Building Software Systems

Lecture 6.2

Blockchain Fundamentals – Part 2

SAURABH SRIVASTAVA
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
IIT (ISM) DHANBAD

Recap …

In the last Module, we had a brief introduction to Blockchains, and enabling technologies

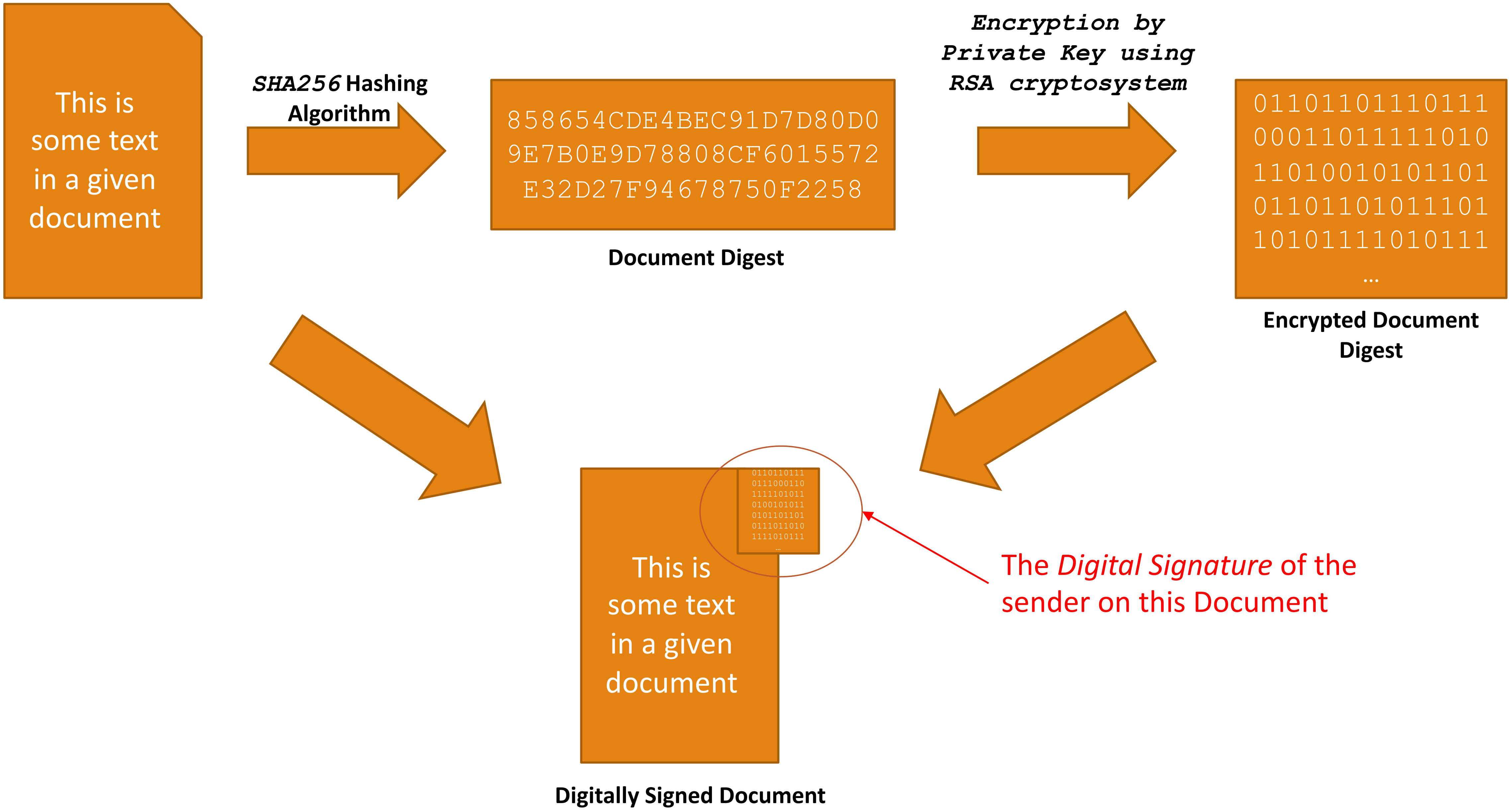
- We saw the NIST definition of a blockchain
- We also saw an example of how a blockchain provides a change-resistant storage option
- We talked, about some enabling technologies, such as hashes and digital signatures

Today, we will continue our introductory session on Blockchain

Digital Signature Certificates

We saw in the last module, how hashing and PKI can be used to generate a Digital Signature

- This signature could be attached with a document to verify that it was indeed, created by the author
- All that is required to be done, is verify the signature with the author’s public key



This is some text in a given document
SHA256 Hashing Algorithm
858654CDE4BEC91D7D80D0
9E7B0E9D78808CF6015572
E32D27F94678750F2258
Document Digest
Encryption by Private Key using RSA cryptosystem
0110110111011 1
0001101111101 0
1101001010110 1
0110110101110 1
1010111101011 1
…
Encrypted Document Digest
0110110111
0111000110
1111101011
0100101011
0101101101
0111011010
1111010111
…
This is some text in a given document
Digitally Signed Document
The *Digital Signature* of the sender on this Document

Digital Signature Certificates

We saw in the last module, how hashing and PKI can be used to generate a Digital Signature

- This signature could be attached with a document to verify that it was indeed, created by the author
- All that is required to be done, is verify the signature with the author's public key

This brings us to the next question – how does the "world" get the public key of an author?

