Lecture 04
Hash Pointers and Data Structures

Dr. Shujaat Hussain

Some Bonuses

- Class participation - 1 absolute for each correctly answered bonus question
- The limit is currently one per person for the course
  - Should the limit be increased?

Do not distribute …

- These slides are not always prepared by me.
- Most of the content comes from the reference book
  - Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction
  - Arvind Narayanan, Joseph Bonneau,
  - Edward Felten, Andrew Miller, Steven Goldfeder

Cryptographic Hash Functions

Hash function:

- takes any string as input
- fixed-size output (we'll use 256 bits)
- efficiently computable

Cryptographic Hash Functions

Security properties:

- collision-free
- hiding
- puzzle-friendly

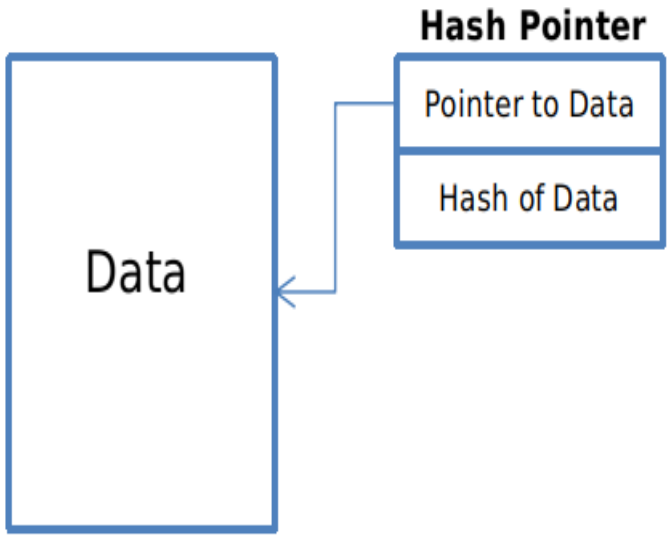
Hash Pointers and Data Structures

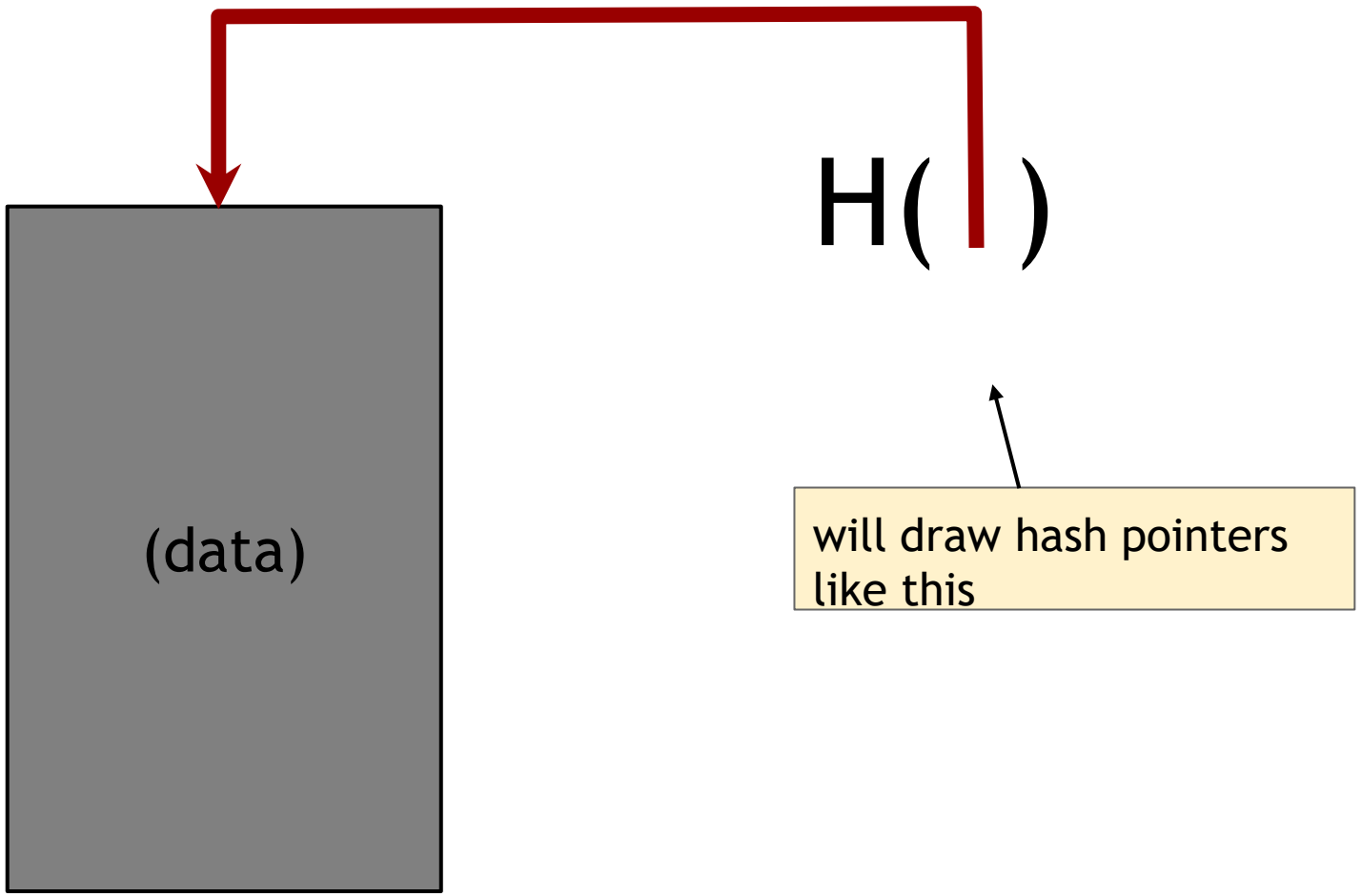
hash pointer is:

