Blockchain 3

This set of slides was developed based on the reference books as listed in the course document.

Task 3: Storing Transaction Data

7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

Transforming a Book into a Blockchain-Data-Structure

We now explain how to turn a book into a small library with an ordering catalog

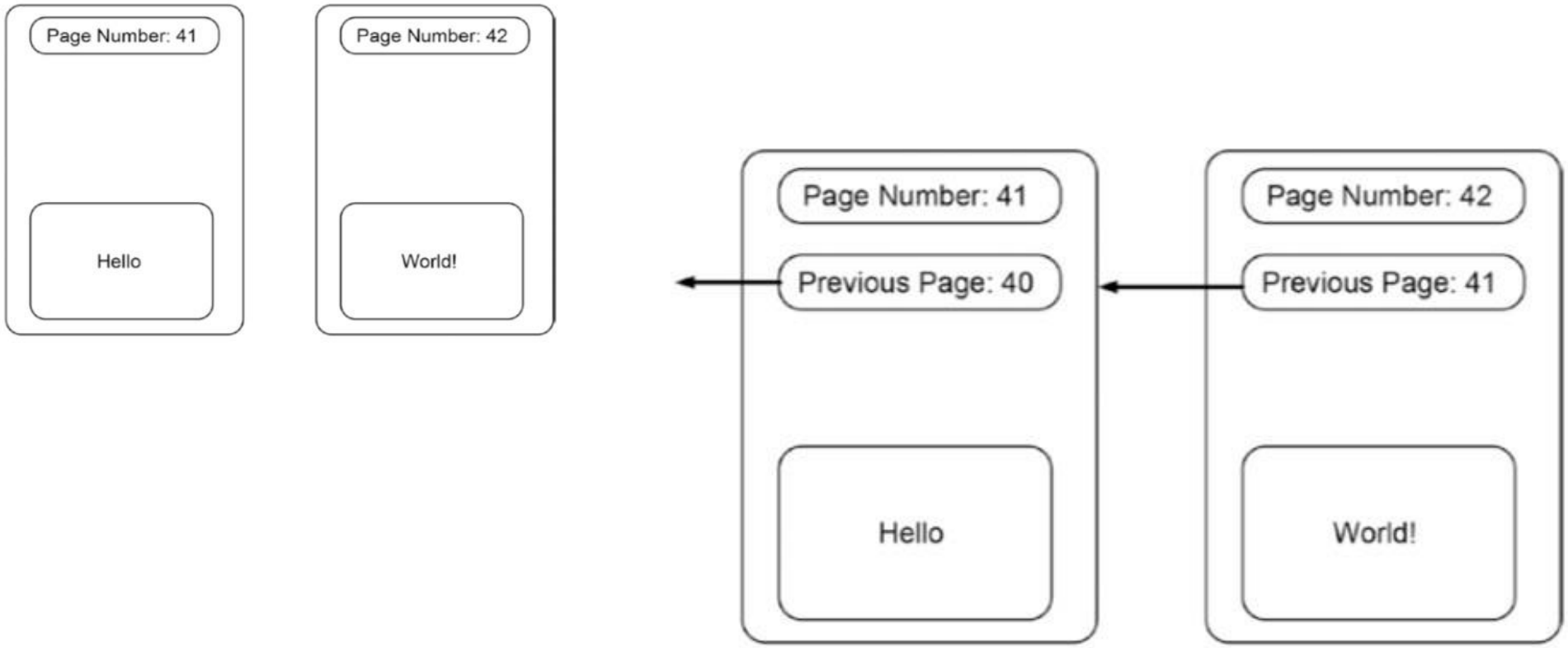
turns out to be a simplified version of the blockchain-data-structure

Some of the important properties of a book include:

- Storing content: Books store content on their pages.
- Ordering:The sentences on the pages as well as the pages within the book are kept in order.
- Connecting pages: Pages are physically connected via the book spine and logically connected via their content and the page numbers.

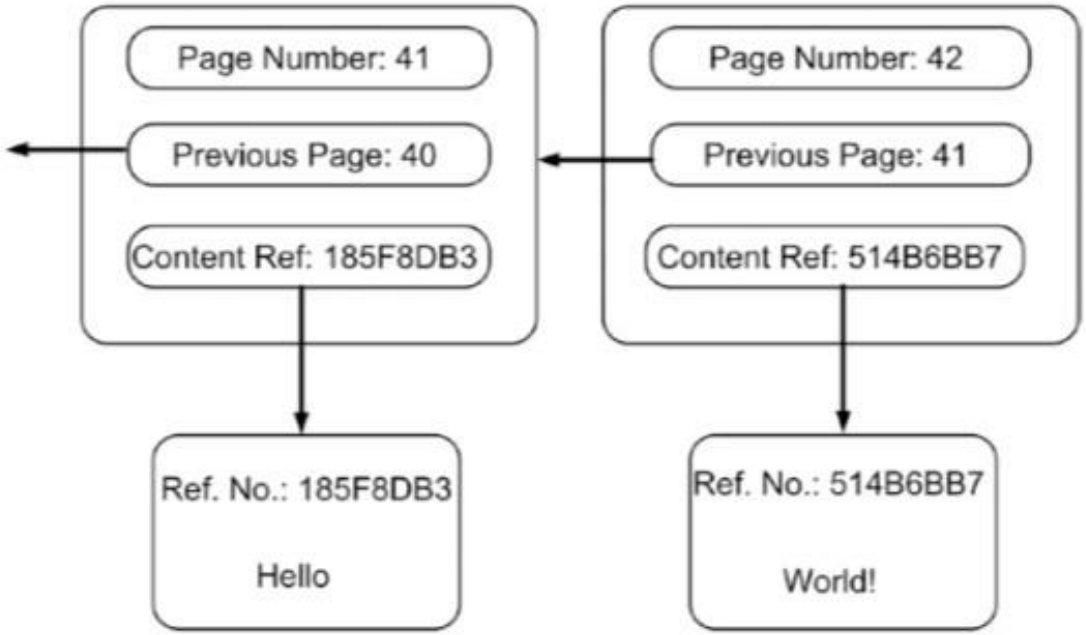
Transforming a Book into a Blockchain-Data-Structure

Transformation 1: Making Page Dependency Explicit



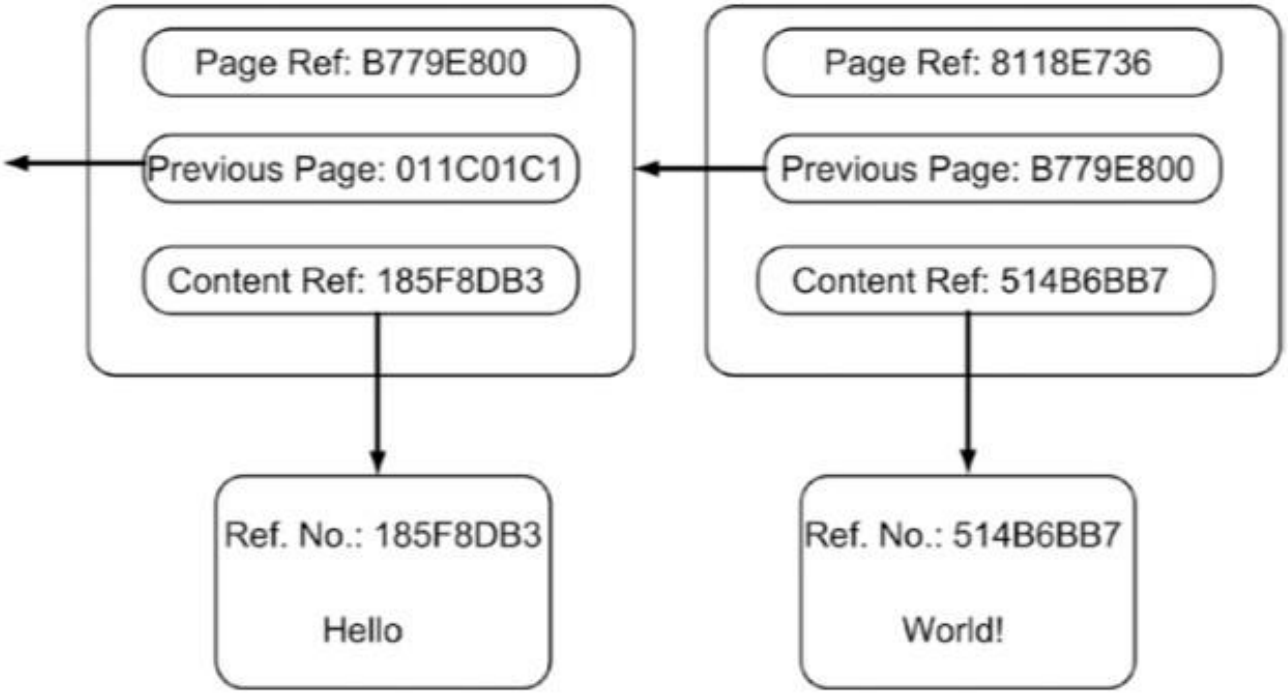
Transforming a Book into a Blockchain-Data-Structure