- Clearly, not everyone can store the public keys of everyone else in the world !!
- Instead, modern software applications rely on some "central agencies" to get these keys as required

A Certificate Authority (CA) issues *Digital Signature* ***Certificates*** to individuals or organisations

- This certificate contains the information about the certificate owner, along with the required public key
- Software applications, e.g., web browsers, talk to these CAs to establish the identities of web domains

On a private network, these certificates may be issued by a *Certification Authority*

- This, in most cases, is a trusted server in the network, which has a mechanism to identify new entrants …
- … and issue them these certificates, so that they can communicate on the network with their peers

Basic Blockchain Terminology

We have now covered the basic technologies involved in a Blockchain deployment

- We are now in a position to look at how Blockchains work in the real world
- Wherever required, we will take examples of a particular public blockchain – the *Bitcoin* network

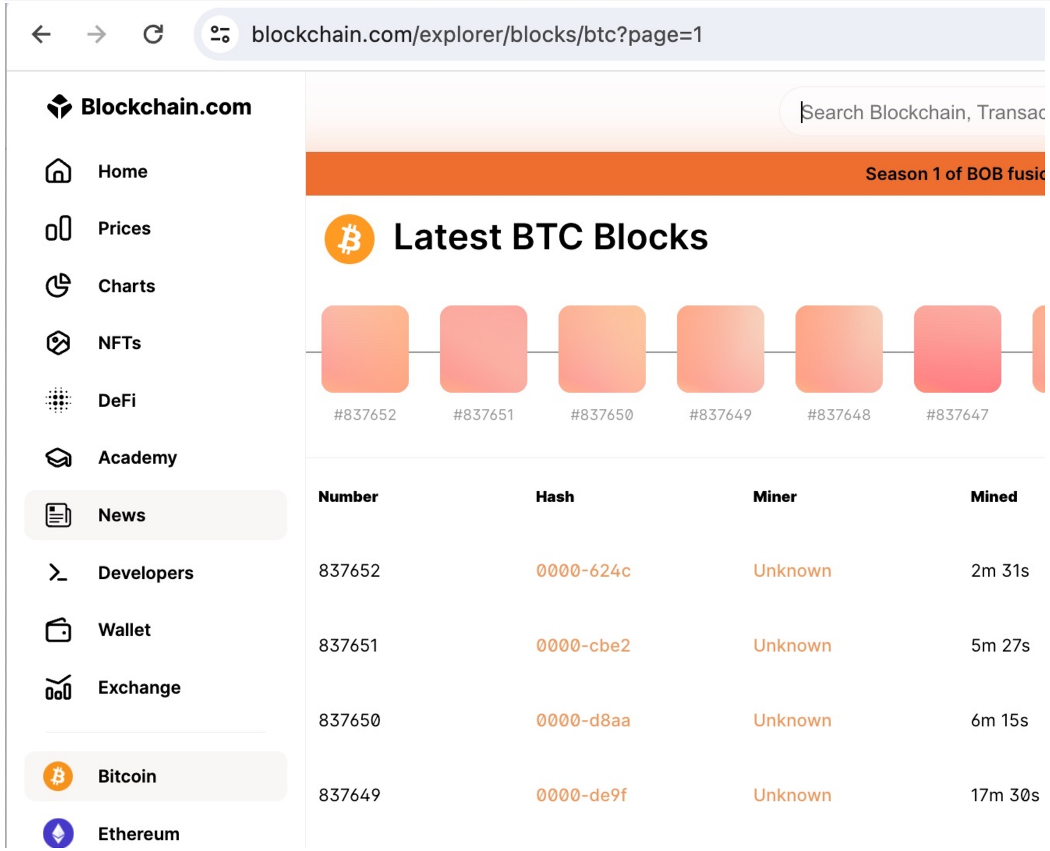
In every blockchain, we have

- A number of *nodes* – machines that perform computations, and store data on a blockchain
- A number of *participants* – people or organisations which are using the blockchain to achieve some goals
- While a node may be associated with a participant, a participant may not add a node to the network

A blockchain may be *permissioned* or *open/permissionless*

- In a permissionless network, any new participant can enter and participate, without any permissions
- The Bitcoin network, is an example of such a network – one can start using Bitcoin any time without approvals
- In a permissioned blockchain, only selected, pre-approved participants can change the blockchain
- A permissioned blockchain is usually also *private* in nature, i.e., its data is not available for public consumption
- In contrast, permissionless blockchains are mostly *public* in nature, giving full transparency to its data

Example of a public Blockchain – Bitcoin



You can always check out the recently added “blocks” of Bitcoin through online sources such as blockchain.com blockchair.com/bitcoin/blocks

BLOCKCHAIR | Search for transactions, addresses, blocks and embedded text data... | Explorers | Features | ENG · USD

Explorers • Bitcoin

Bitcoin blocks

API

Similar data from blockchair.com !!

Table fields 10 / 36

- [x] Height
- [x] Hash
- [x] Time
- [x] Guessed miner
- [ ] Coinbase data
  - [ ] Bin
  - [ ] Hex
- [x] Transaction count
- [ ] Witness tx count
- [ ] Input count
- [ ] Output count
- [ ] Coindays destroyed
- [ ] Input
  - [ ] BTC
  - [ ] USD
- [x] Output
  - [x] BTC
  - [x] USD