* pointer to where some info is stored, and
* (cryptographic) hash of the info

if we have a hash pointer, we can

* ask to get the info back, and
* verify that it hasn't changed

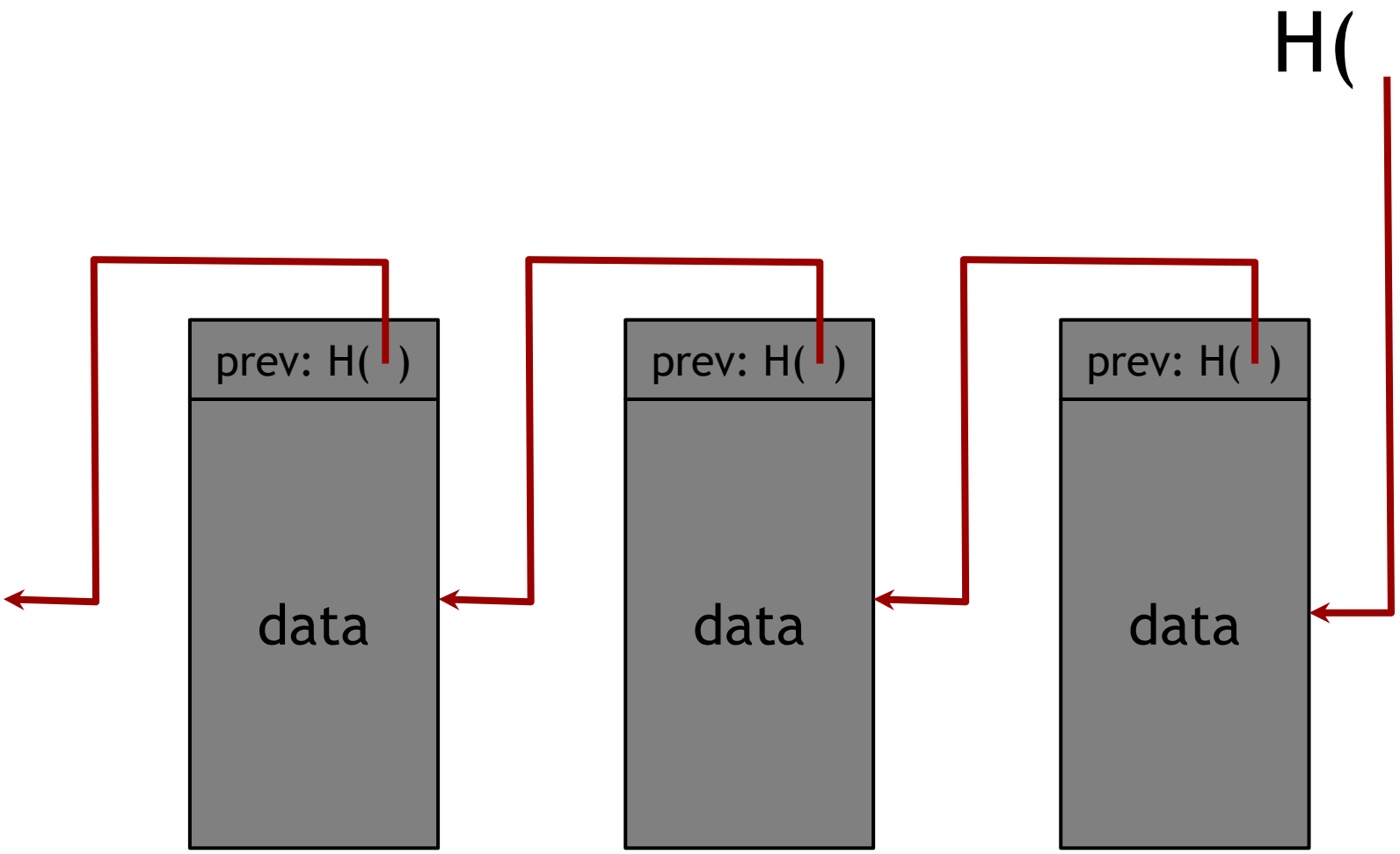



H( )
(data)
will draw hash pointers like this

key idea:

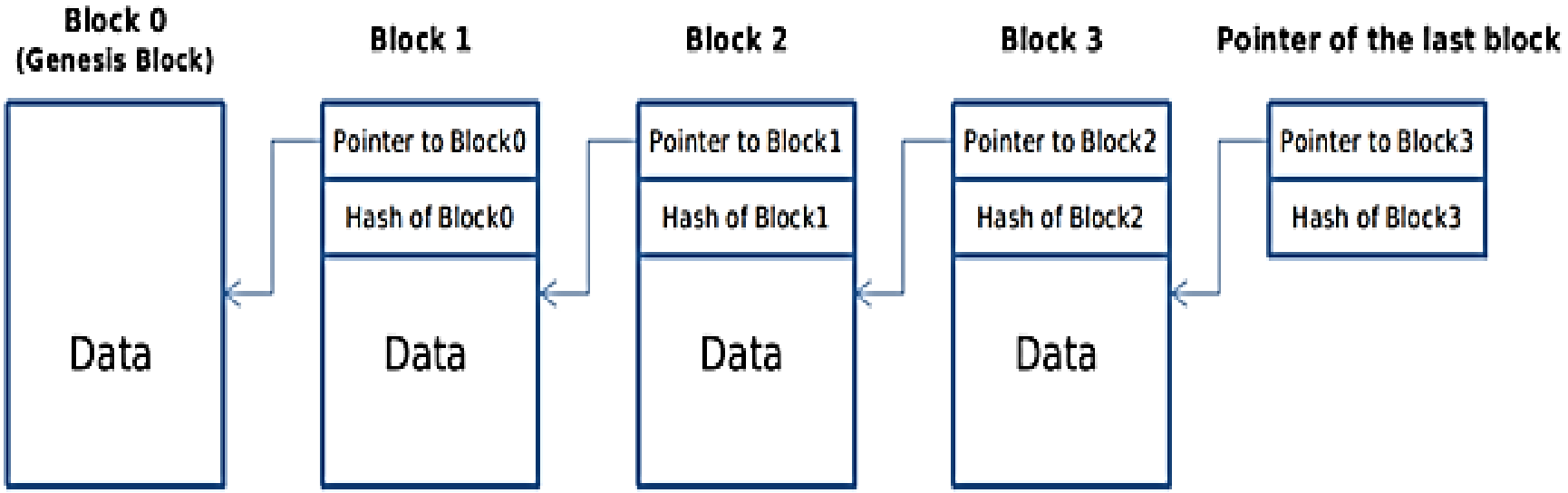
build data structures with hash pointers

linked list with hash pointers = “block chain”



use case: tamper-evident log

linked list with hash pointers = “block chain”



each block not only tells us where the value of the previous block was, but it also contains a digest of that value that allows us to verify that the value hasn’t changed.

