Hashing: Block Chain

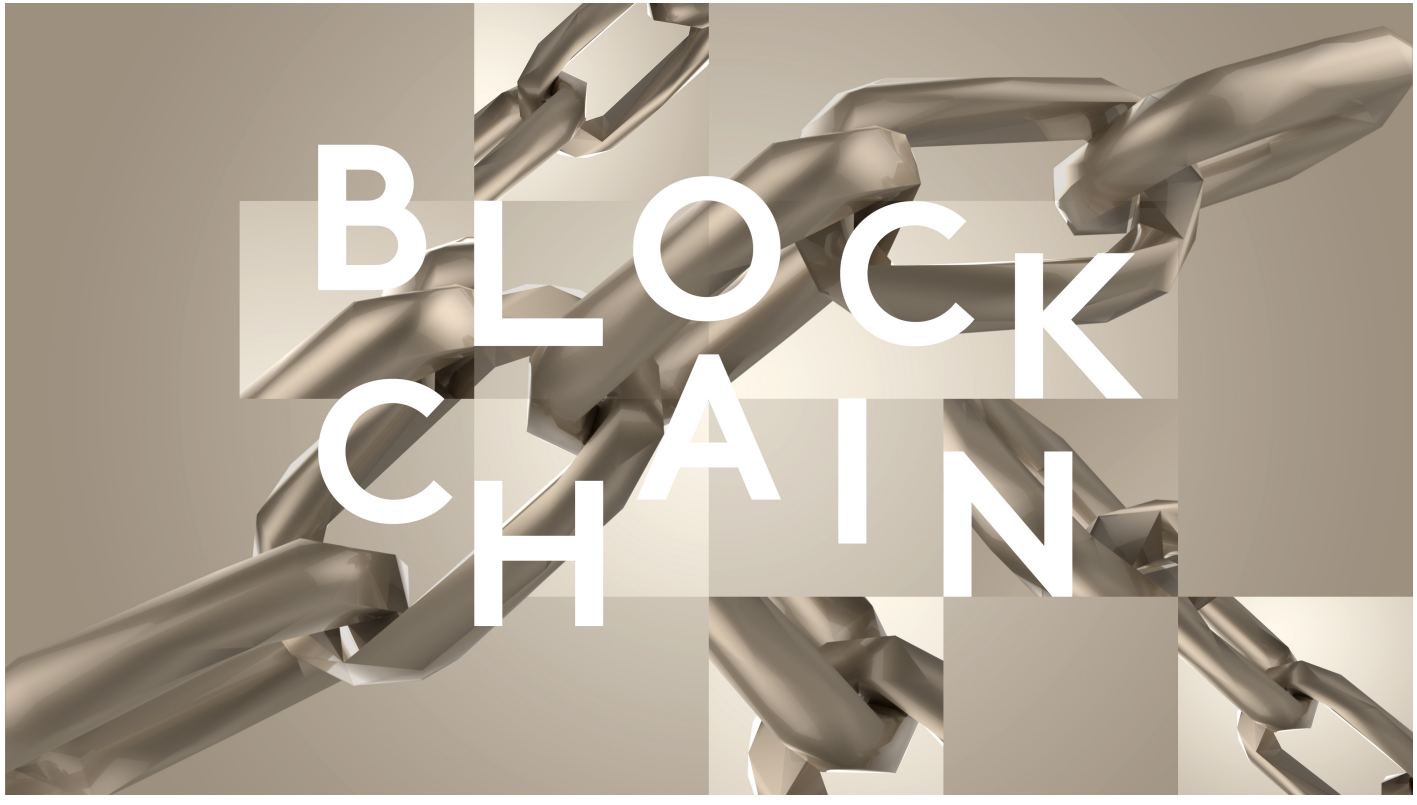
Michael Levin

Department of Computer Science and Engineering
University of California, San Diego

Data Structures Fundamentals
Algorithms and Data Structures

Outline

Blockchain



Julia's Diary

Julia started to keep a diary like this:

1	Had breakfast
2	Went for a walk
...	...
239	Lent $100 to Daniel
240	Watched "House of Cards"

Julia's Diary

Julia tried to keep the diary diligently and write down everything she did. If she had an argument with anybody about some of her prior actions, she took out the diary and showed the corresponding record.

Julia's Diary

At some point, Julia had an argument with Daniel regarding whether she lent him $100 or not. She didn't have her diary with her, but promised to bring it tomorrow and show it to him.

Julia's Diary

In the evening, while she was away, Daniel broke into Julia's room and forged the diary:

...	...
239	Watched "Game of Thrones"
240	Watched "House of Cards"
...	...

Julia's Diary

Julia checked the diary next day, didn't find any record regarding this and apologized.

Julia's Diary

10 years later, Daniel confessed everything to Julia out of remorse. Julia forgave him but decided to enhance her diary so that it would be harder to forge it.

Julia's Diary

Julia learned that it is possible to use hash functions to make short digests of arbitrary strings. She also learned that for some hash functions (like MD5) it is extremely hard to find another string with exactly the same hash value.

Julia's Diary

She decided to add a hash value column $H[i]$ to each record in the diary:

$$H[i+1] = h(h(Text[i+1]) \circ H[i])$$

Julia's Diary

She decided to add a hash value column $H[i]$ to each record in the diary:

$$H[i+1] = h(h(Text[i+1]) \circ H[i])$$

Julia's Diary

She decided to add a hash value column $H[i]$ to each record in the diary:

$$H[i+1] = h(h(Text[i+1]) \circ H[i])$$

Julia's Diary

0		0000
1	Had breakfast	2308
2	Went for a walk	204
...	...	...