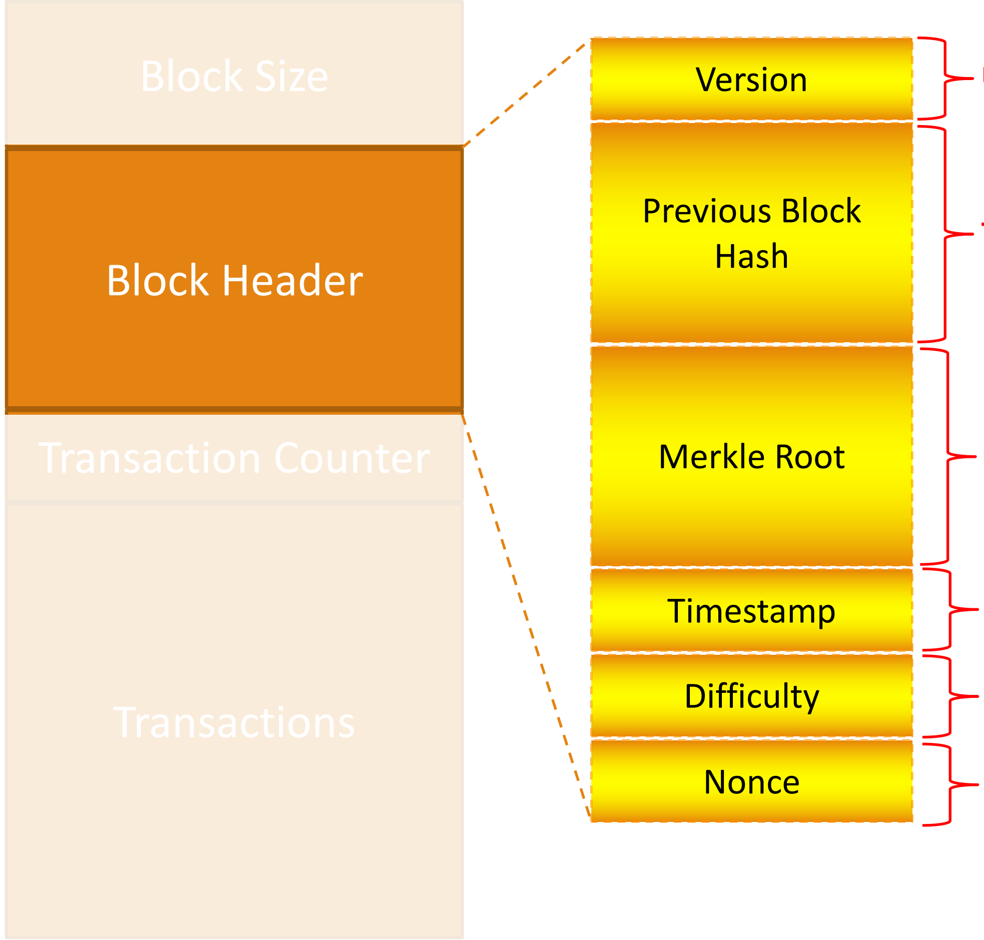
Height	Hash	Time (UTC)	Guessed miner	Transaction count	Output (BTC)
837,651	00000•••4cbe2	2024-04-04 07:09:26	Binance	1025	65.02 BTC
837,650	00000•••ed8aa	2024-04-04 07:08:38	AntPool	3063	4,075.60 BTC
837,649	00000•••1de9f	2024-04-04 06:57:23	SlushPool	2309	4,044.35 BTC
837,648	00000•••347bf	2024-04-04 06:49:30	Unknown	3205	9,336.88 BTC
837,647	00000•••0cd2b	2024-04-04 06:23:55	ViaBTC	3457	1,052.28 BTC
837,646	00000•••d9848	2024-04-04 06:21:16	AntPool	3255	8,078.76 BTC
837,645	00000•••9103f	2024-04-04 05:55:00	Foundry USA Pool	2886	4,696.03 BTC
837,644	00000•••76782	2024-04-04 05:44:07	F2Pool	544	865.65 BTC
837,643	00000•••2827c	2024-04-04 05:39:06	Unknown	2612	6,195.78 BTC
837,642	00000•••74e3c	2024-04-04 05:25:55	AntPool	1300	1,276.13 BTC

Typical Structure of a Blockchain Block

Block Size	**The total size of the block excluding this field in bytes (4 bytes)**
Block Header	**A wide range of metadata related to this block (80 bytes)**
Transaction Counter	**Number of Transactions in this block (1-9 bytes)**
Transactions	**Transactions data (*variable*)**

Source:
Mastering Bitcoin: Unlocking Digital Cryptocurrencies, O'Reilly Media, Inc.
[**Andreas M. Antonopoulos**]

Typical Structure of a Blockchain Block



Version – Useful if there are frequent changes to the agreed protocols

Previous Block Hash – The hash of the previous block's header

Merkle Root – A hash of the merkle tree's root node (more on this shortly)

Timestamp – Approximate creation time of the block (counted as the elapsed seconds since Unix Epoch)

Difficulty – The difficulty target chosen for the proof-of-work (we will discuss this a little later in the lecture)

Nonce – A counter, that is incremented, till the hash of this header data does not match the expectations for the proof-of-work (we'll get back to this again)

Source:
Mastering Bitcoin: Unlocking Digital Cryptocurrencies, O'Reilly Media, Inc.
[**Andreas M. Antonopoulos**]

Transactions in a blockchain

Transactions mean a unit of data to be stored over the blockchain

- An example of a transaction may look like , "*Alice paid $5 to Bob*"
- A block of a blockchain may have thousands or millions of transactions
- In a decentralised system, the trust over a transaction's legitimacy is provided through PKI
- For example, the above transaction will require verification of digital signature of Alice …
- … as well as a verification that the transaction was *signed* by the Private Key of Alice
- Remember, while only Alice has her Private Key, her Public Key is available as part of the certificate …
- … certified by a trusted authority

Transactions doesn't mean financial transactions only

- It can, for instance, store land records or marks/grades of students
- Essentially, any piece of data that needs to be stored on a blockchain, needs to be represented as a transaction

Merkle Trees

Merkle Trees are a data structure designed to create a *summary* of a large number of *transactions*

- Here, transactions mean a unit of data to be stored over the blockchain, e.g., “*Alice paid $5 to Bob*”
- A block of a blockchain may have thousands or millions of transactions
- A Merkel tree is a tree which uses a hash-based method to create cumulative summaries …
- … that are created over all the transactions in a block