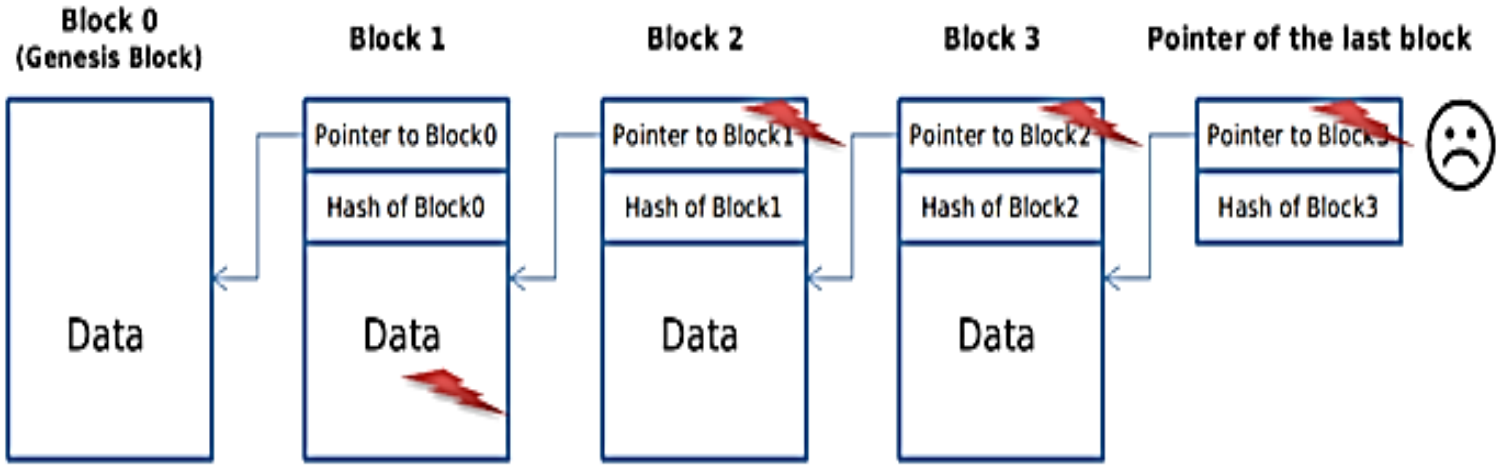
Assignment 01

- Create a Blockchain in golang https://golang.org

- It will make it very easy to do network programming – a task for next assignments.

detecting tampering

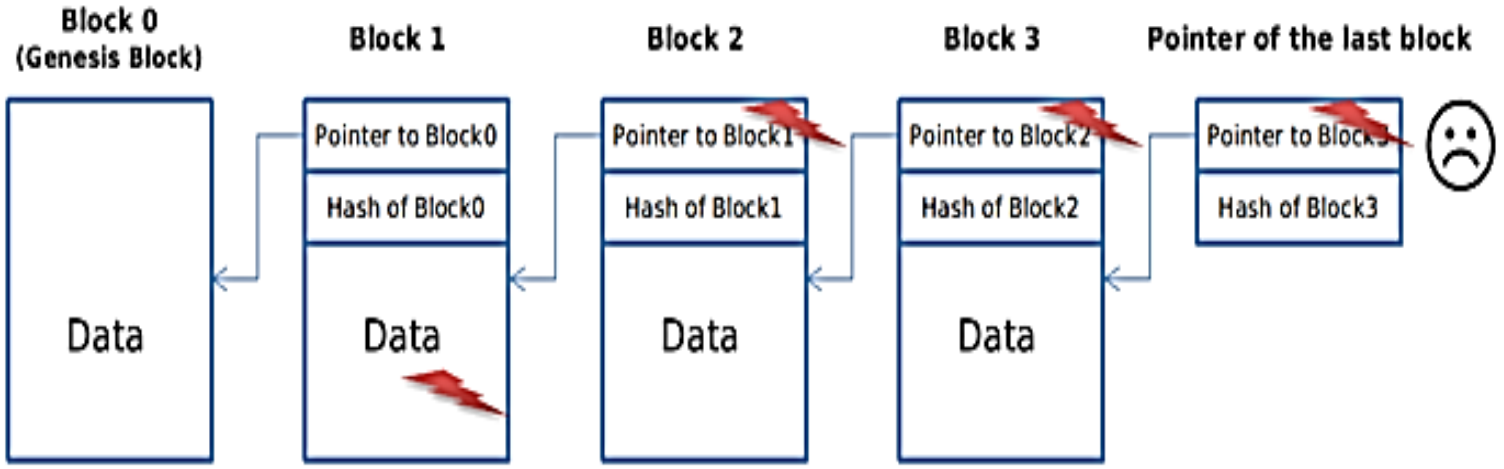
use case: tamper-evident log



- An attacker wants to tamper with one block of the chain, let's say, block 1.
- The attacker changed the content of block 1, because of "collision free" property of the hash function, he is not able to find another data which has the same hash with the old one. So now the hash of this modified block is also changed.