Transformation 2: Outsourcing the Content



Transforming a Book into a Blockchain-Data-Structure

Transformation 3: Replacing Page Numbers



Transforming a Book into a Blockchain-Data-Structure

Transformation 4: Creating Reference Numbers

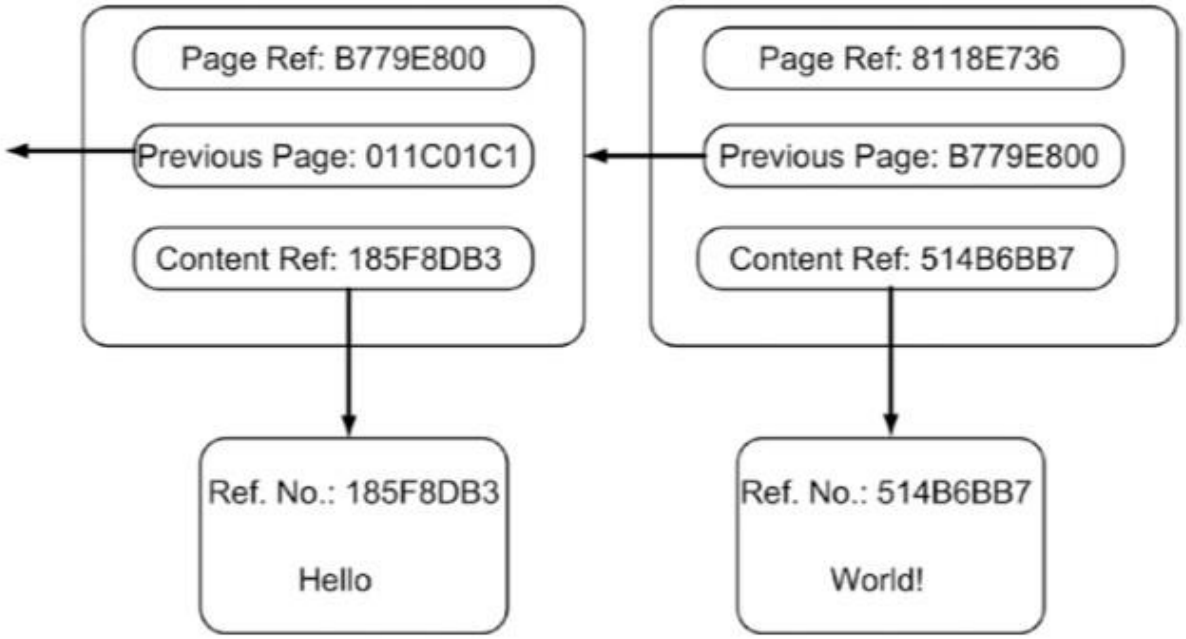
This transformation create unique reference numbers by using cryptographic hash values:

- For example, the content page that contains the word Hello is identified by the shortened hash value of Hello that is 185F8DB3.
- The reference value of our book pages are calculated based on their content, which is the content reference number and the reference number of the preceding page. For example, the page reference number B779E800 is the hash value of 011C01C1 185F8DB3.

Transforming a Book into a Blockchain-Data-Structure

Transformation 4: Creating Reference Numbers

The reference value of our book pages are calculated based on their content, which is the content reference number and the reference number of the preceding page. For example, the page reference number B779E800 is the hash value of 011C01C1 185F8DB3.



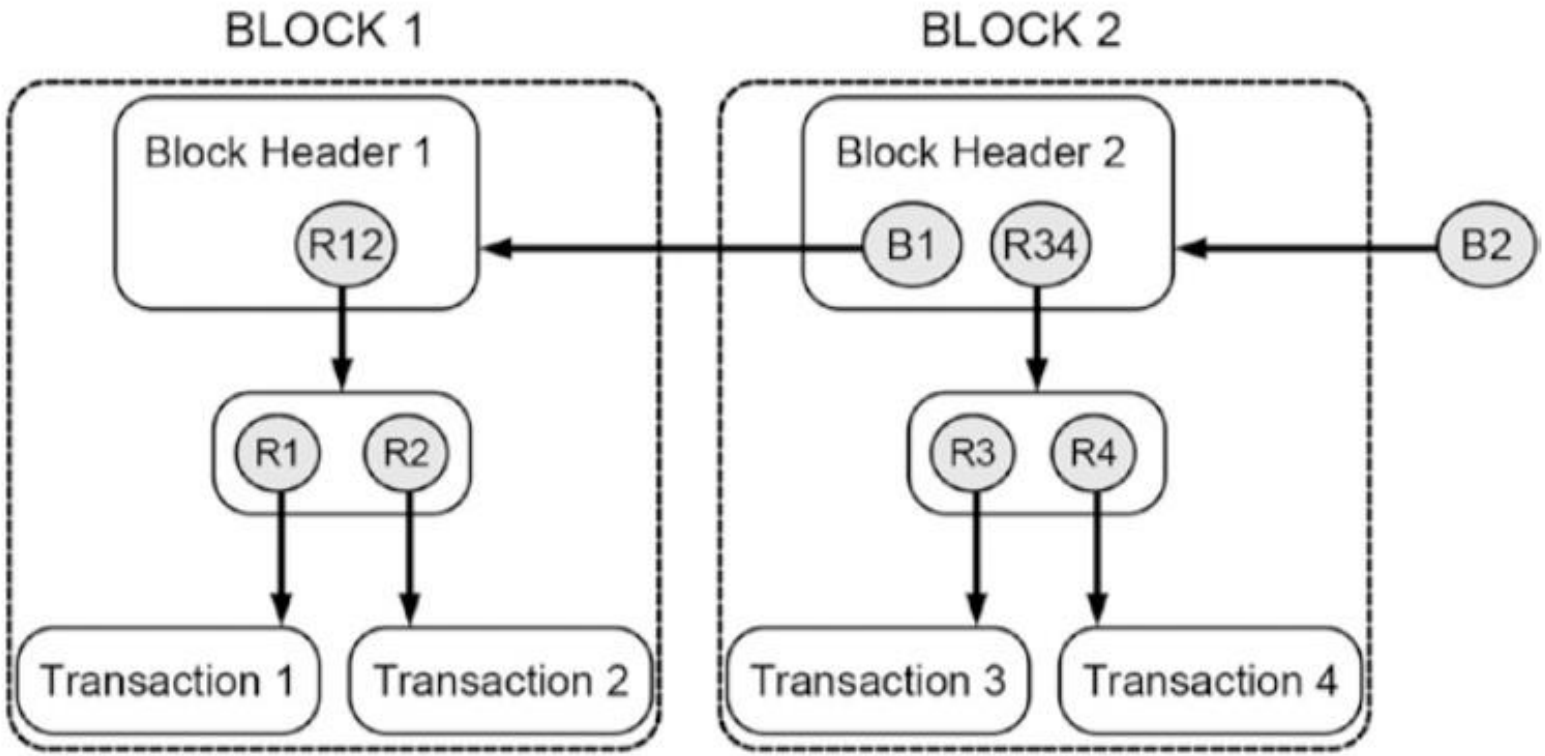
The Blockchain-Data-Structure

Transformed Book	Blockchain-Data-Structure[1]
A page in the ordering catalog	A block header
The whole ordering catalog	The chain of block headers
The reference number of a page in the ordering catalog	The cryptographic hash value of a block header
The reference number to the preceding page	The cryptographic hash value of the preceding block header
Content	Transaction data
A content page	A Merkle tree containing transaction data
Reference to the content page	The root of the Merkle tree that contains transaction data
The mental unit of a page of the ordering catalog and its corresponding content page	One block of the blockchain-data-structure
The whole ordering catalog and all content pages together	The blockchain-data-structure

Merkle tree: https://www.youtube.com/watch?v=fB41w3JcR7U

The Blockchain-Data-Structure:

A simplified blockchain-data-structure containing 4 transactions



Using the Data Store

Task 4: Preparing Ledgers to be distributed in an Untrustworthy Environment

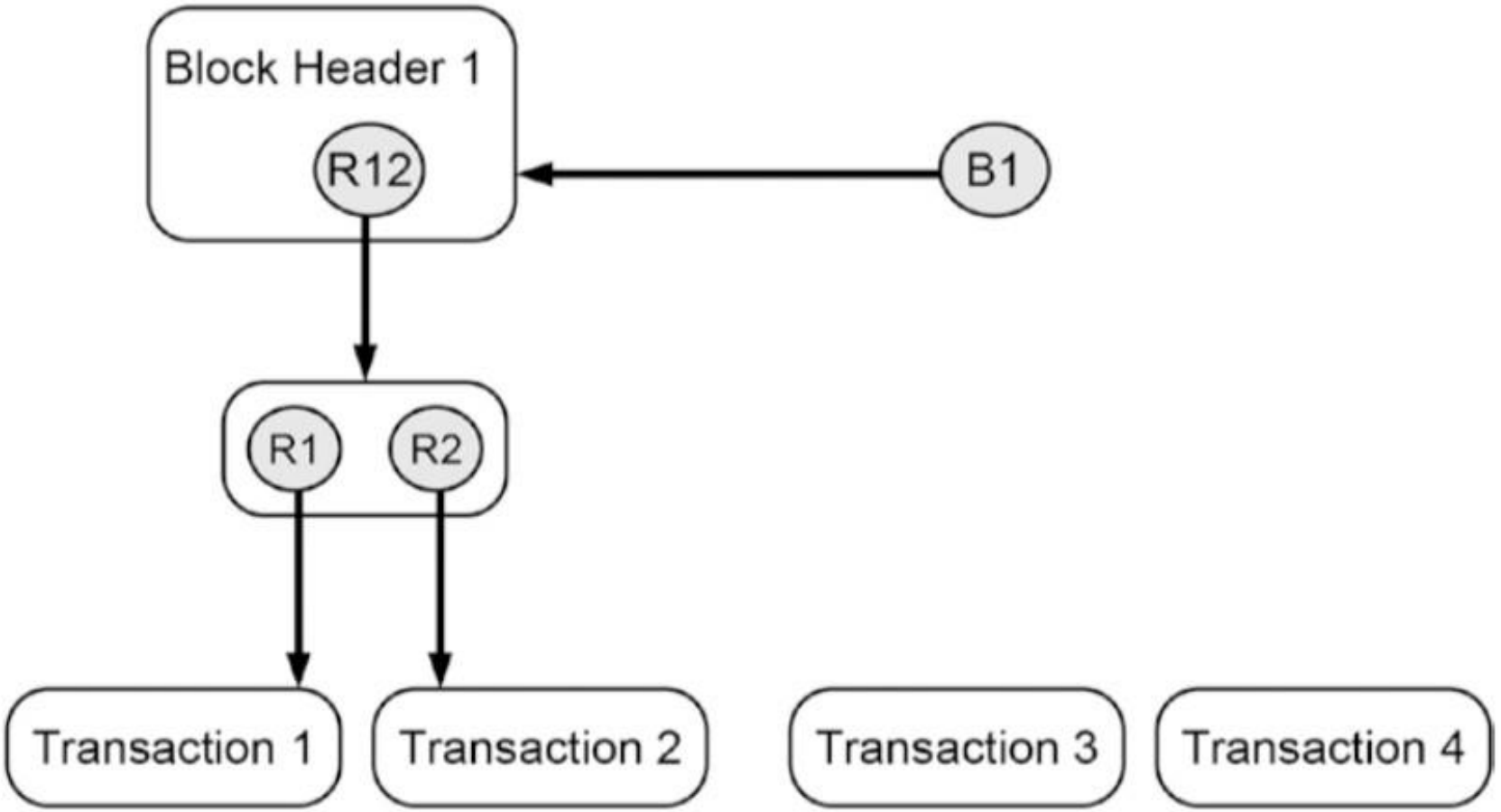
7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

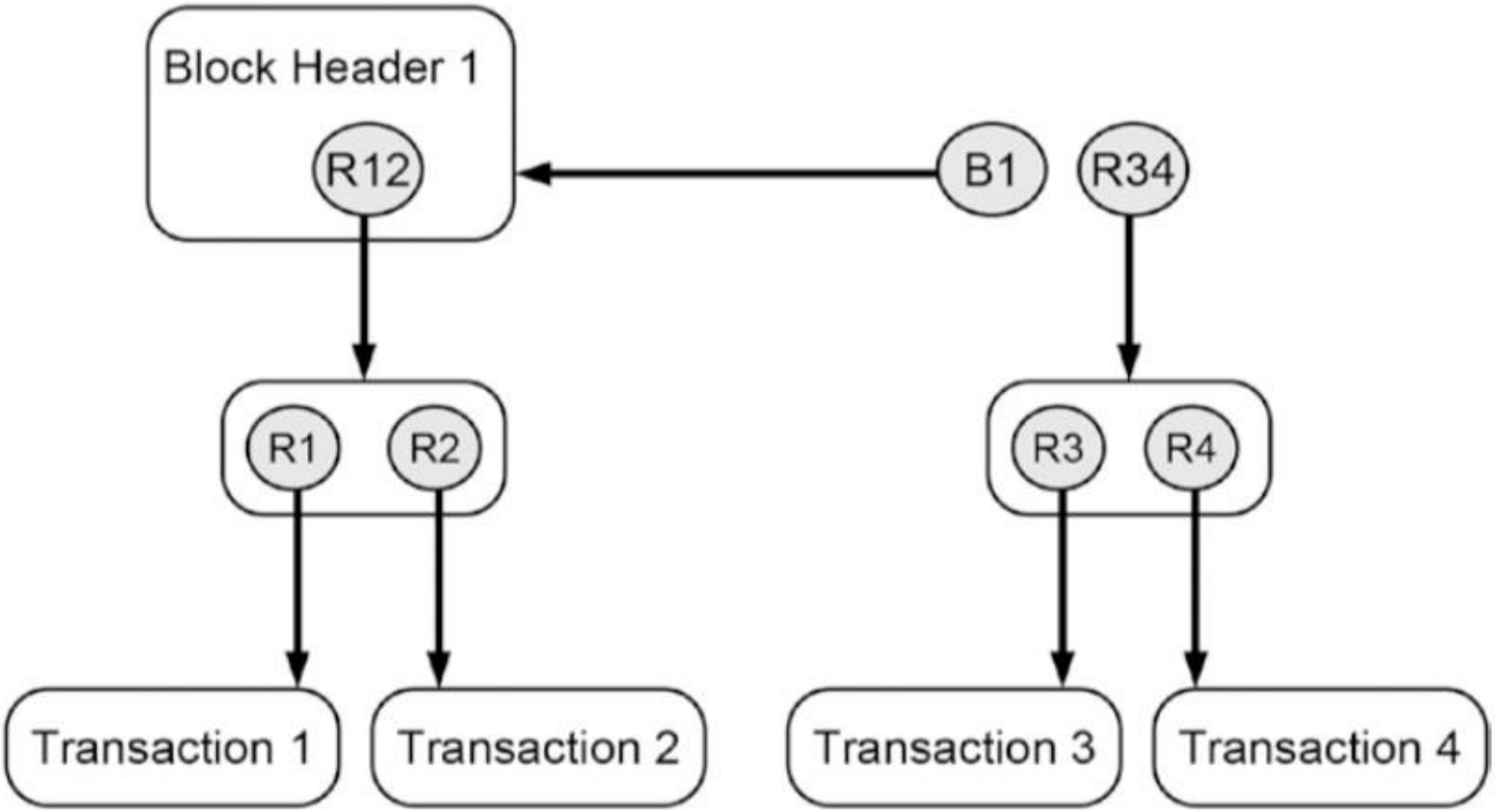
Adding New Transactions

Initial situation: Two new transactions (Transaction 3 and Transaction 4) should be added to the existing blockchain-data-structure