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

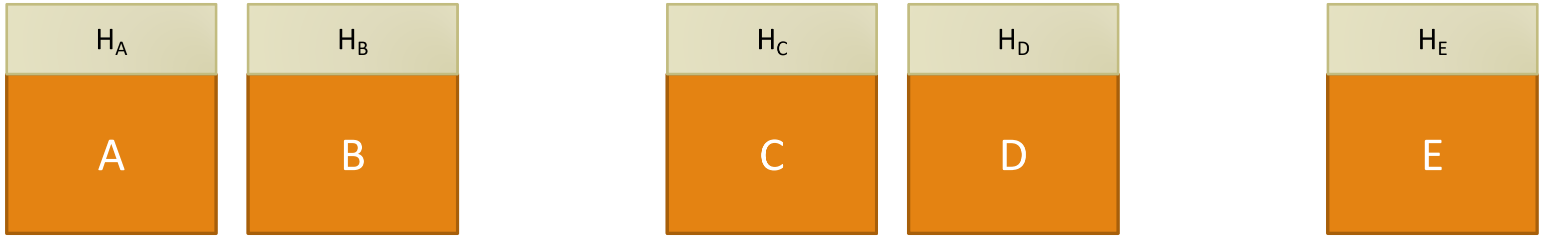
Assume that we have the above five transactions, that are part of a block, "in that order"

Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

We calculate hashes (which are often the application of SHA256 algorithm, twice) for each transactions first



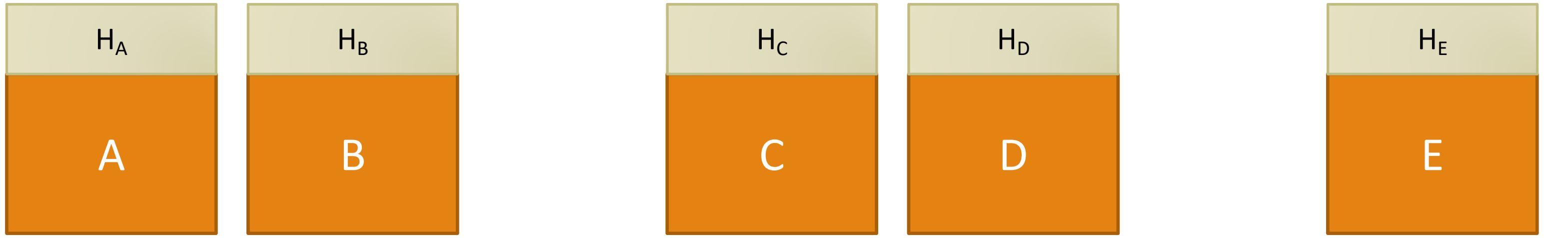
Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

We calculate hashes (which are often the application of SHA256 algorithm, twice) for each transactions first

For our discussion, we define the function *H* as:
`H(X) = SHA256(SHA256(X))`
The output is 32 bytes long, irrespective of the length of `X`

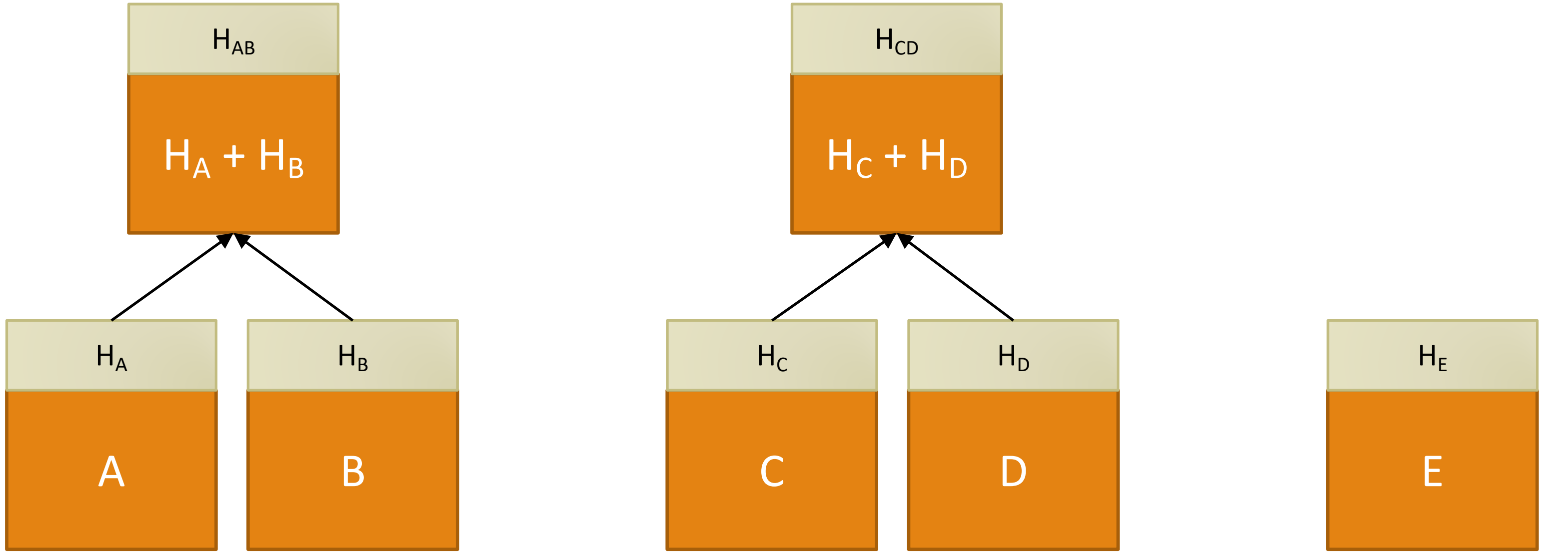


Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

We then calculate the hashes for the next level of the tree, by combining the hashes of two consecutive transactions, and rehashing it !!