$h(\text{“Had breakfast”}) = 364,$
$h(\text{“3640000”}) = 2308$
$h(\text{“Went for a walk”}) = 1782,$
$h(\text{“17822308”}) = 204$

Julia's Diary

This way, to forge any particular record in the diary, one would have to also forge all the records after it to fix the hash values. This is already harder and is good enough for lending small amounts of money to friends.

Learn in the next video what happened when Julia decided to open a bank and keep record of all the transactions.

Outline

Julia's Bank

Seeing the success of her diary in keeping everything recorded, Julia decided to open a bank. The bank gave credits to lenders, and Julia used her diary to keep track of outstanding debts.

Julia's Bank

Julia's bank grew fast and started giving credits for large amounts of money. One night, a desperate lender hired a squad, broke into Julia's room and forged the record about his debt and all the following records. This way, he didn't have to repay his debt.

Julia's Bank

Julia remembered very vividly giving this credit. However, according to her own rules, she couldn't demand it back. To make forgery even harder and economically non-viable, she came up with a new idea.

Julia's Bank

She decided to add a number (called nonce) in brackets to the end of each record in such a way that the hash column value always ends with three zeros (is divisible by 1000).

Julia's Bank

0		0000
1	Had breakfast(263)	2000
2	Went for a walk(352)	7000
...	...	...

$h(\text{“Had breakfast(263)”}) = 125,$

$h(\text{“1250000”}) = 2000$

$h(\text{“Went for a walk(352)”}) = 9876,$

$h(\text{“98762000”}) = 7000$

Julia's Bank

- To come up with the right nonce, Julia has to try them all, one by one

Julia's Bank

- To come up with the right nonce, Julia has to try them all, one by one
- The remainder of the resulting hash value modulo 1000 will be any number between 0 and 999 with approximately the same probability

Julia's Bank

- To come up with the right nonce, Julia has to try them all, one by one
- The remainder of the resulting hash value modulo 1000 will be any number between 0 and 999 with approximately the same probability
- Thus, it would take her on average around 500 steps until she finds a nonce resulting in remainder 0 modulo 1000

Julia's Bank

- It is unknown how to find a string with a given hash value faster than by trying all possible strings

Julia's Bank

- It is unknown how to find a string with a given hash value faster than by trying all possible strings
- Thus anyone who wants to forge a record would need to find a nonce in 500 steps for every record after it

Julia's Bank

- It is unknown how to find a string with a given hash value faster than by trying all possible strings
- Thus anyone who wants to forge a record would need to find a nonce in 500 steps for every record after it
- All this makes forgery 500 times harder

Julia's Bank

- It is unknown how to find a string with a given hash value faster than by trying all possible strings
- Thus anyone who wants to forge a record would need to find a nonce in 500 steps for every record after it
- All this makes forgery 500 times harder
- Julia can control the tradeoff between her effort to keep the diary and the effort needed to forge it

Outline

Checking Lender's Transactions

- Prove to lenders their transactions are recorded
- Inefficient to show them the diary on each request
- Insecure to show them a whole diary page with other people's transactions

Solution: Hash Chain

- Instead of showing the diary itself, show just the hashes of records and the hash value column

Solution: Hash Chain

- Instead of showing the diary itself, show just the hashes of records and the hash value column
- Mark the record which contains the lender's transaction

Solution: Hash Chain

- Instead of showing the diary itself, show just the hashes of records and the hash value column
- Mark the record which contains the lender's transaction
- The lender can see his transaction's hash is correct, and verify $H[i+1] = h(h(T[i]) \circ H[i])$ for all i

Block Chain

- To avoid sending each lender hashes of the whole diary to check just one transaction, we can separate it into blocks of transactions (corresponding to pages in a real physical diary)

Block Chain

- To avoid sending each lender hashes of the whole diary to check just one transaction, we can separate it into blocks of transactions (corresponding to pages in a real physical diary)
- This represents our diary as a chain of blocks, or a Block Chain — thus the name

Block Chain

Block Chain in essense is just a diary which is very hard to forge. Typically, it is distributed and allows anybody to generate blocks which are verified and then added into the chain.

Outline

Merkle Tree

- To check that a transaction is recorded in a block, we saw that we need to send the whole chain of transactions inside the block and compute the whole chain of hashes