Adding New Transactions - Step 1

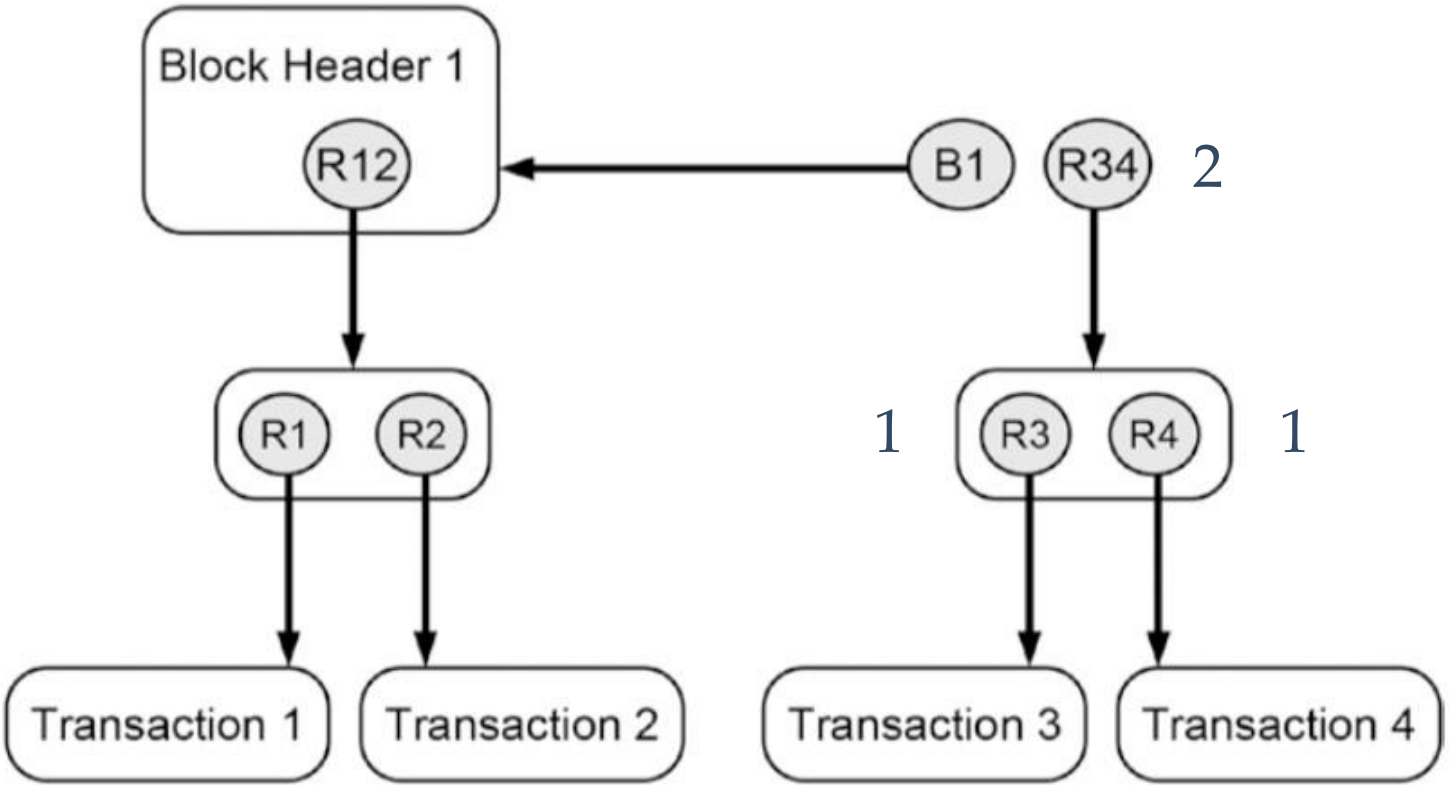
Step 1: Creating a new Merkle tree that contains the new transactions



Adding New Transactions - Step 1

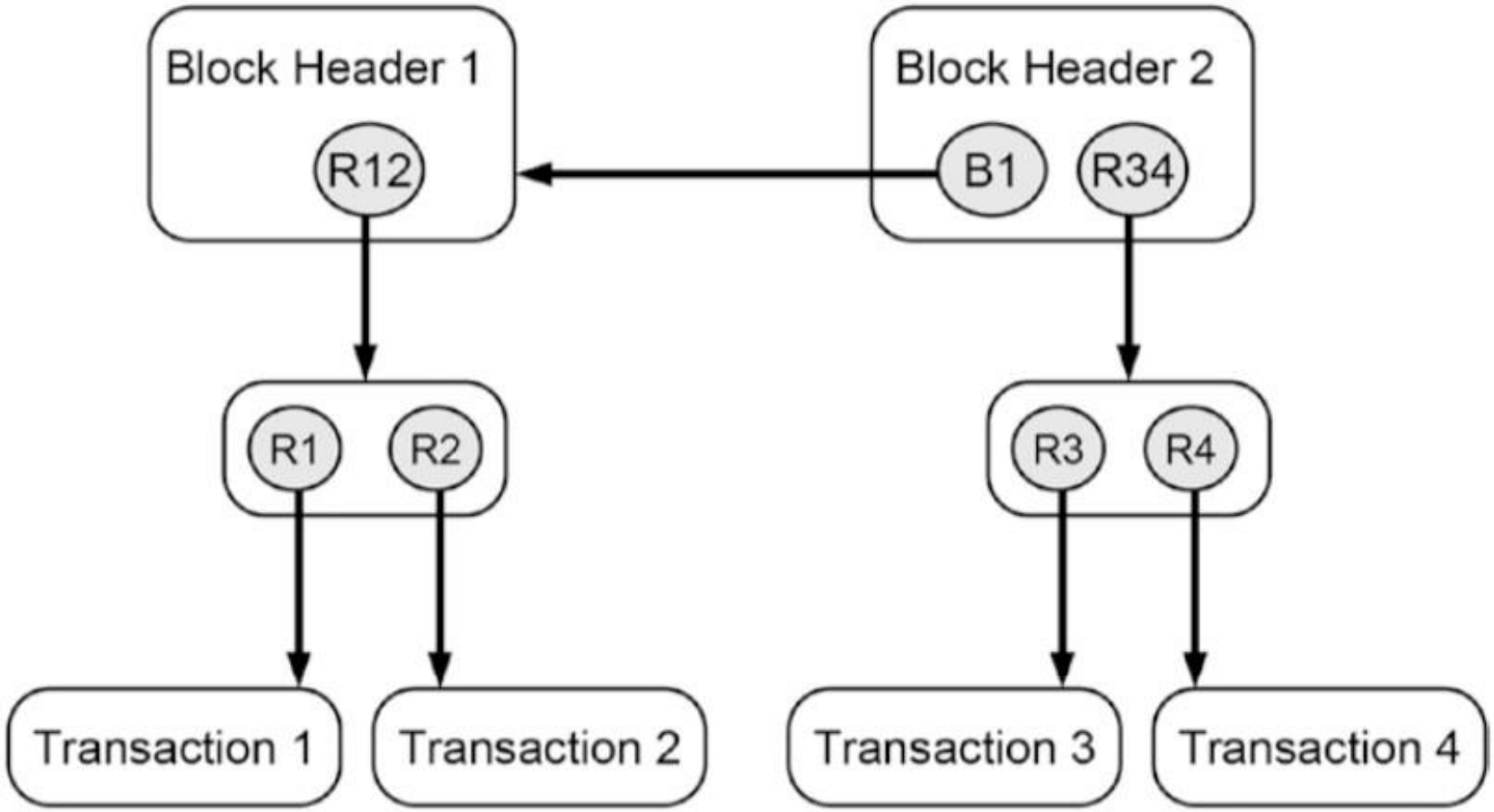
Merkle tree is a tree in which:

1. every leaf node is labelled with the hash of a data block, and
2. every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.