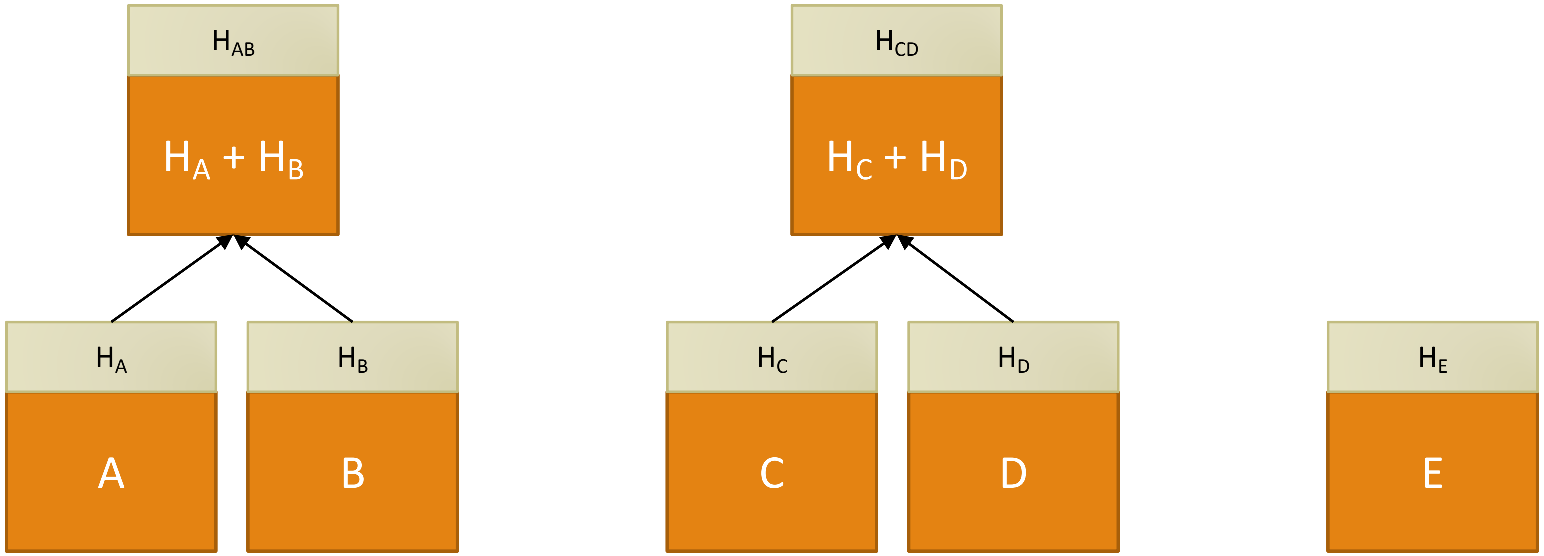
Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

We then calculate the hashes for the next level of the tree, by combining the hashes of two consecutive transactions, and rehashing it !!

Here,
`X + Y = concatenate(H, Y)`
The output is the concatenated string containing `X` followed by `Y`

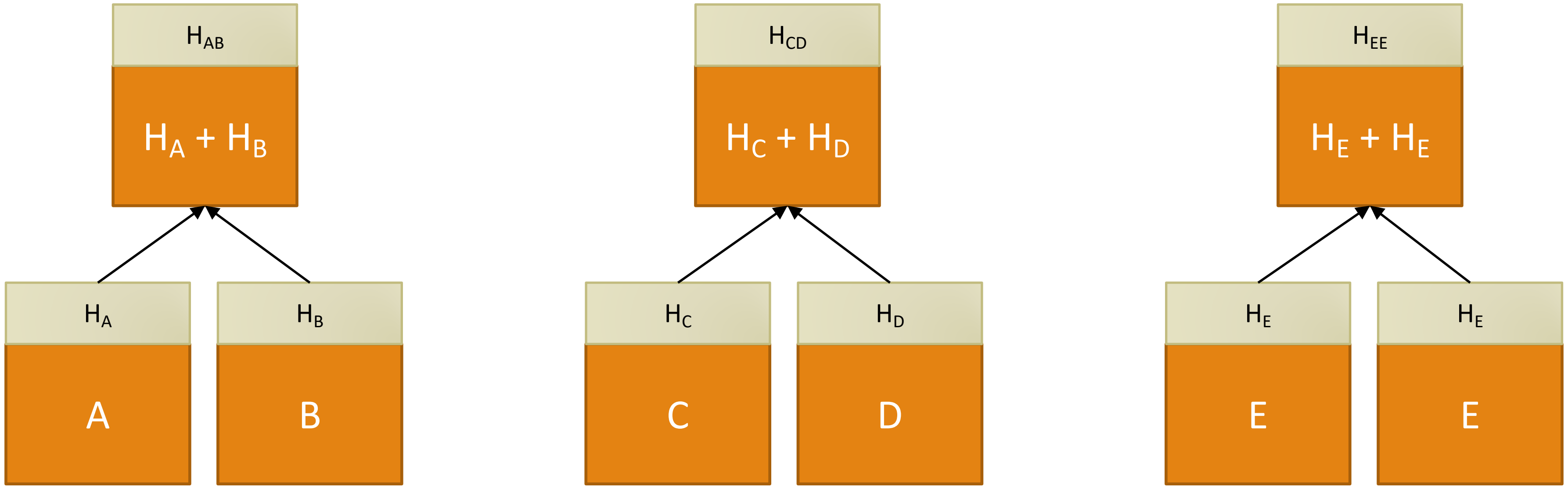


Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

For any transactions that could not be paired, we can create a dummy node in the tree, repeating the transaction