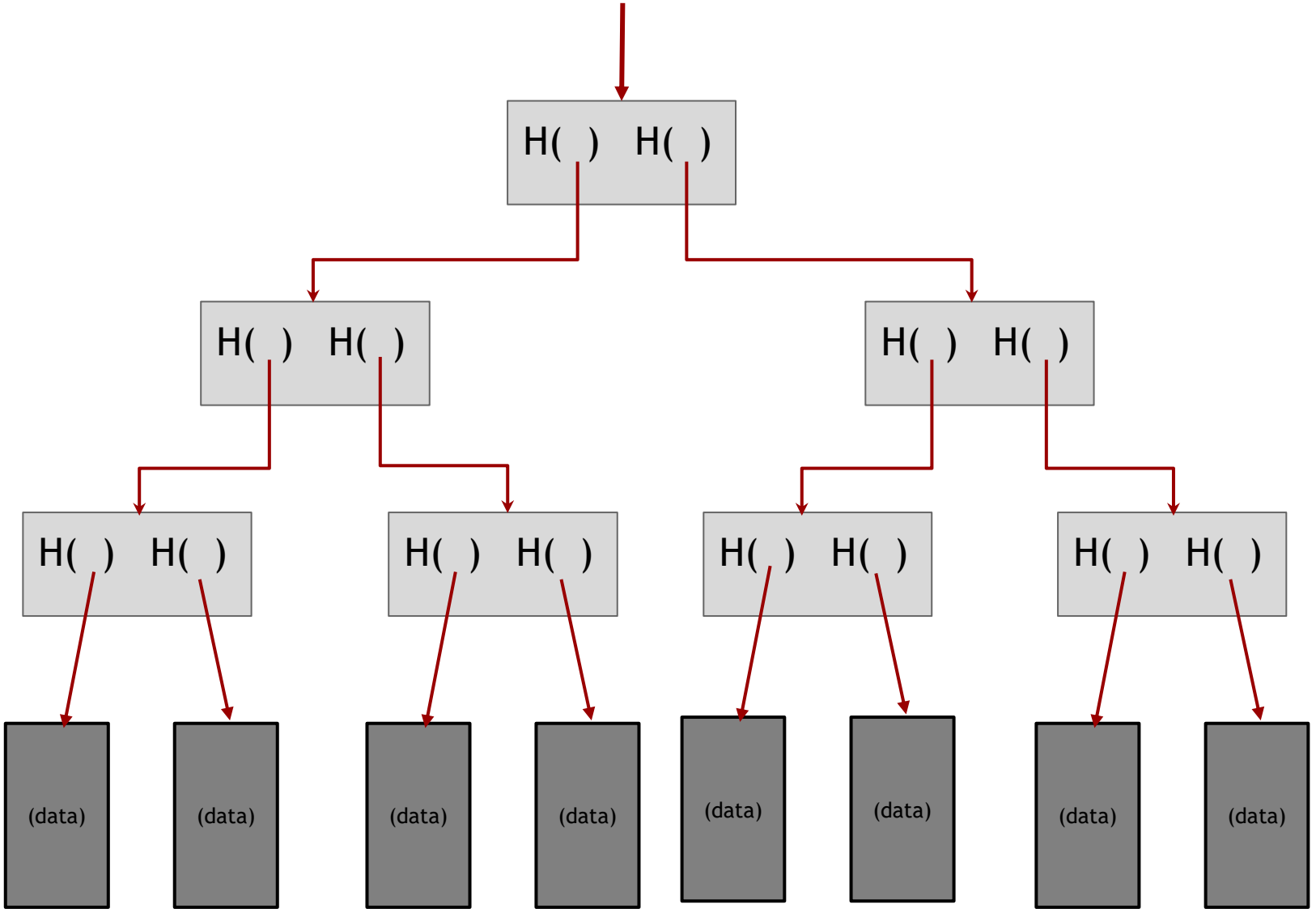
detecting tampering

use case: tamper-evident log



- To avoid others noticing the inconsistency, he also needs to change the hash pointer of that block in the next block, which is block 2.
- Now the content of block 2 is changed, so to make this story consistent, the hash pointer in block3 must be changed.
- Finally, the attacker goes to the hash pointer to the last block of the blockchain, which is a roadblock for him, because we keep and remember that hash pointer.

binary tree with hash pointers = “Merkle tree”



Application: InterPlanetary File System (IPFS

- Today's web is inefficient and expensive
- Today's web uses location based addressing – this leads to redundancy

The centralized web: Location-based addressing

- URLs are based on the location where data is stored,
- Through the domain name, URLs indicate which *authority* we should go to for the data.
- This also results in lot of redundancy

The decentralized web: Content addressing

- On the decentralized web, we can host each other's data
- Cryptographic hashing enables content addressing
- Hashes can be derived from the content of the data itself, if hashes are same so is the content

- When we want a specific photo of an adorable pet, we ask for it by its content address (hash).
- Who do we ask? The whole network!

Data structures - Decentralized Web

- The Web is not only data
- The content is linked together as well

- How can we structure data using content based addressing?