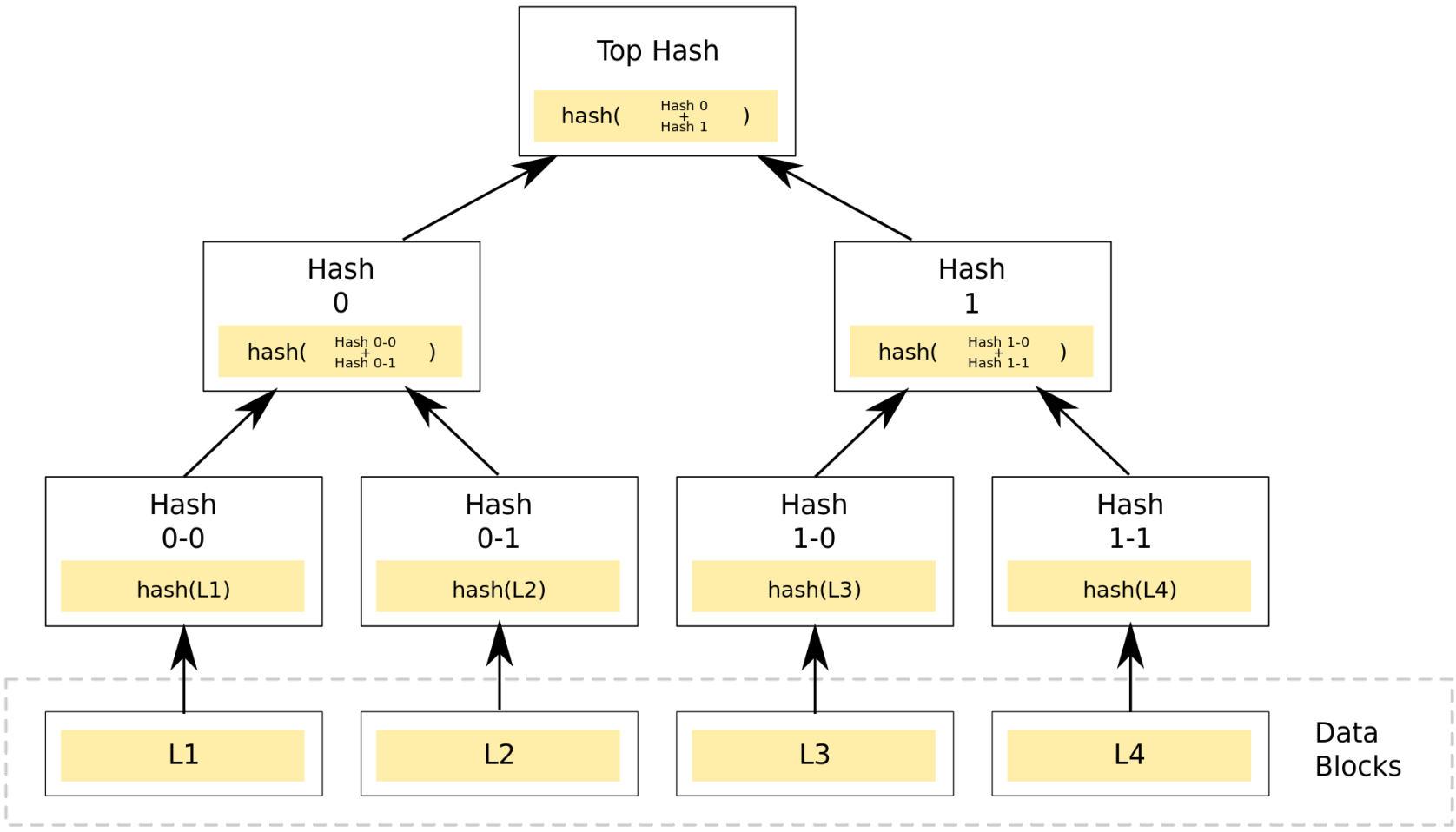
Merkle Tree

- To check that a transaction is recorded in a block, we saw that we need to send the whole chain of transactions inside the block and compute the whole chain of hashes
- Thus, it requires $O(n)$ time for n transactions in a block

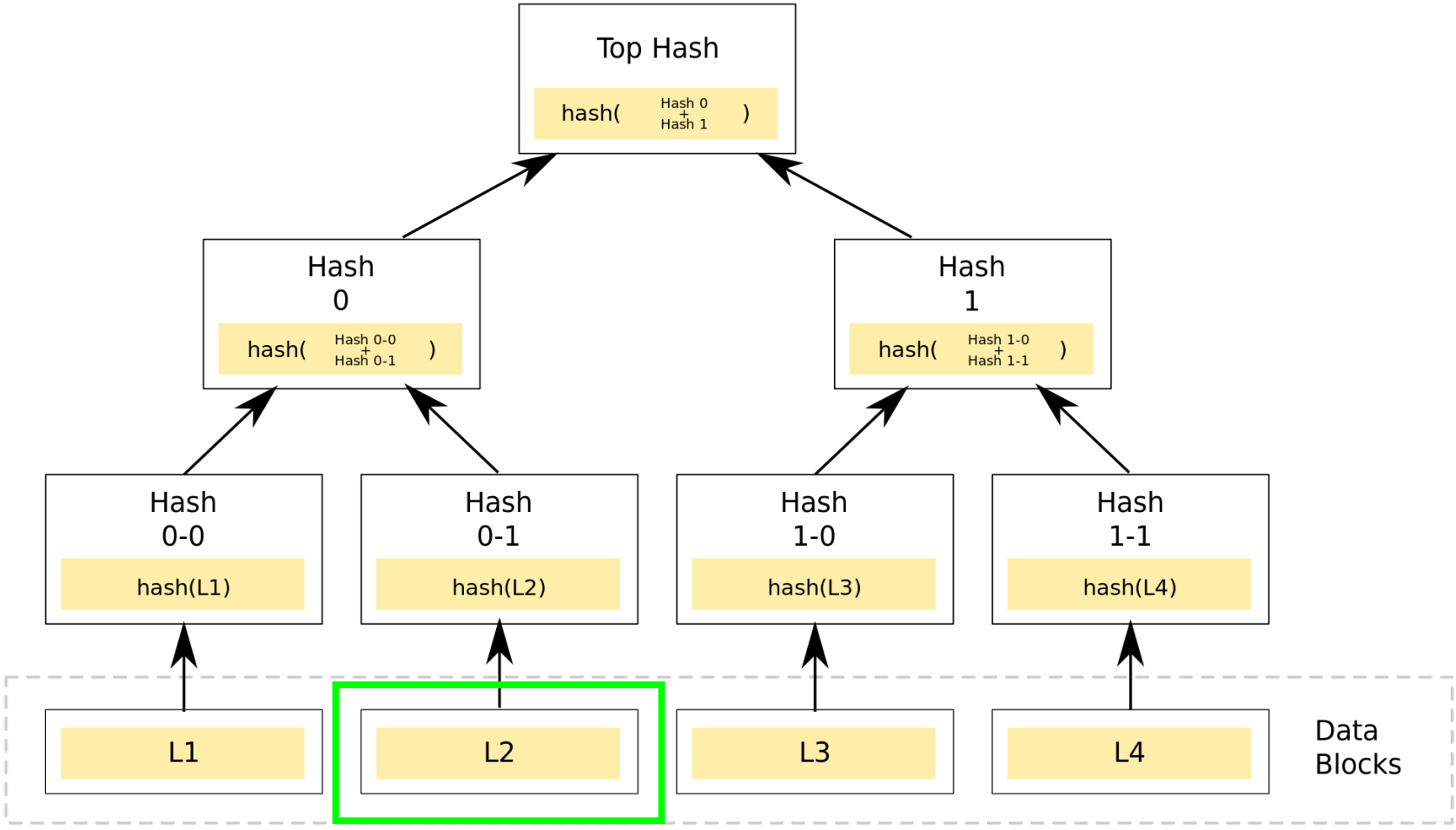
Merkle Tree

- To check that a transaction is recorded in a block, we saw that we need to send the whole chain of transactions inside the block and compute the whole chain of hashes
- Thus, it requires $O(n)$ time for n transactions in a block
- Merkle Tree allows to reduce this time to $O(\log n)$