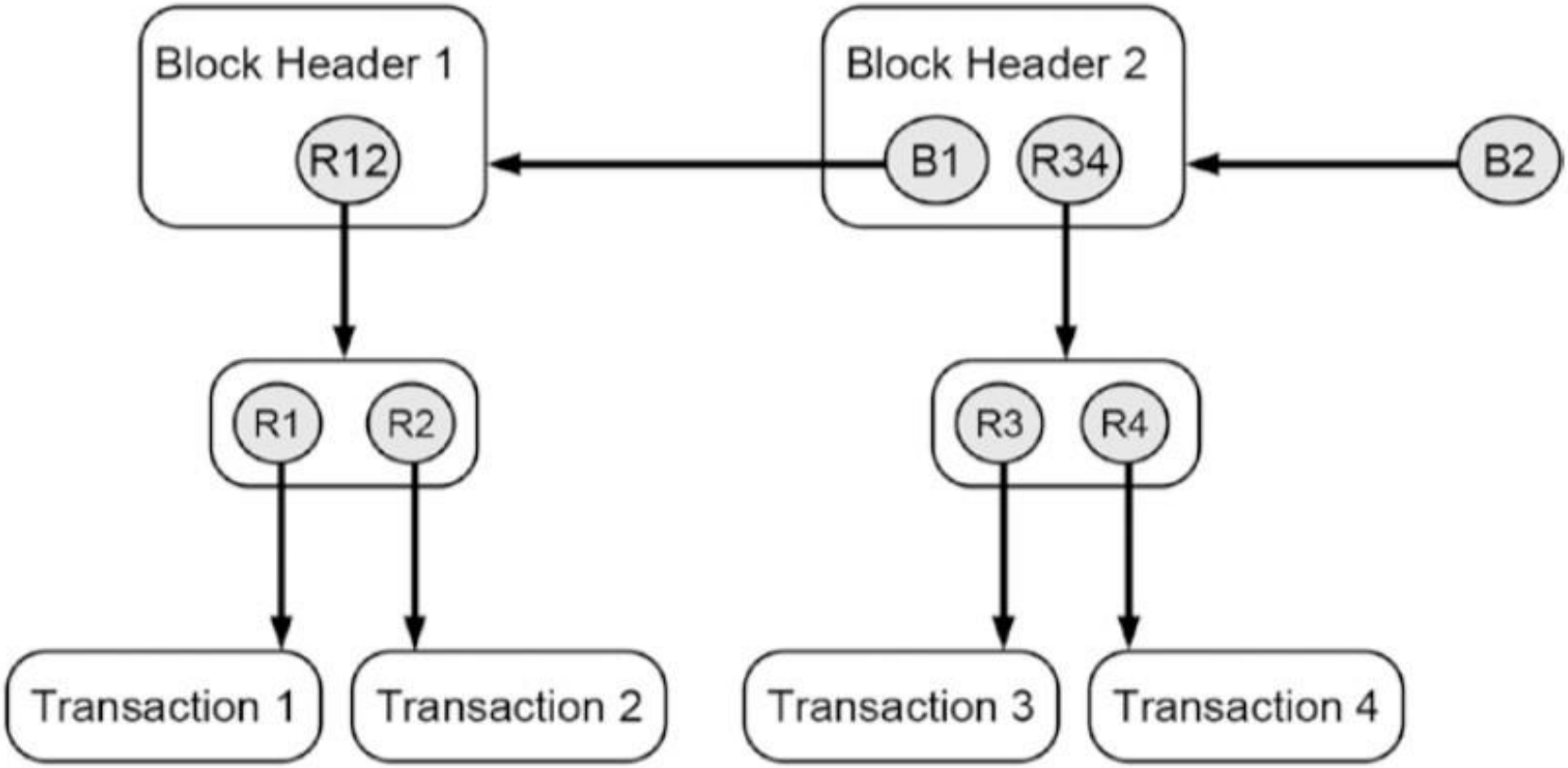
Adding New Transactions - Step 2

Step 2: Create a new block header that contains both the hash reference to its preceding header and the root of the Merkle tree that contains the new transaction data



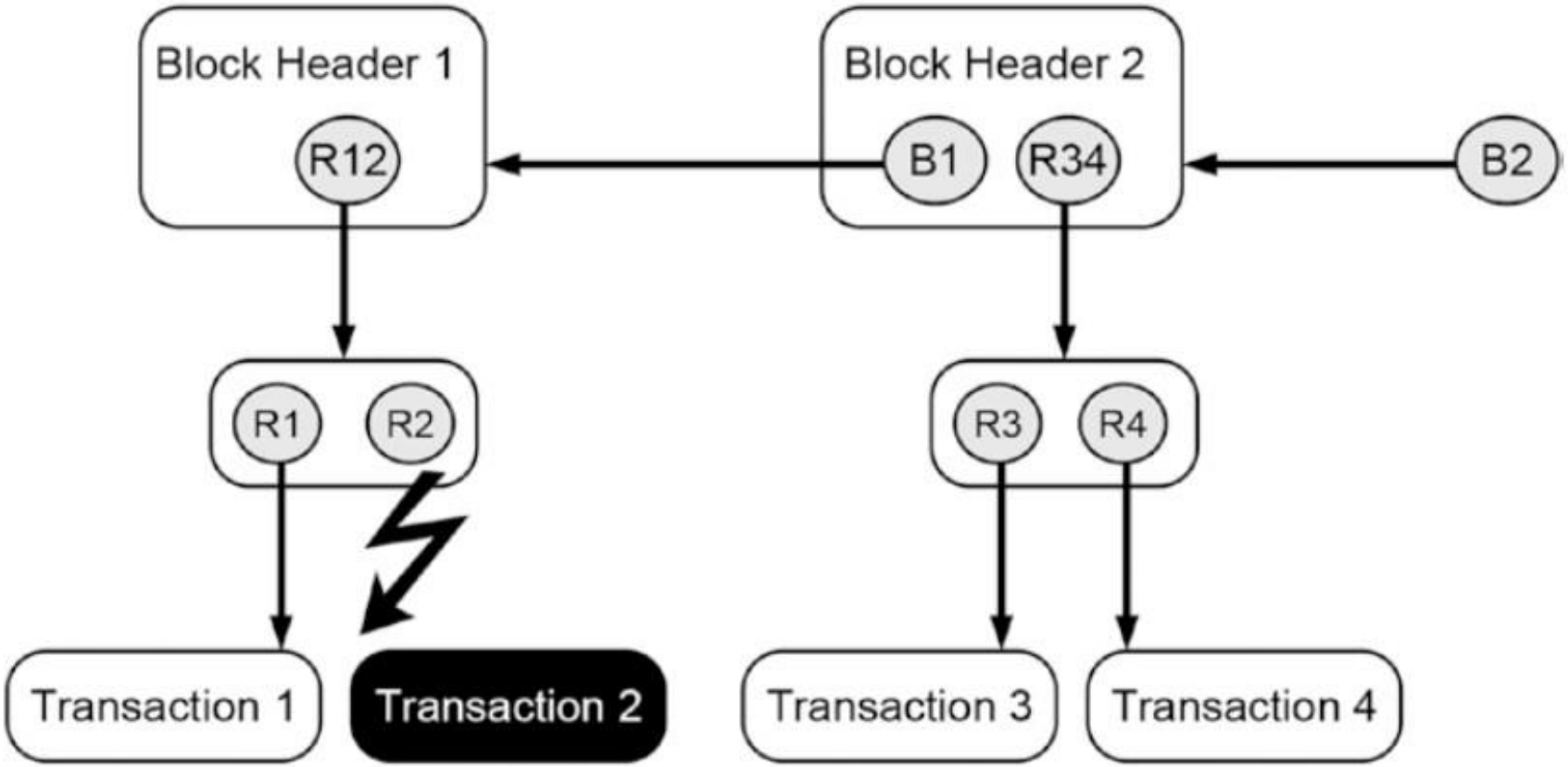
Adding New Transactions - Step 3

Step 3: Create a new hash reference that points to the new block header, which is now the new head of the whole updated blockchain-data-structure