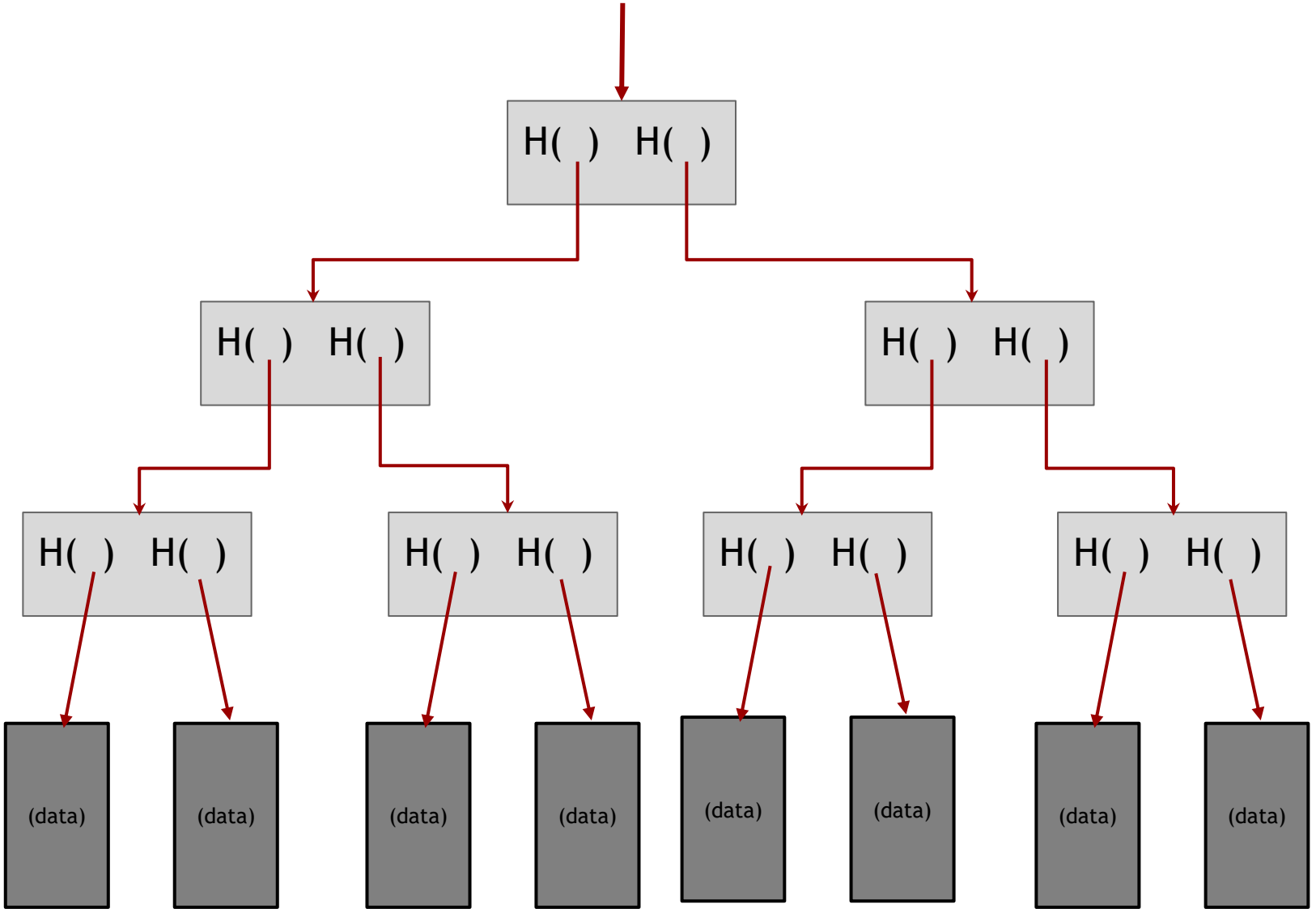
Case Study: Distributing Large Datasets

- The person sharing the file is responsible for maintaining the file server and its associated costs
- The same server is likely used to respond to all the requests
- The data itself may be distributed monolithically, as a single file archive
- It’s hard to locate alternative providers of the same data
- It’s hard for others to share datasets that build on the original data

Case Study: Distributing Large Datasets

- Anybody who wants can help distribute the file
- Nodes from all over the world can participate in serving the data
- Each part of the DAG has its own CID that can be distributed independently
- It’s easy to find alternative providers of the same data
- The nodes forming the DAG are small, and can be downloaded in parallel from many different providers
- Larger datasets encompassing the original can simply link the original dataset as a child of a larger DAG