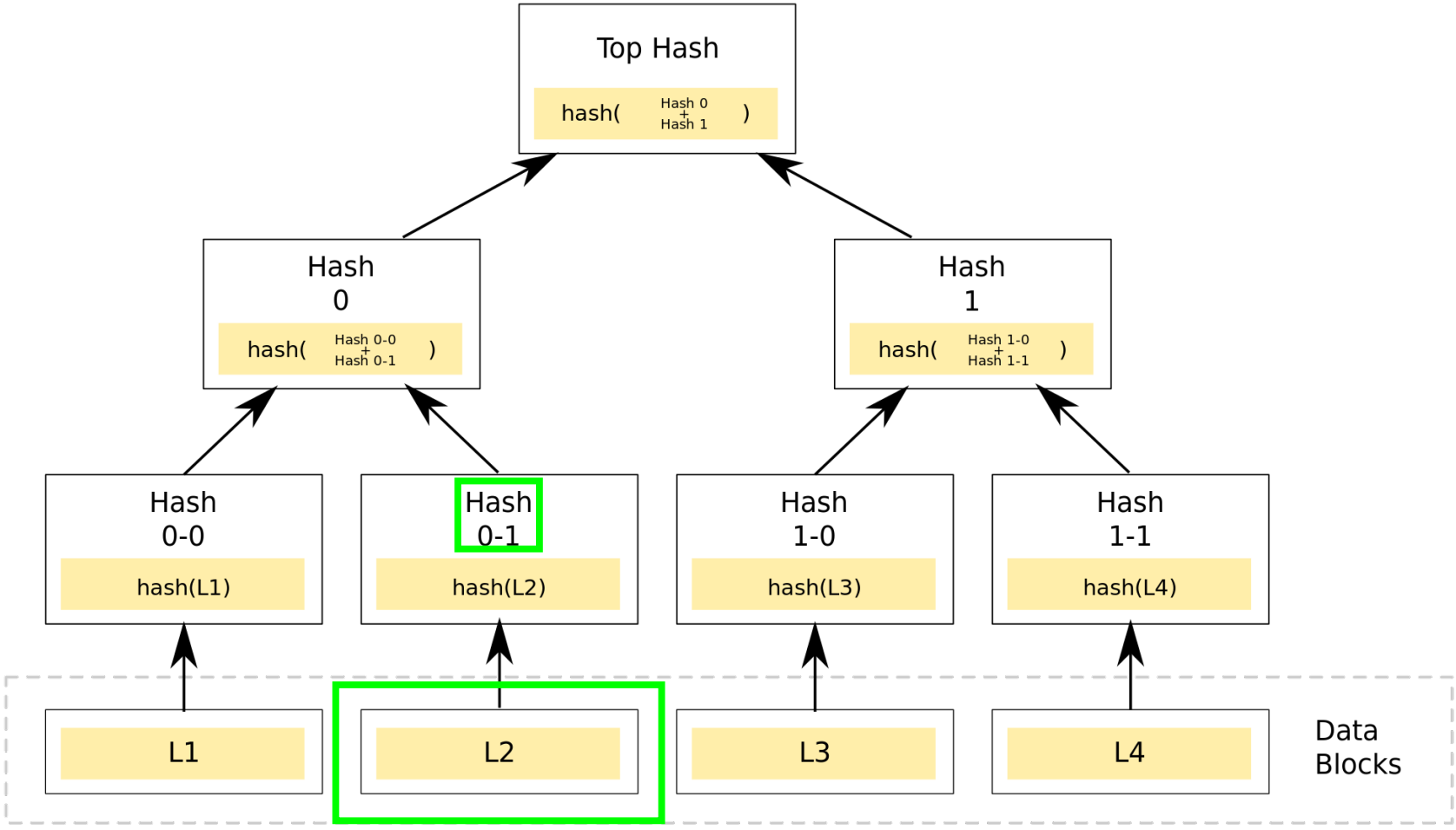
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