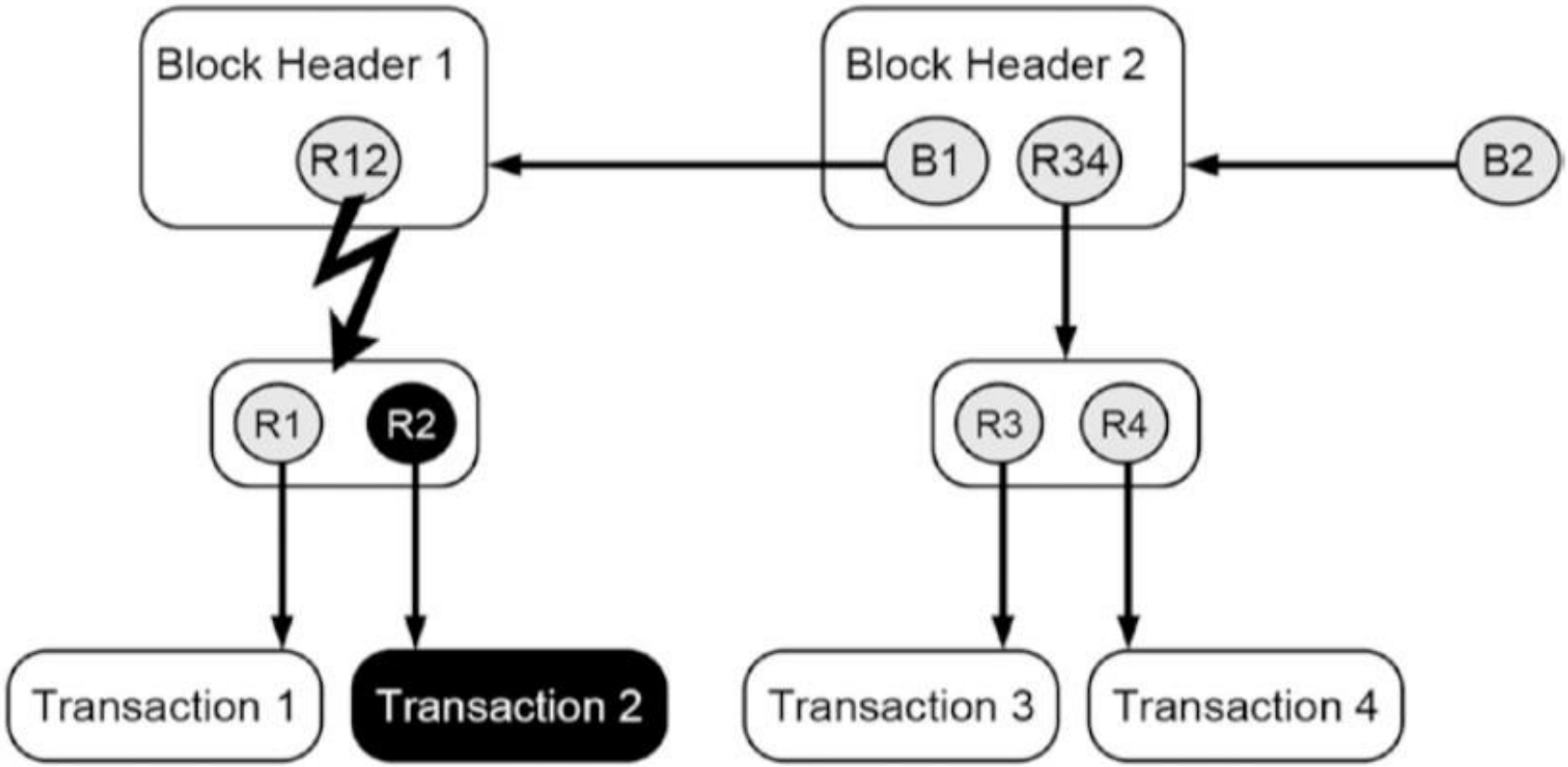
Changing the Content of Transaction Data

Changing the details of a transaction invalidates the hash reference that pointed to the original data, which invalidates the whole data structure



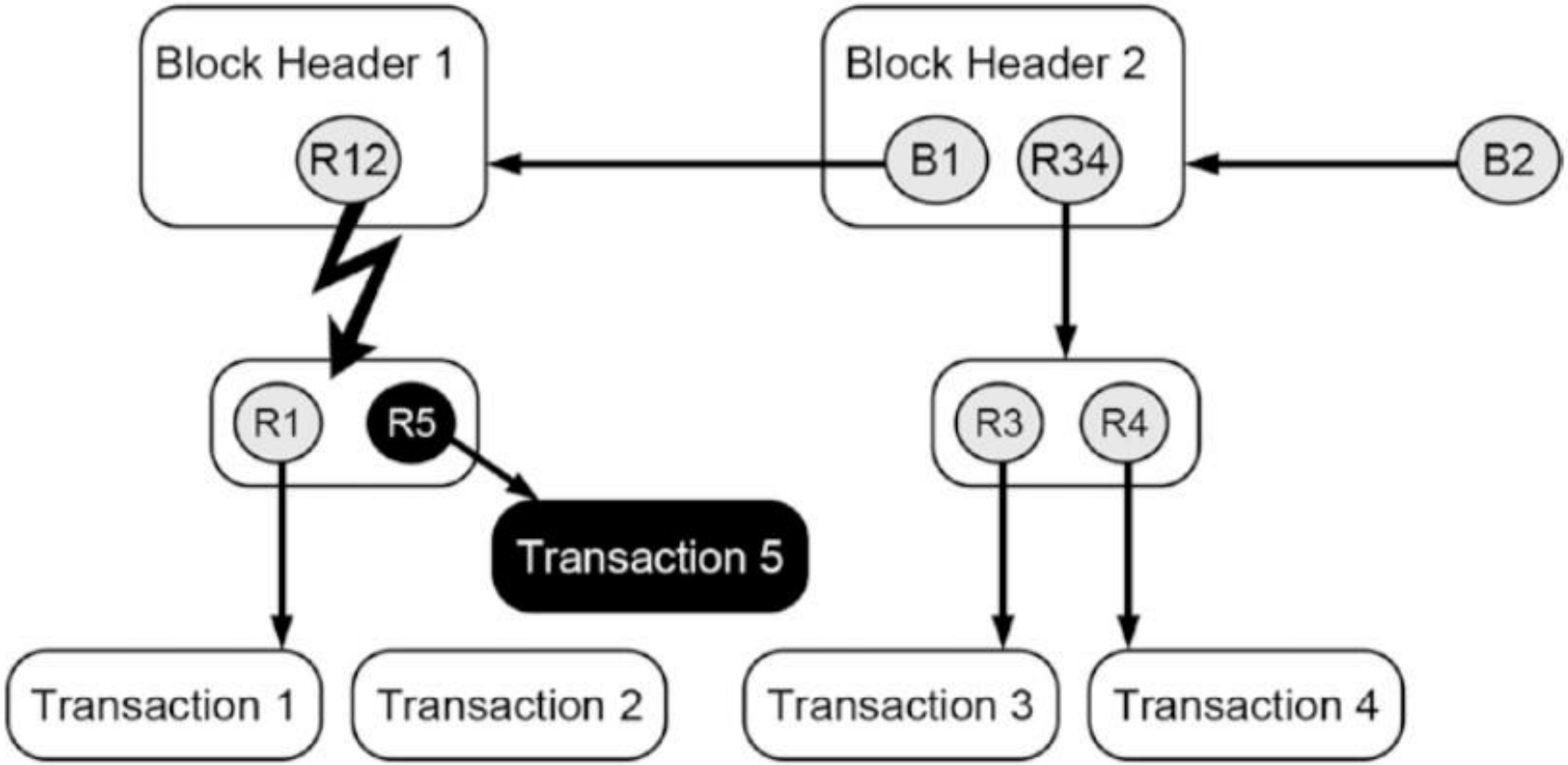
Changing a Reference in the Merkle Tree

Changing a transaction and its hash reference in the Merkle tree invalidates the root of the Merkle tree, which invalidates the whole data structure