…back to bitcoin

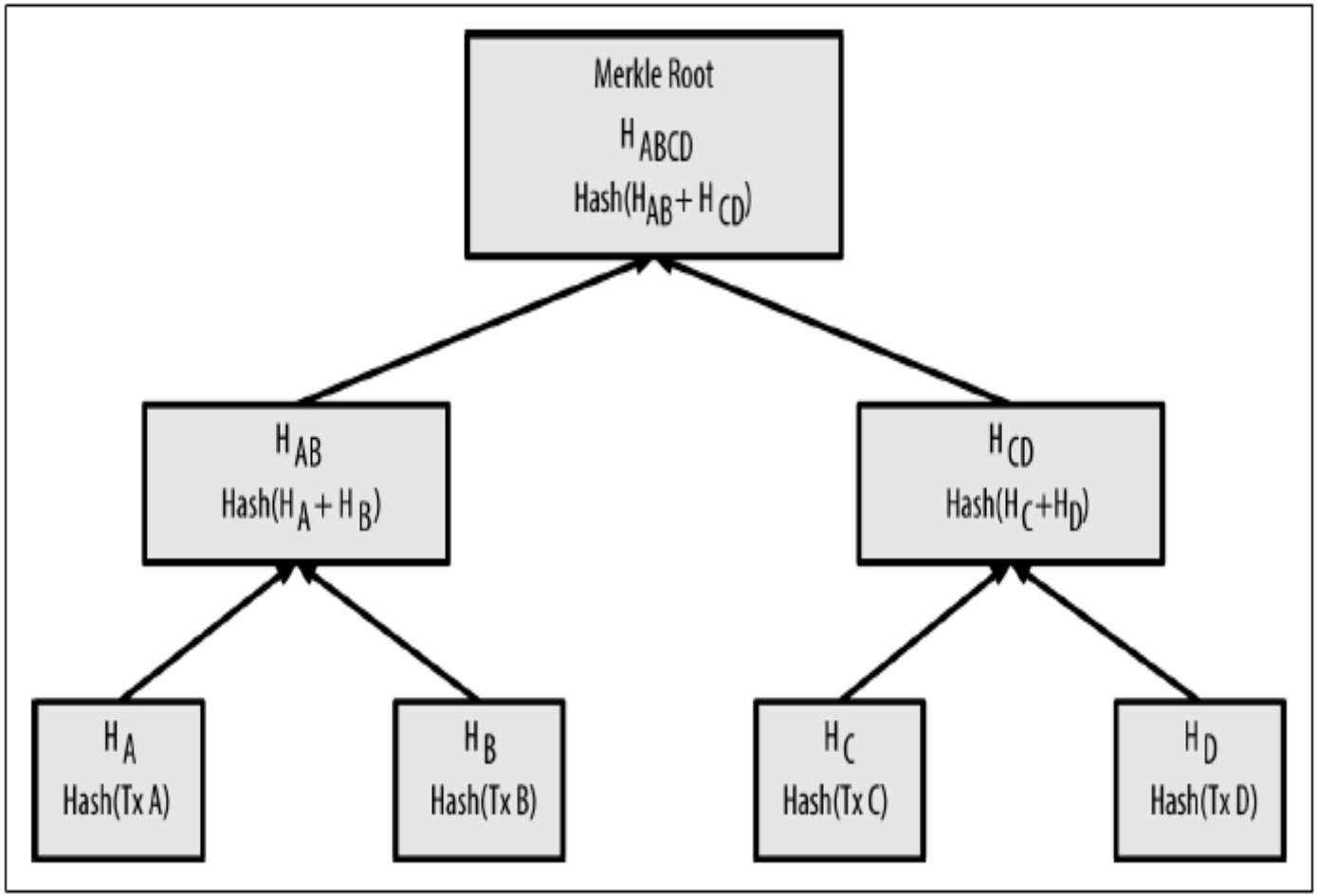


binary tree with hash pointers = "**Merkle tree**"

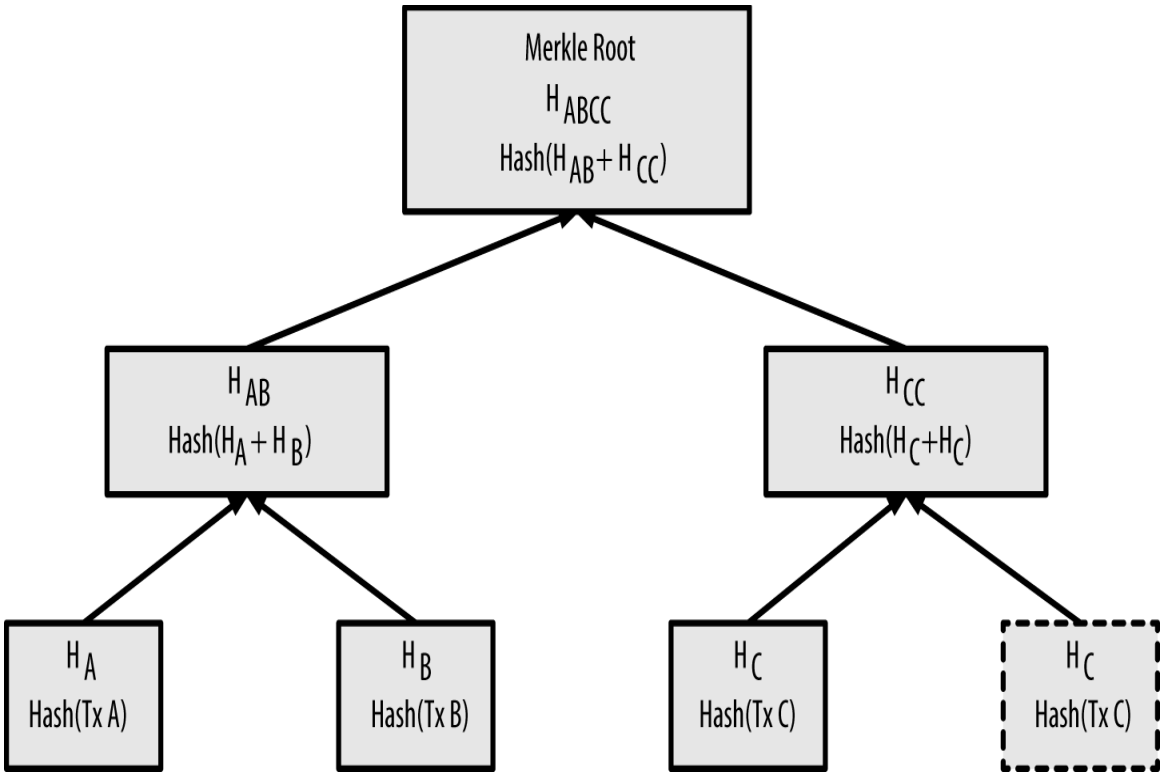
- Merkle trees are used in bitcoin to summarize all the transactions in a block, producing an overall digital fingerprint of the entire set of transactions, providing a very efficient process to verify whether a transaction is included in a block.
- The merkle tree is constructed bottom-up. In the following example, we start with four transactions, A, B, C, and D, which form the leaves of the merkle tree, as shown in the Figure below.
- The transactions are not stored in the merkle tree; rather, their data is hashed and the resulting hash is stored in each leaf node as HA, HB, HC, and HD:

HA = SHA256(SHA256(Transaction A))

binary tree with hash pointers = “Merkle tree”



- Because the merkle tree is a binary tree, it needs an even number of leaf nodes. If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a balanced tree.