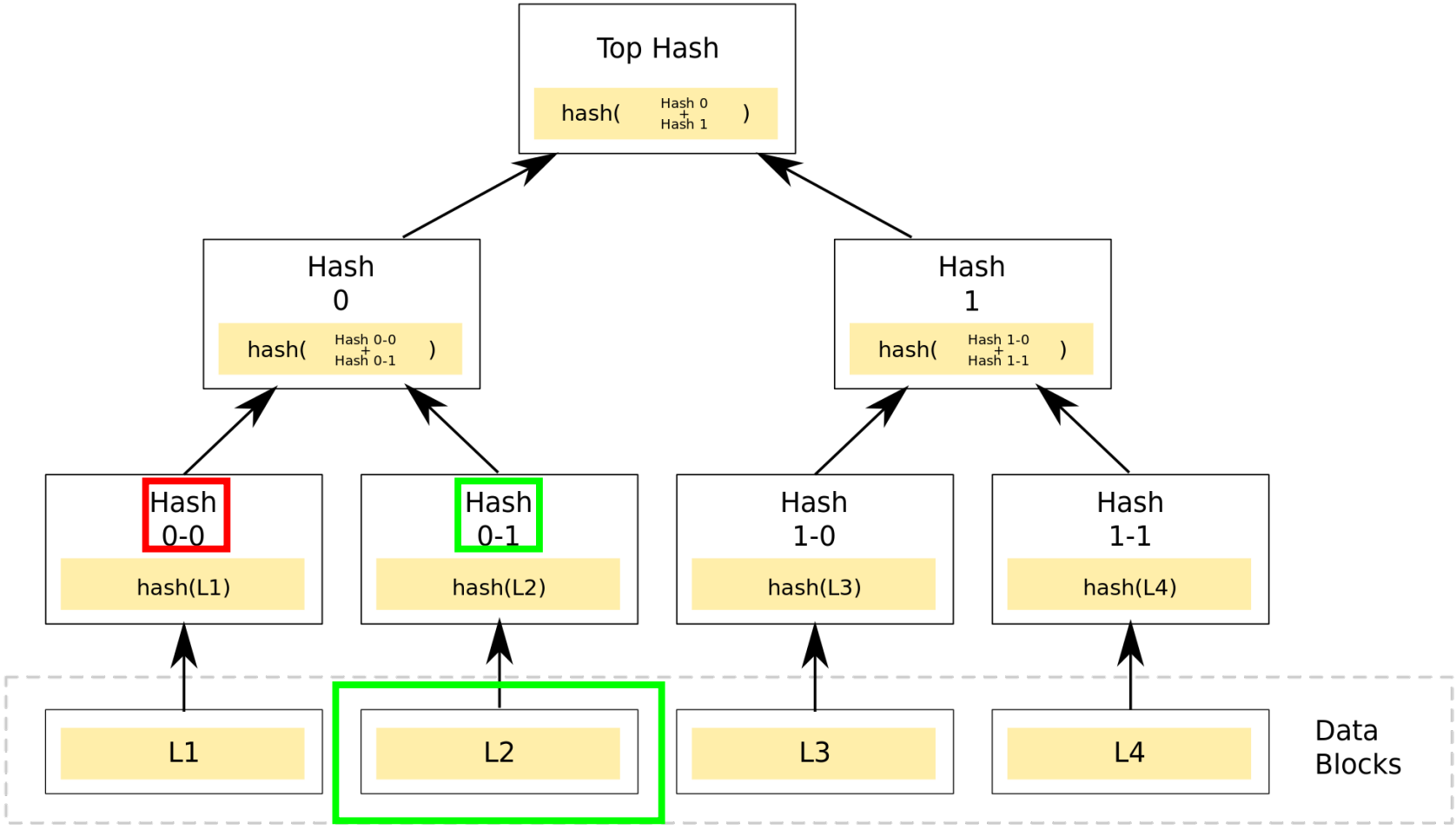
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

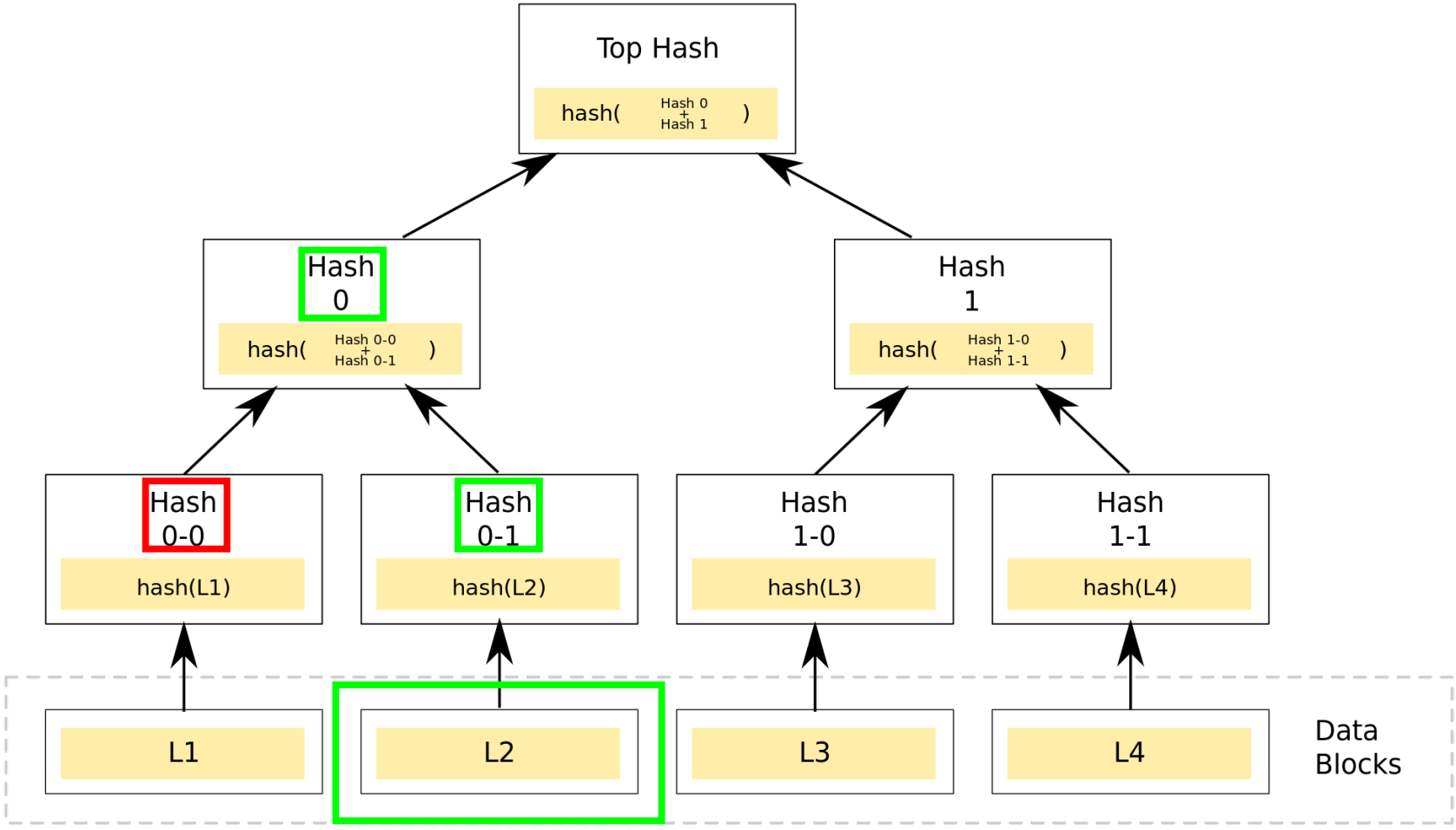
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