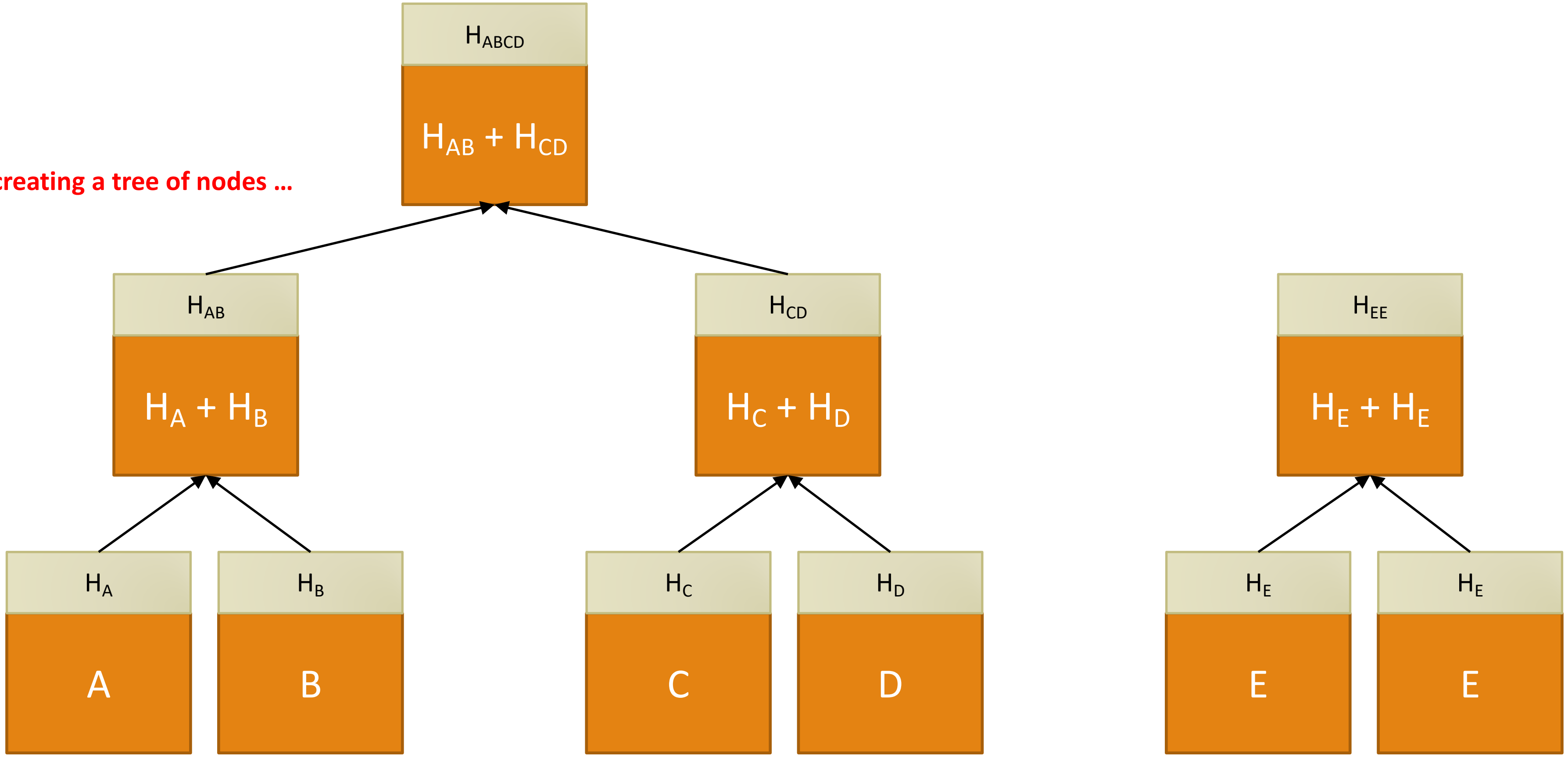


Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

The process continues upwards, creating a tree of nodes …

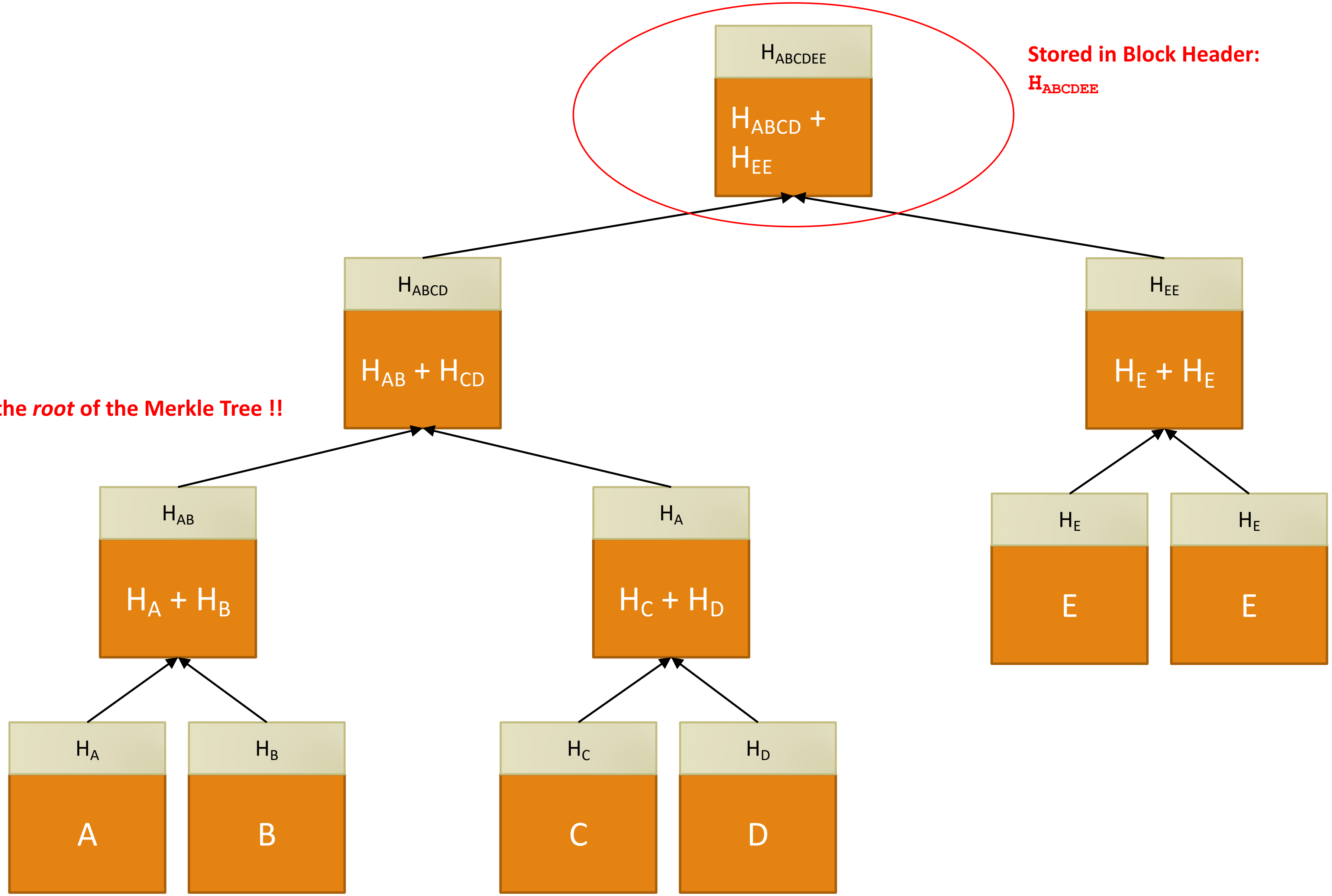


Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

… until we have just one node – the *root* of the Merkle Tree !!



Stored in Block Header:
H_{ABCDEE}

Using Merkel Trees to create transaction summaries

Merkle Trees

Merkle Trees are a data structure designed to create a *summary* of a large number of *transactions*