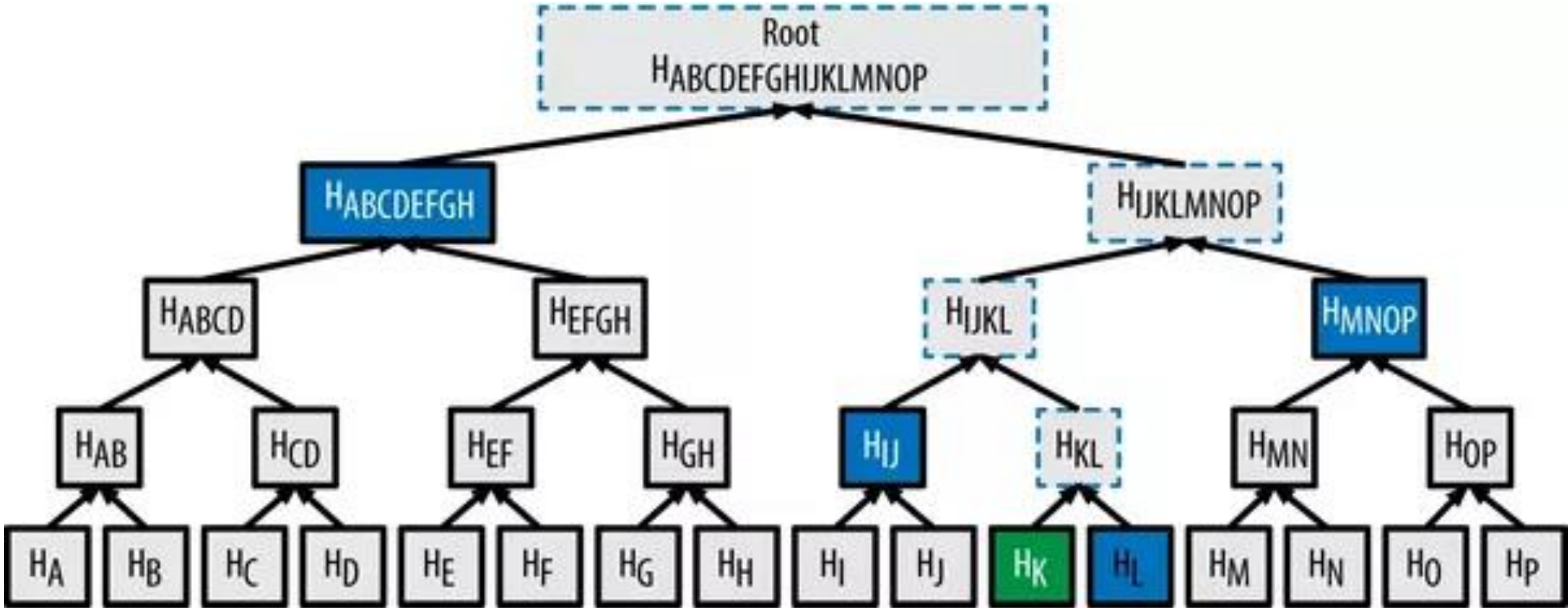


Why they are used?

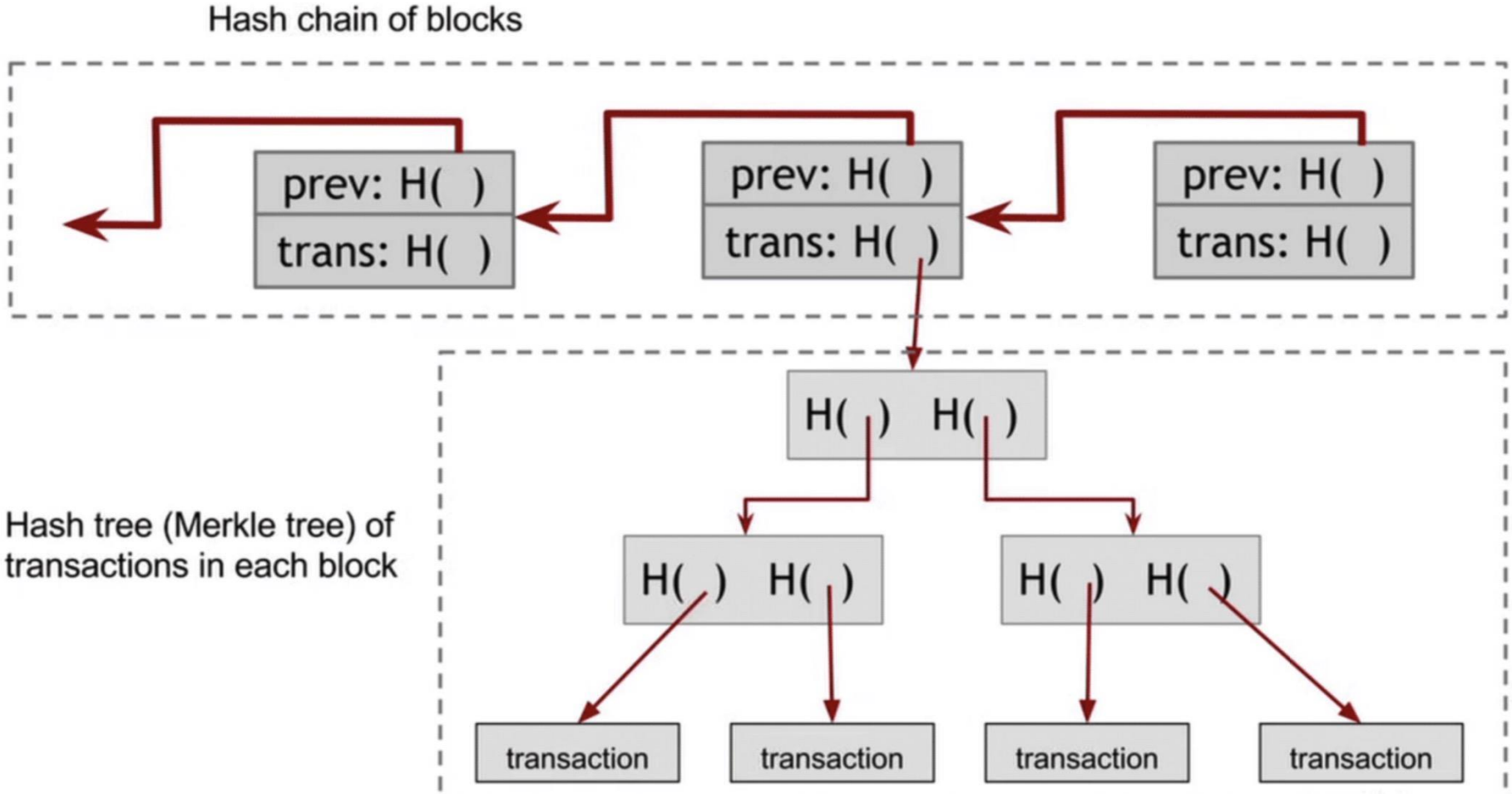
- They serve many purposes,
  - They can be used to verify membership in O(log n) time/space
  - Membership of transaction in a block !!
  - We can use something called a “Merkle proof” to show that some content is actually part of this tree. For example, let’s prove that A is part of the above tree. All we need to do is provide each of A’s siblings on the way up, recompute the tree, and make sure everything matches.