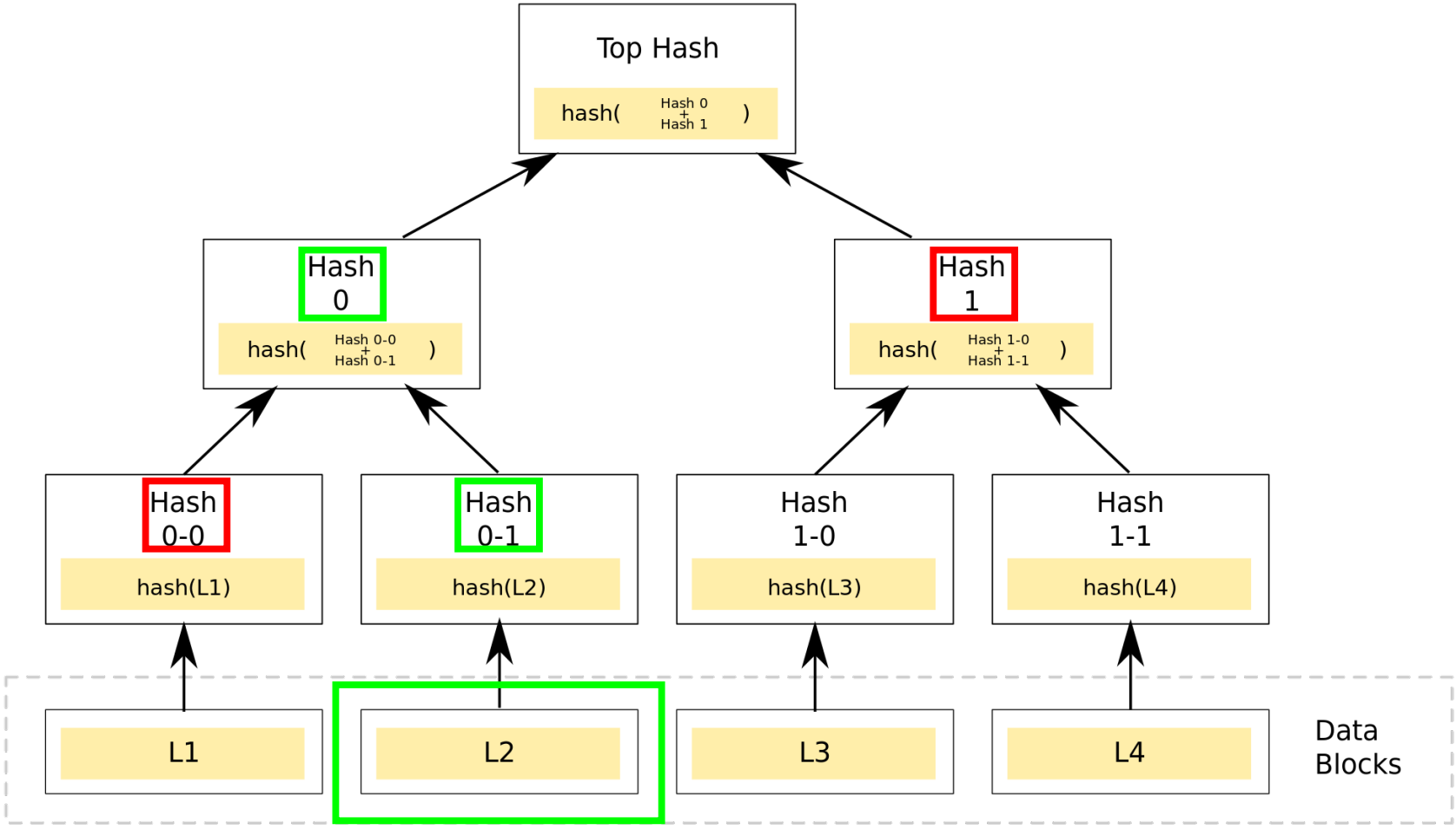
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

