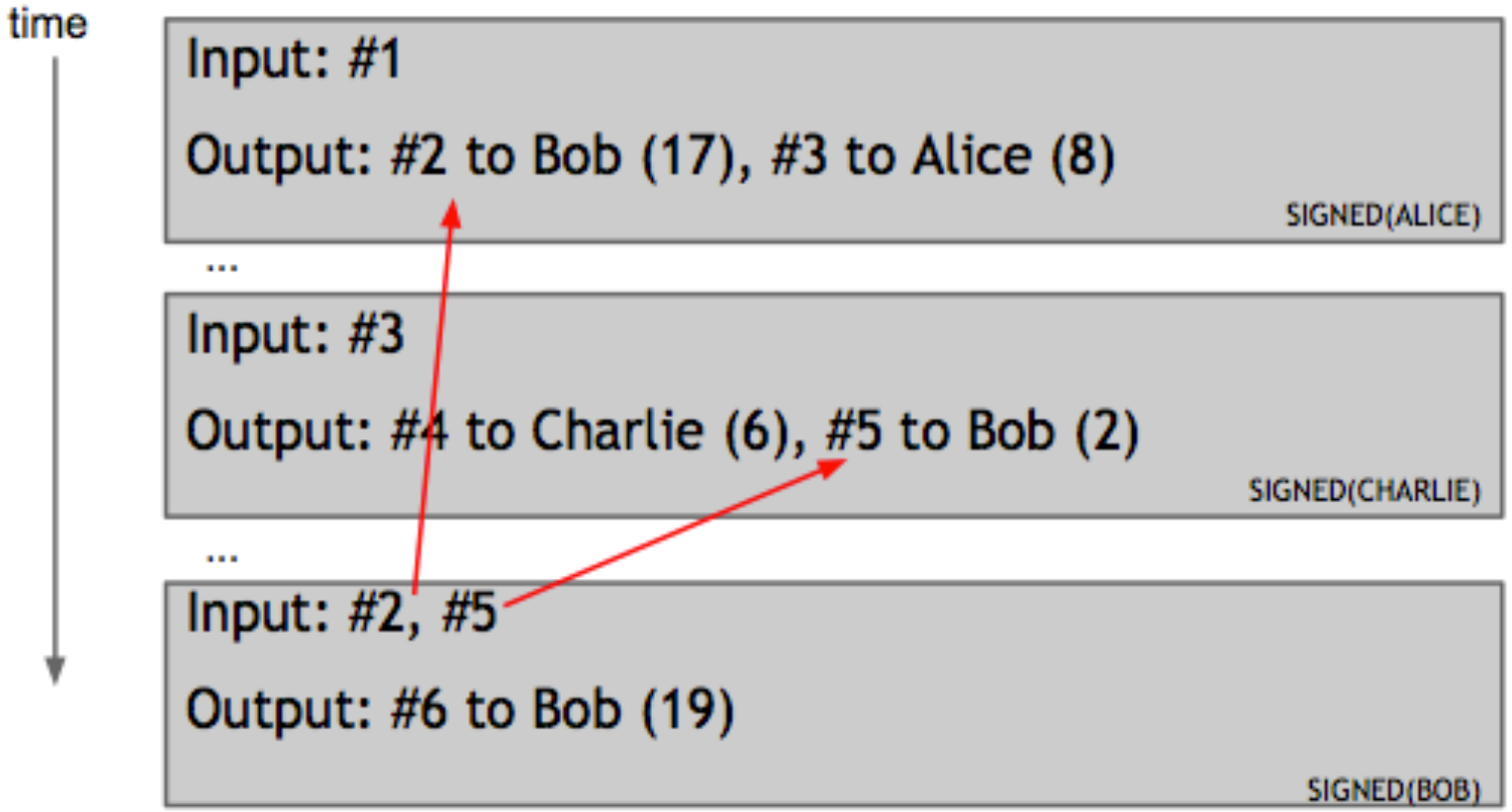
- A coin is related to its previous coin
  - E.g., Check coin #3 needs to check coin #1