- Here, transactions mean a unit of data to be stored over the blockchain, e.g., "*Alice paid $5 to Bob*"
- A block of a blockchain may have thousands or millions of transactions
- A Merkel tree is a tree which uses a hash-based method to create cumulative summaries …
- … that are created over all the transactions in a block

The root node of the Merkle tree is dependent on each node in the tree

- Thus, if even one transaction in the block is tempered with, the hash of the root changes, which can be detected

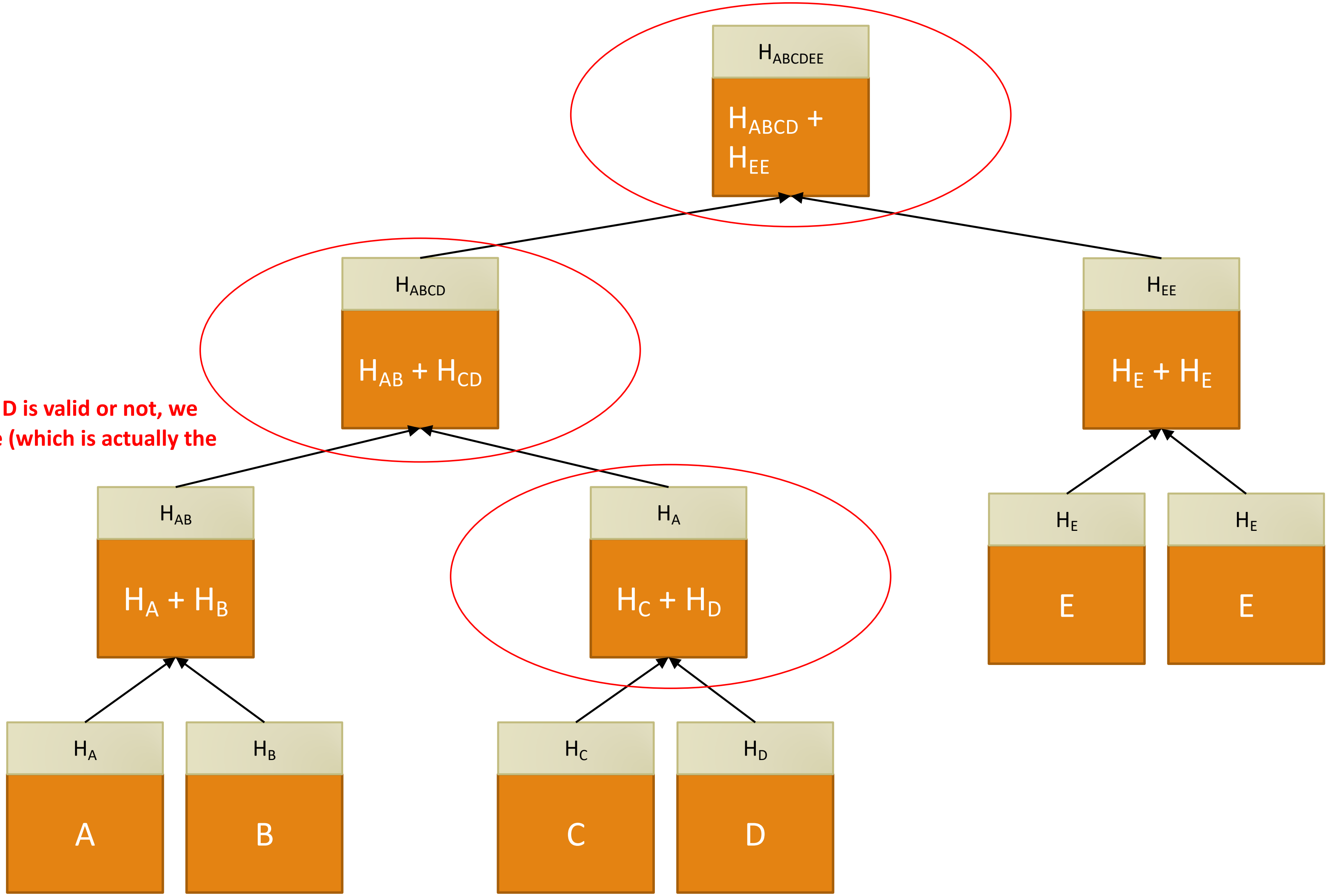
Merkle Trees are useful for non-full nodes in a blockchain