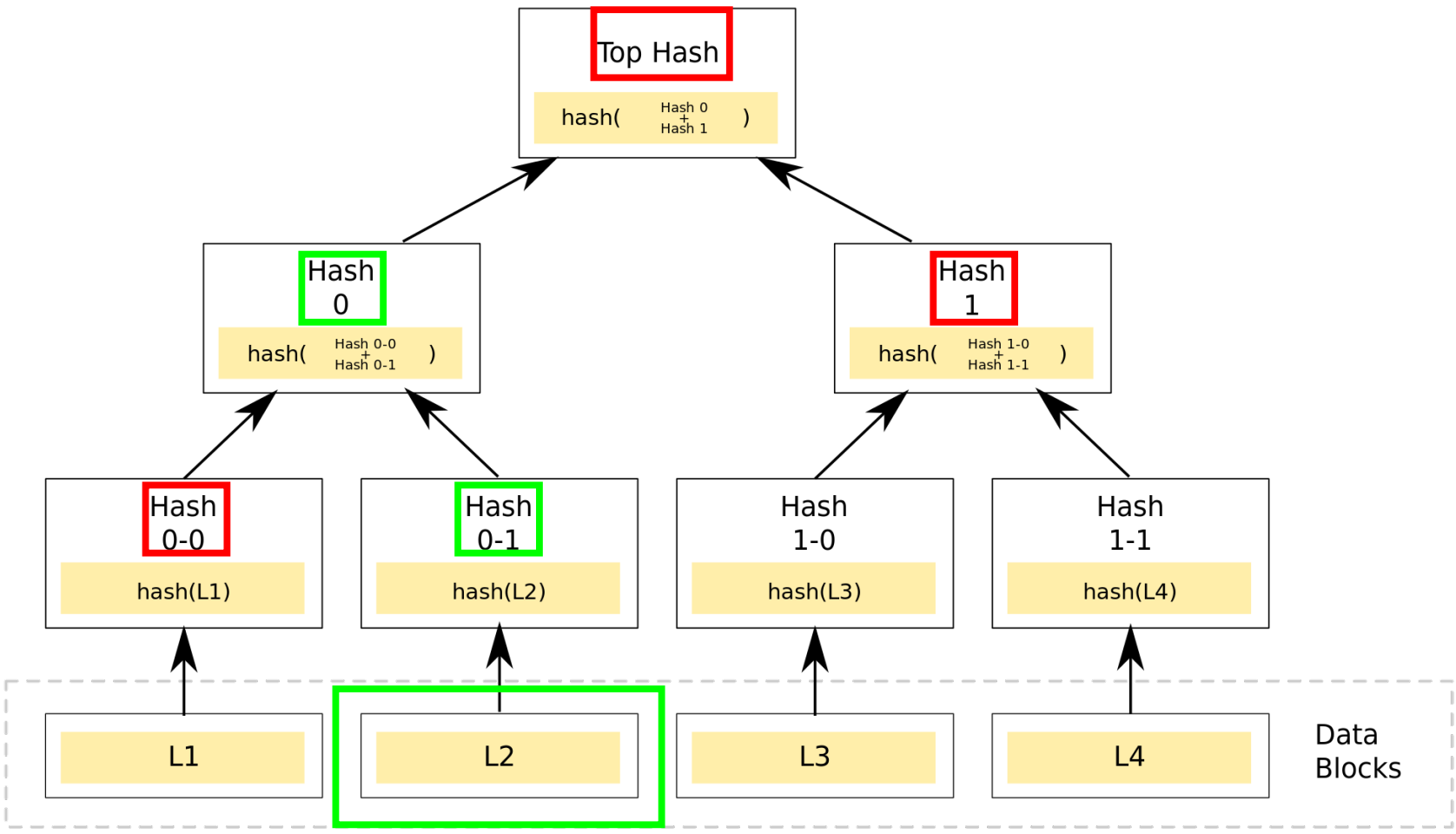
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

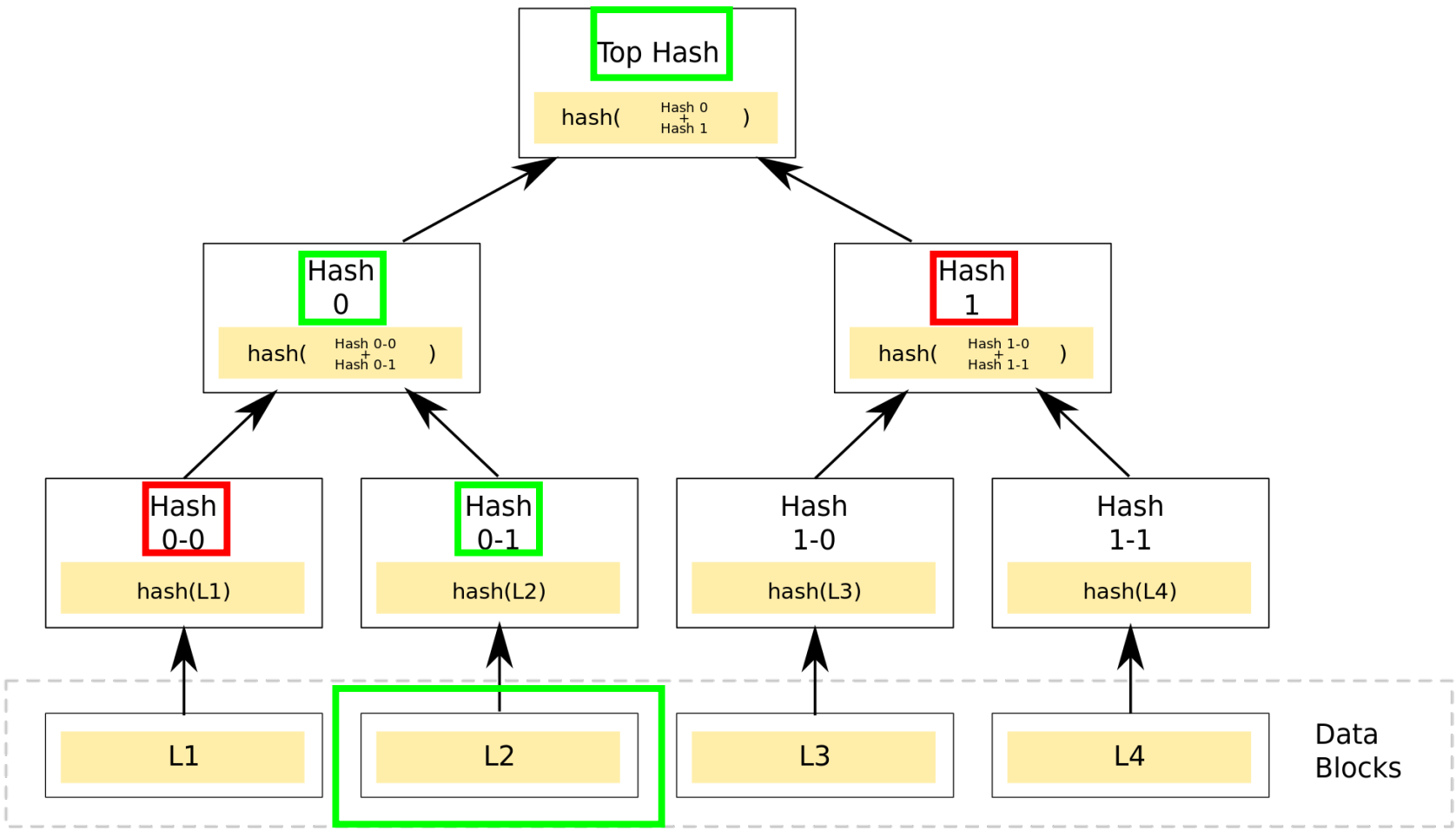
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