#6 to David (6)

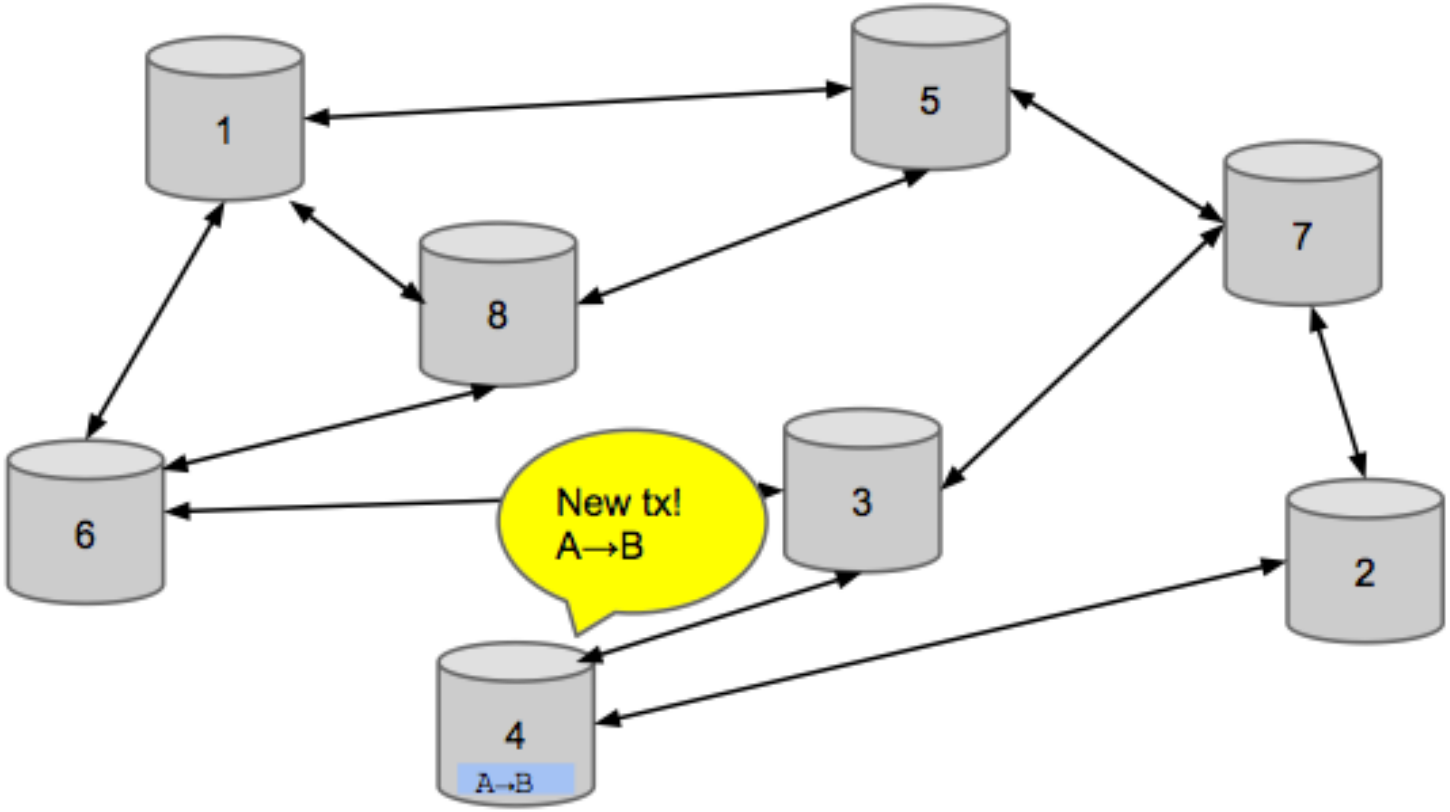
Bitcoin Transaction

- Bob can merge two coins into 1 coin
  - E.g., coin #2 and coin #5 —> coin #6



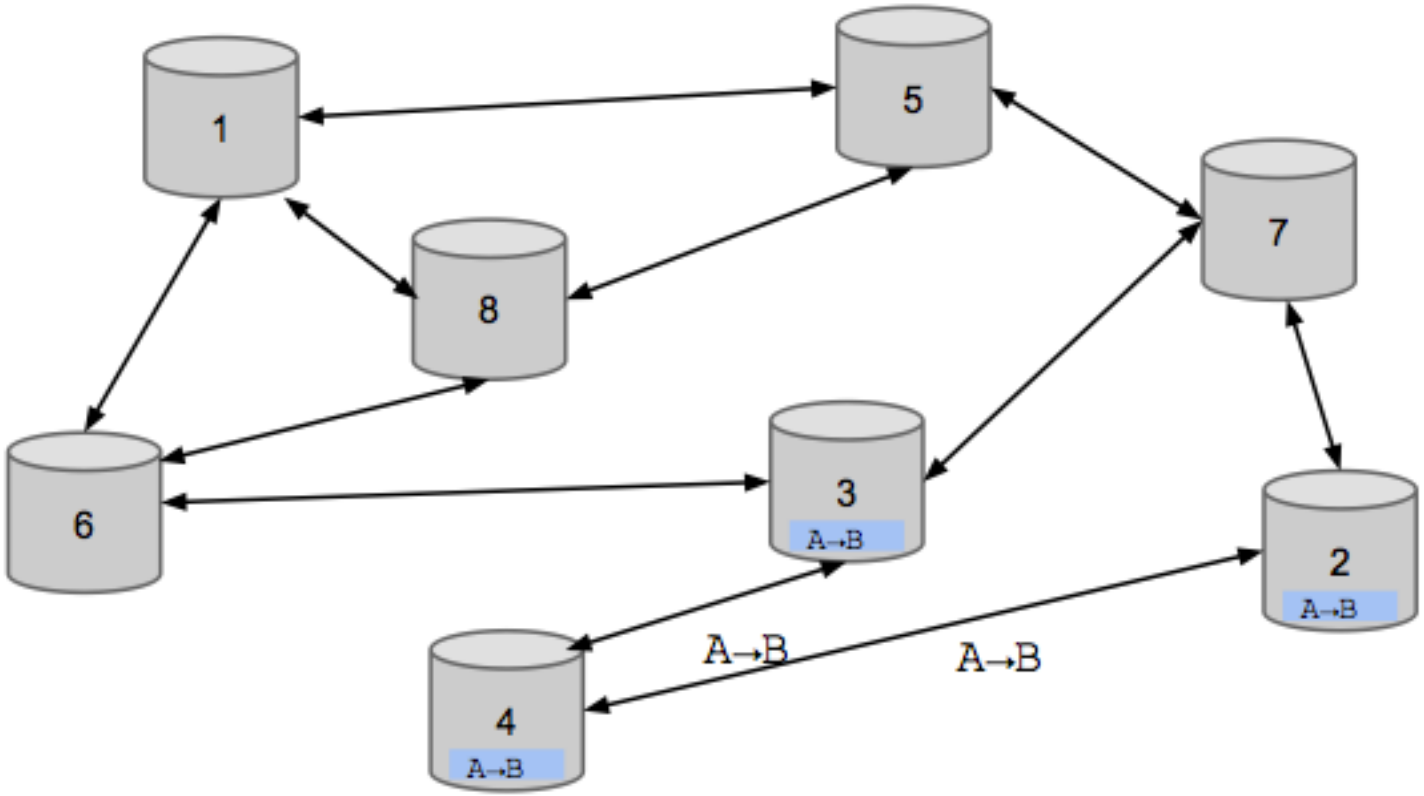
Transaction Propagation

- Each trans will be propagated to (almost) the entire network



Transaction Propagation

- Each trans will be propagated to (almost) the entire network



Transaction Propagation

- Each trans will be propagated to (almost) the entire network