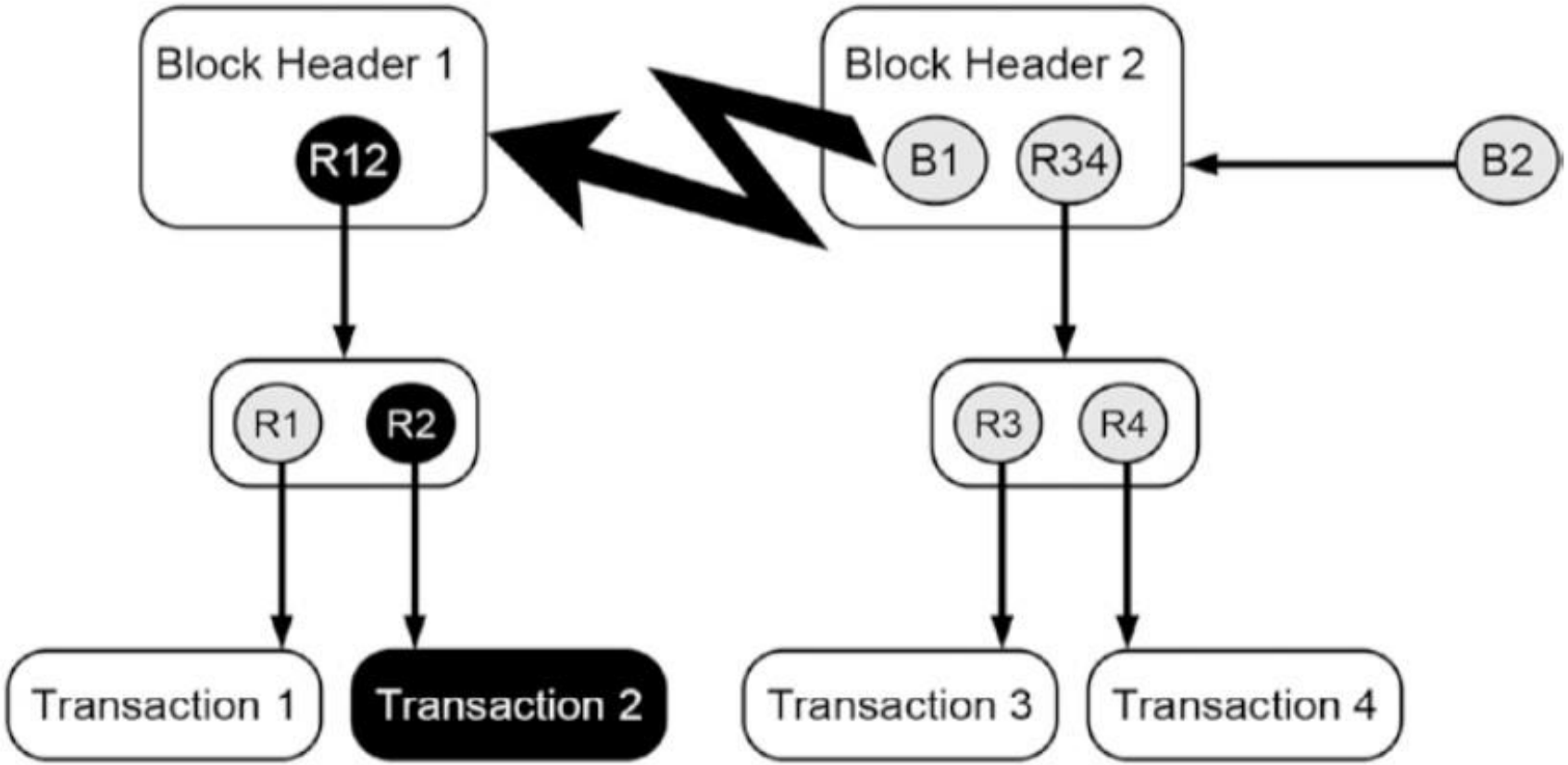
Replacing a Transaction

Replacing a transaction and its hash reference in the Merkle tree invalidates the root of the Merkle tree, which invalidates the whole data structure



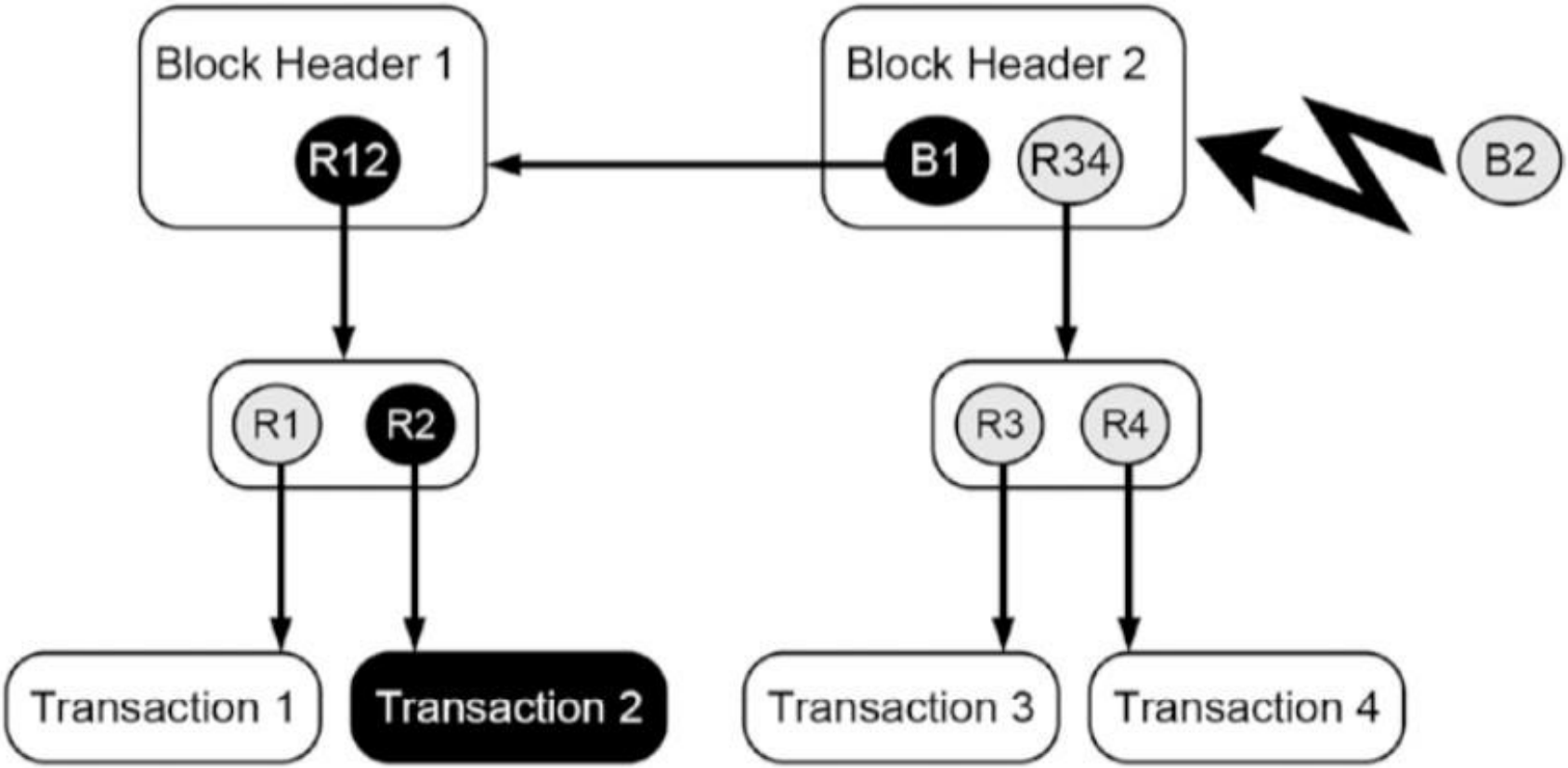
Changing the Merkle Root

Changing a Merkle tree invalidates the hash reference that points to the block header that contains it, which in turn invalidates the whole data structure