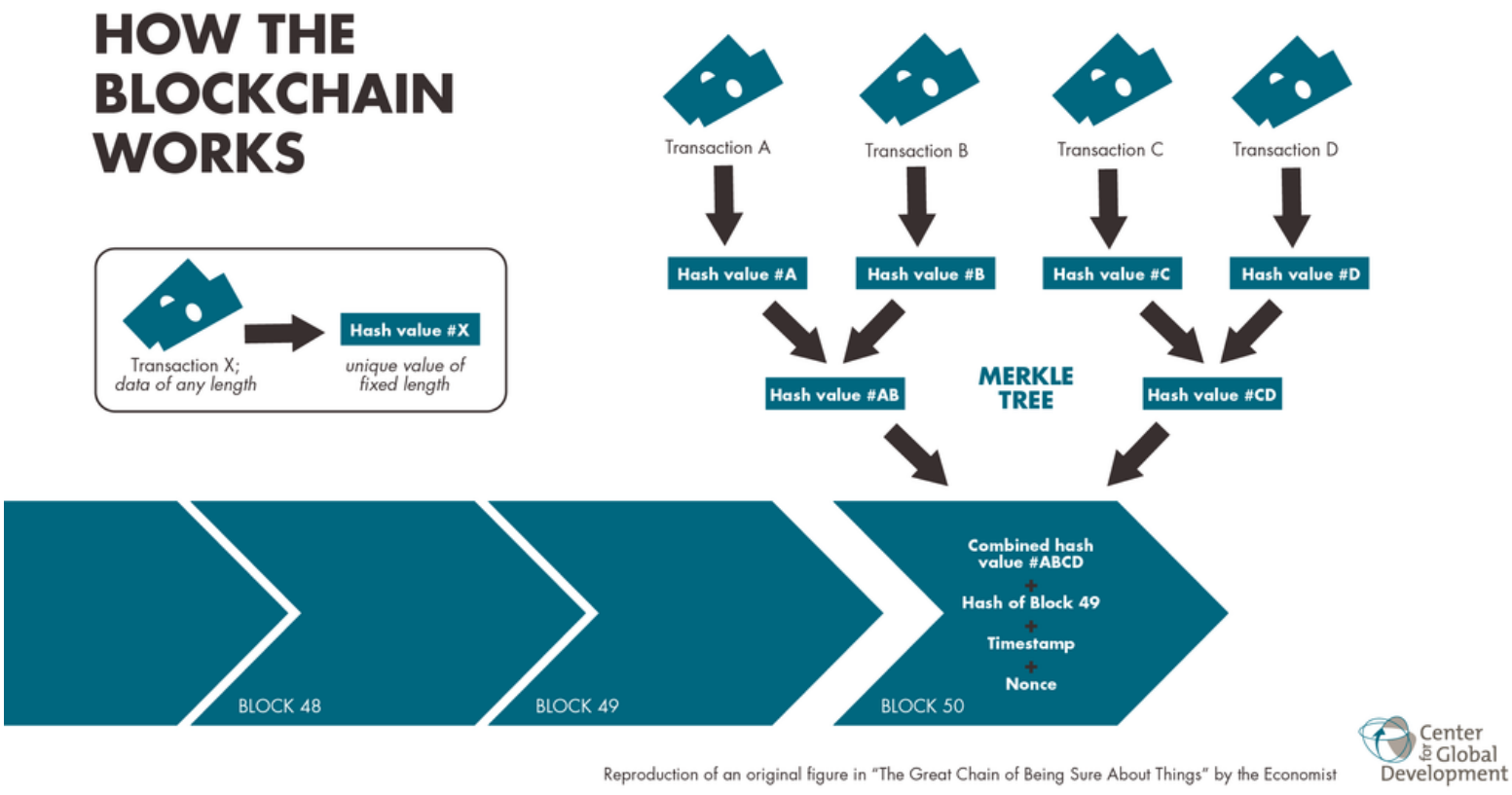
Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

Block Chain



Reproduction of an original figure in "The Great Chain of Being Sure About Things" by the Economist

Center for Global Development

Conclusion

- Block Chain is just a distributed diary which is difficult to forge

Conclusion

- Block Chain is just a distributed diary which is difficult to forge
- It uses hashing heavily to ensure its properties

Conclusion

- Block Chain is just a distributed diary which is difficult to forge
- It uses hashing heavily to ensure its properties
- It also uses binary trees to improve efficiency of operations

Conclusion

- Block Chain is just a distributed diary which is difficult to forge
- It uses hashing heavily to ensure its properties
- It also uses binary trees to improve efficiency of operations
- Learn more about binary trees and efficient operations in the next module!