Example

- Is transaction K part of the tree?



Where they are used?



Where they are used?

- A light client may need fewer details.
- Can verify membership in O(log n) time/space

Simplified Payment Verification

- It should be possible to verify that a certain transaction has happened

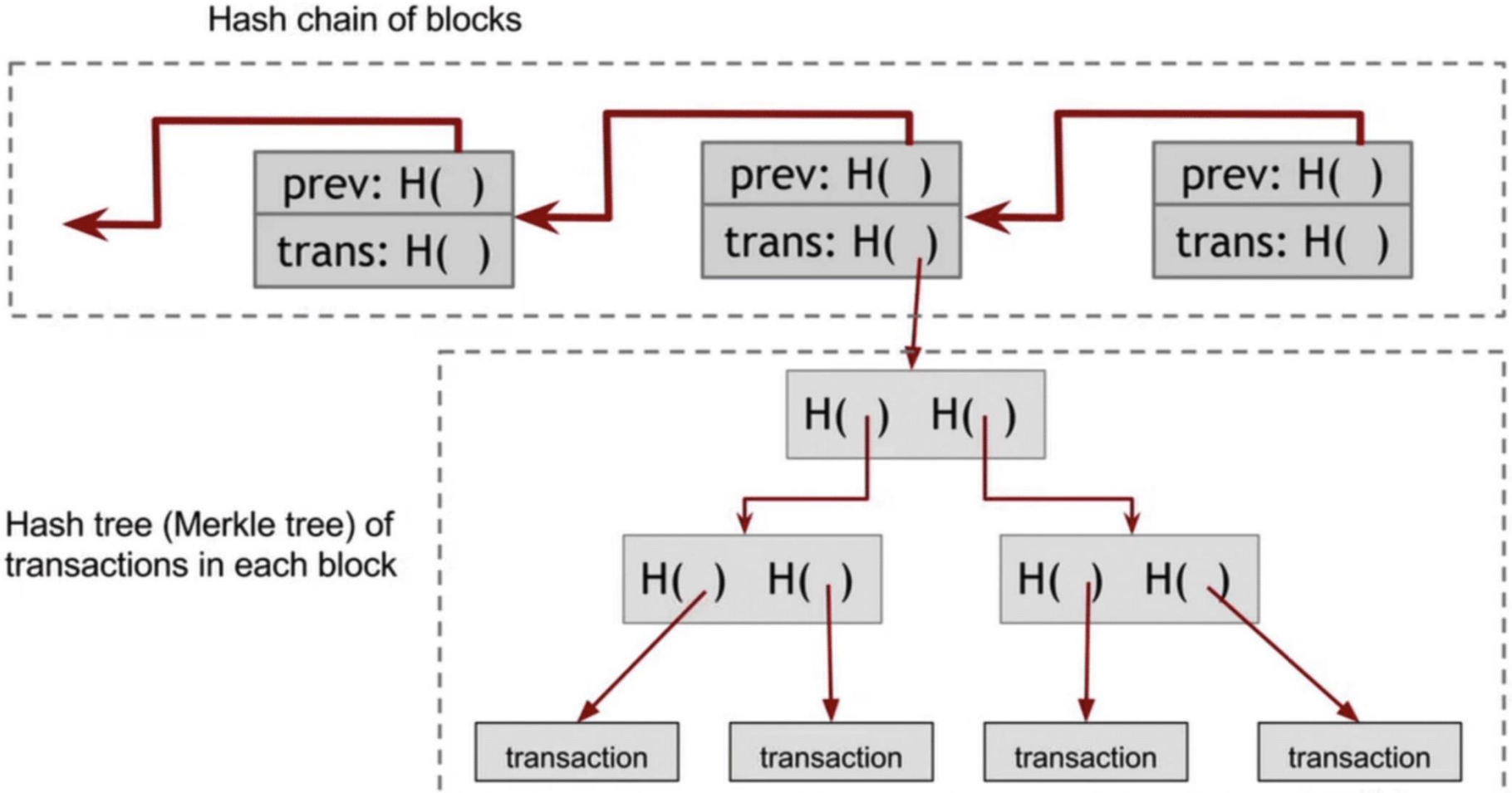
- The node thus needs to access the blockchain and may need a full local copy.
- This is a lot to do!! Is it possible to verify bitcoin payments without running a full network node?

Transaction verification

- What is needed to verify that a transaction has happened?
  - It just must be included in a block which must be in the longest chain (not orphan)
  - The block must have valid chain of previous blocks back to the genesis block

- How to do that on a regular node?

Merkle Trees in Bitcoin



Simplified Transaction Verification

- How merkle trees can help?
  - The size of block header is small as they only contain merkle root.
  - For a block header size of 80 bytes (like Bitcoin) it requires blockheight * 80 bytes. At height 530361 it requires 40 Megabytes.

Simplified Transaction Verification

- How merkle trees can help for verifying transaction inclusion in a block?
  - We do not need to have all the hashes.
  - The merkle proof will always consist of one hash for every step up the tree.
  - A block of 16 transactions would require 4 hashes
  - For 1000 transactions – 10 hashes. Average is more than 2500 transaction these days!

Do not distribute …

- These slides are not always prepared by me.
- Most of the content comes from the reference book
  - Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction
  - Arvind Narayanan, Joseph Bonneau,
  - Edward Felten, Andrew Miller, Steven Goldfeder