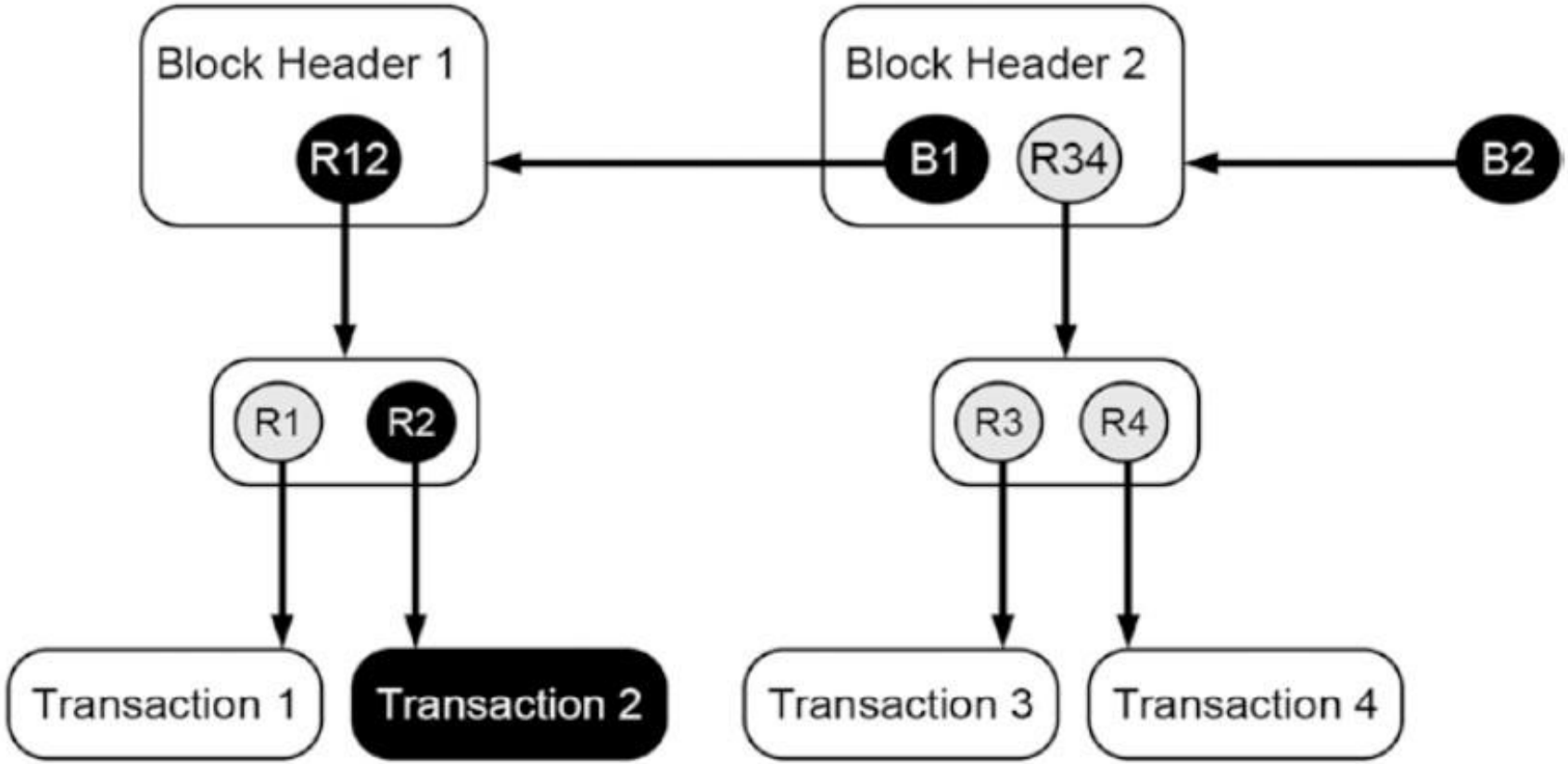
Changing a Block Header Reference

Changing a hash reference within a block header invalidates the hash reference that points to the manipulated block header, which in turn invalidates the whole data structure



Changing Data Orderly

Changing a transaction orderly includes changing all subsequent hash references



Protecting the Data Store

Protecting the Data Store

The above discussion conclude with the finding that the blockchain-data-structure stores data in a *change-sensitive manner*

Now we explains how that property can be used to prepare the history of transaction data to *be shared and distributed* in an *untrustworthy environment* without having to fear that dishonest members of a peer-to-peer system can manipulate its content for its own advantage.

Protecting the Data Store

The procedure to add a new block to the blockchain-data-structure, as discussed above, *is not computationally expensive* because

- it only requires adding the hash reference that points to the current head of the chain to the new block header and declaring it as the new head of the chain.

The method of making the blockchain-data-structure immutable is to make *adding a new block a computationally expensive task*. The following aspects need to be considered in the course of achieving this:

- Validation rules for block headers
- Compulsory data of block headers
- The process of creating a new block header

Compulsory data of block headers

Every block header of the blockchain-data-structure has to carry at least the following data:

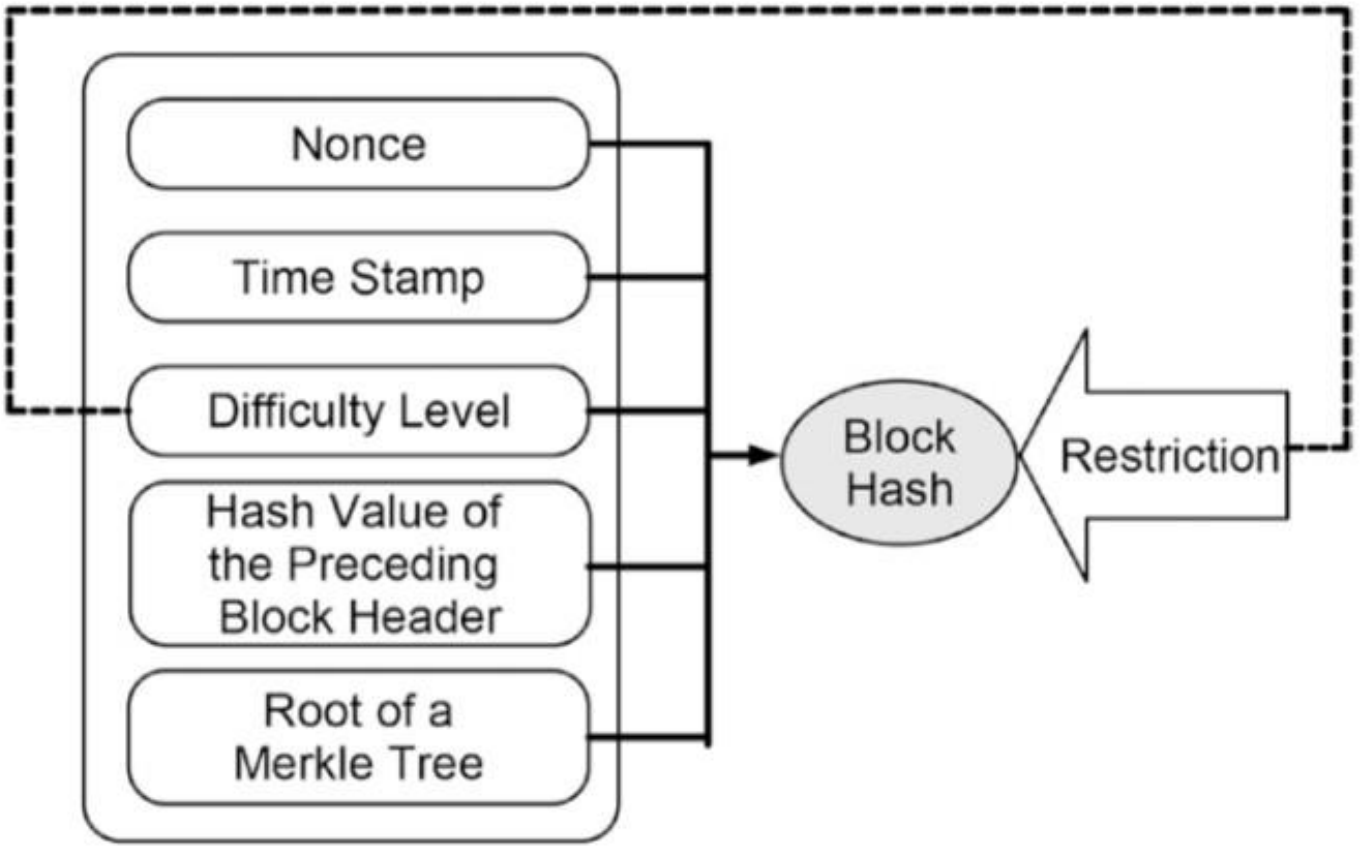
- The root of a Merkle tree containing transaction data
- A hash reference to the header of the preceding block
- The difficulty level of the hash puzzle
- The time when solving the hash puzzle started
- The nonce that solves the hash puzzle

The process of creating a new block header

Creating a new block involves the following steps:

1. Get the root of the Merkle tree that contains the transaction data to be added.
2. Create a hash reference to the header of that block that will be the predecessor from the new block **header's** point of view.
3. Obtain the required difficulty level.
4. Get the current time.
5. Create a **preliminary block header** that contains the data mentioned in points 1 to 4.
6. Solve the hash puzzle for the preliminary block header.
7. Finish the new block by adding the **nonce** that solves the hash puzzle to the preliminary header.

The process of creating a new block header



Note that the difficulty level is part of the block header and hence is also part of the block's hash value. This ensures that no one can bypass the computational costs of the hash puzzle by arbitrarily reducing the difficulty level.

Validation rules for block headers

Every block header of has to fulfill the following rules:

1. It must contain a valid hash reference to a previous block.
2. It must contain a valid root of a Merkle tree containing transaction data.
3. It must contain a correct difficulty level.
4. Its time stamp is after the time stamp of its preceding block header.
5. It must contain a nonce.
6. The hash value of all the five pieces of data combined together fulfills the difficulty level.

7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

Task 5: Forming a System of Distributed the Ledgers

Distributing the Data Store Among Peers

The above turned the blockchain-data-structure into an *immutable append-only data store*, which can be used as a *manipulation-resistant ledger* for transaction data.

Having one immutable append-only history of transaction data in insolation may be of limited value for the goal of clarifying ownership based on a group of computers that acts as witnesses of ownership-related events. Hence, we need establishing a purely distributed peer-to-peer system that allows sharing of information about transactions.

We will not go further on how to achieve that.