- A full-node in a blockchain, is a node that contains every bit of information for every block of the blockchain
- This includes the whole chain from the *genesis block* (the first block in the blockchain), till the last added block
- For each block, it stores the complete Transaction Details as well as the complete Merkel Tree
- If a non-full node wishes to check a transaction's validity, it can be done by contacting some full node …
- … and "only" retrieving some hashes, which can then be verified locally !!

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*

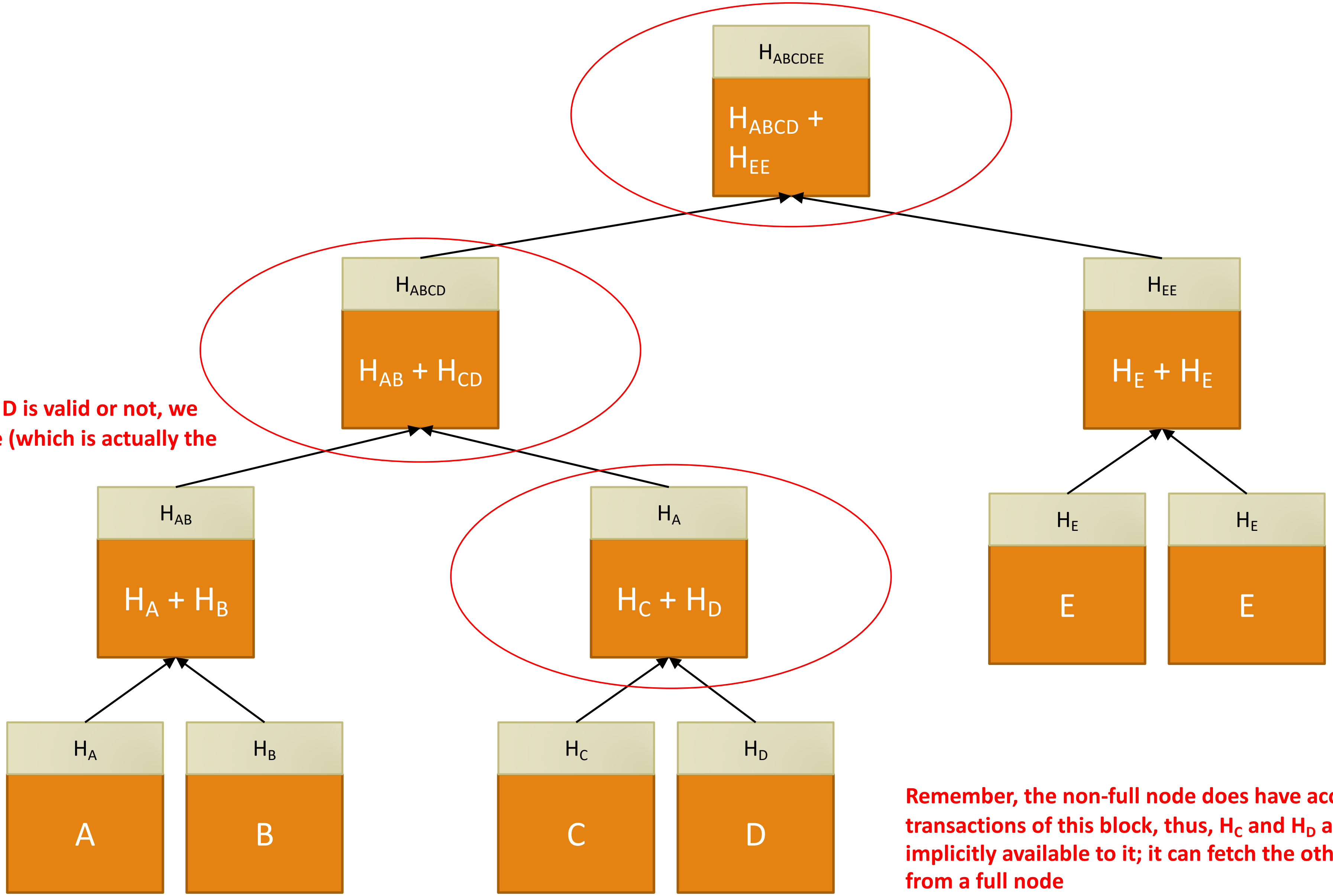
If we wish to check if transaction D is valid or not, we need only the hashes circled here (which is actually the path from D to the root node)



Using Merkel Trees to create transaction summaries

List of Transactions in the block:

1. *A*
2. *B*
3. *C*
4. *D*
5. *E*



If we wish to check if transaction D is valid or not, we need only the hashes circled here (which is actually the path from D to the root node)

Remember, the non-full node does have access to all transactions of this block, thus, H_C and H_D are implicitly available to it; it can fetch the other hashes from a full node

Using Merkel Trees to create transaction summaries