Module 4

Ethereum

Ethereum is like a decentralized computing network.

It allows developers to create and run applications on its blockchain using smart contracts.

Ethereum was initially released in 2015. Within two years of its release, it was ranked the second-best blockchain network, Bitcoin is the first.

Ethereum is a Blockchain network that introduced a built-in Turing-complete programming language that can be used for creating various decentralized applications(also called Dapps).

The Ethereum network is fueled by its own cryptocurrency called ‘ether’.

Ethereum is often called Blockchain 2.0 since it proved the potential of blockchain technology beyond the financial sector.

The consensus mechanism used in Ethereum is Proof of Stakes(PoS), which is more energy efficient when compared to that used in the Bitcoin network, that is, Proof of Work(PoW).

History of Ethereum

- **2013:** Ethereum was first described in Vitalik Buterin’s white paper in 2013 with the goal of developing decentralized applications.
- **2014:** In 2014, EVM was specified in a paper by Gavin Wood, and the formal development of the software also began.
- **2015:** In 2015, Ethereum created its genesis block marking the official launch of the platform.
- **2018:** In 2018, Ethereum took second place in Bitcoin in terms of market capitalization.

- **2021:** In 2021, a major network upgrade named London included Ethereum improvement proposal 1559 and introduced a mechanism for reducing transaction fee volatility.
- **2022:** In 2022, Ethereum has shifted from PoW(Proof-of-Work) to PoS(Proof-of-State) consensus mechanism, which is also known as Ethereum Merge. It has reduced Ethereum's energy consumption by ~ 99.95%.

Features of Ethereum

- **Smart contracts**: Ethereum allows the creation and deployment of smart contracts. Smart contracts are created mainly using a programming language called solidity. Solidity is an Object Oriented Programming language that is comparatively easy to learn.

- **Ethereum Virtual Machine (EVM)**: It is designed to operate as a runtime environment for compiling and deploying Ethereum-based smart contracts.

- **Ether**: Ether is the cryptocurrency of the Ethereum network. It is the only acceptable form of payment for transaction fees on the Ethereum network.

- **Decentralized applications (Daaps):** Dapp has its backend code running on a decentralized peer-to-peer network. It can have a frontend and user interface written in any language to make calls and query data from its backend. They operate on Ethereum and perform the same function irrespective of the environment in which they get executed.

- **Decentralized autonomous organizations (DAOs):** It is a decentralized organization that works in a democratic and decentralized fashion. DAO relies on smart contracts for decision-making or decentralized voting systems within the organization

Type of Ethereum Accounts

Ethereum has two types of accounts:

1) **Externally owned account (EOA)**
2) **Contract Account**

(EOA) :

- They are controlled by private keys.
- Each EOA has a public-private key pair.
- The users can send messages by creating and signing transactions.

Contract Account:

- Contract accounts are controlled by contract codes.
- These codes are stored with the account.
- Each contract account has an ether balance associated with it.

How Does Ethereum Work?

- Ethereum implements an *execution environment* called Ethereum Virtual Machine **(EVM).**

- When a transaction triggers a smart contract all the *nodes* of the network will *execute every instruction.*

- All the *nodes* will *run the EVM* as part of the block verification.

- All the *nodes* on the network must *perform the same calculations* for keeping their ledgers in sync.

- Every ***transaction*** must include:
  - ***Gas limit*** *(amount of processing time for transaction)*
  - ***Transaction Fee (****that the sender is willing to pay for the transaction).*

The Ethereum blockchain

- Ethereum can be visualized as a *transaction-based state machine*.
- In the following diagram, the *Ethereum state transition function* is shown, where a transaction execution has resulted in a state transition:

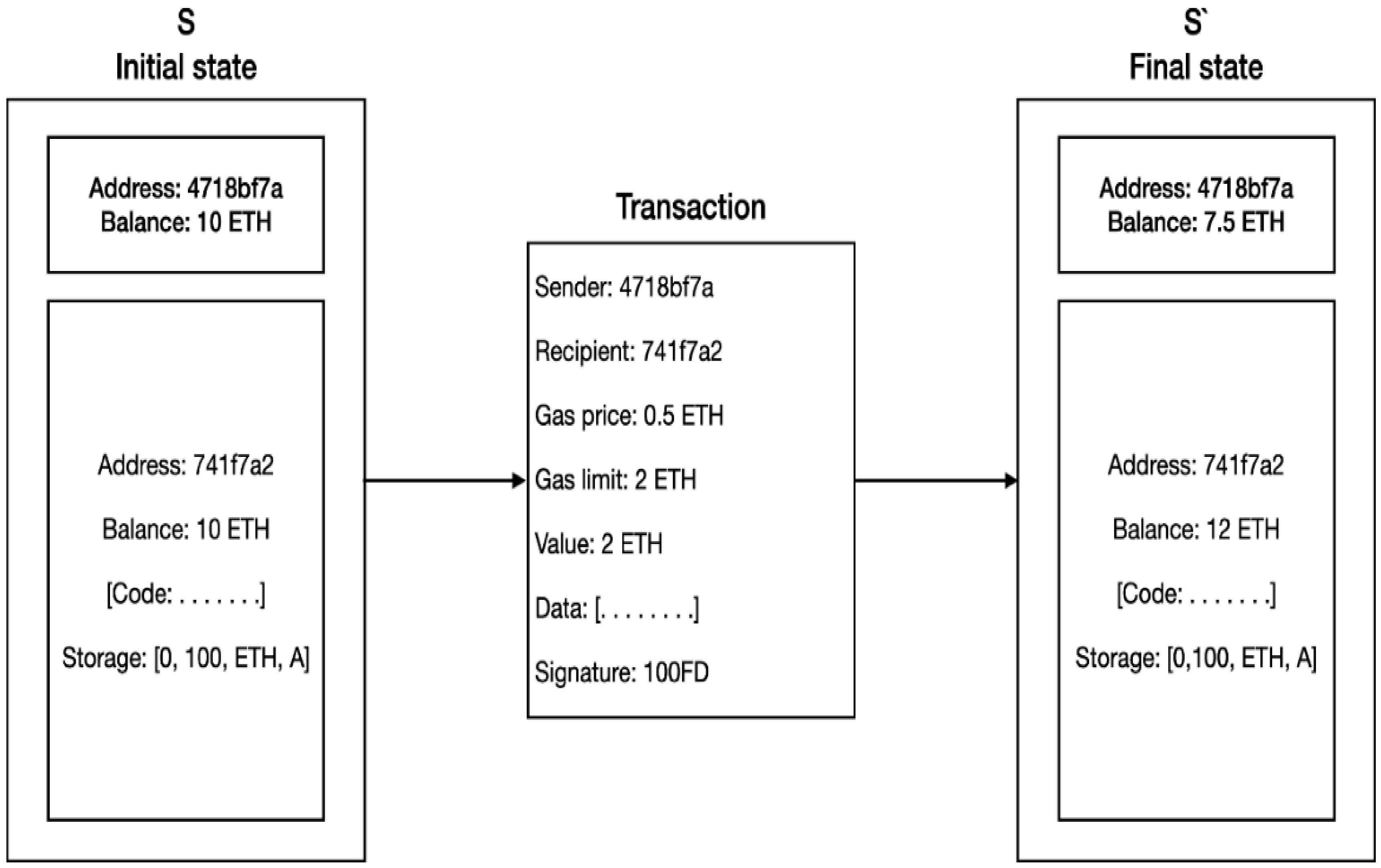


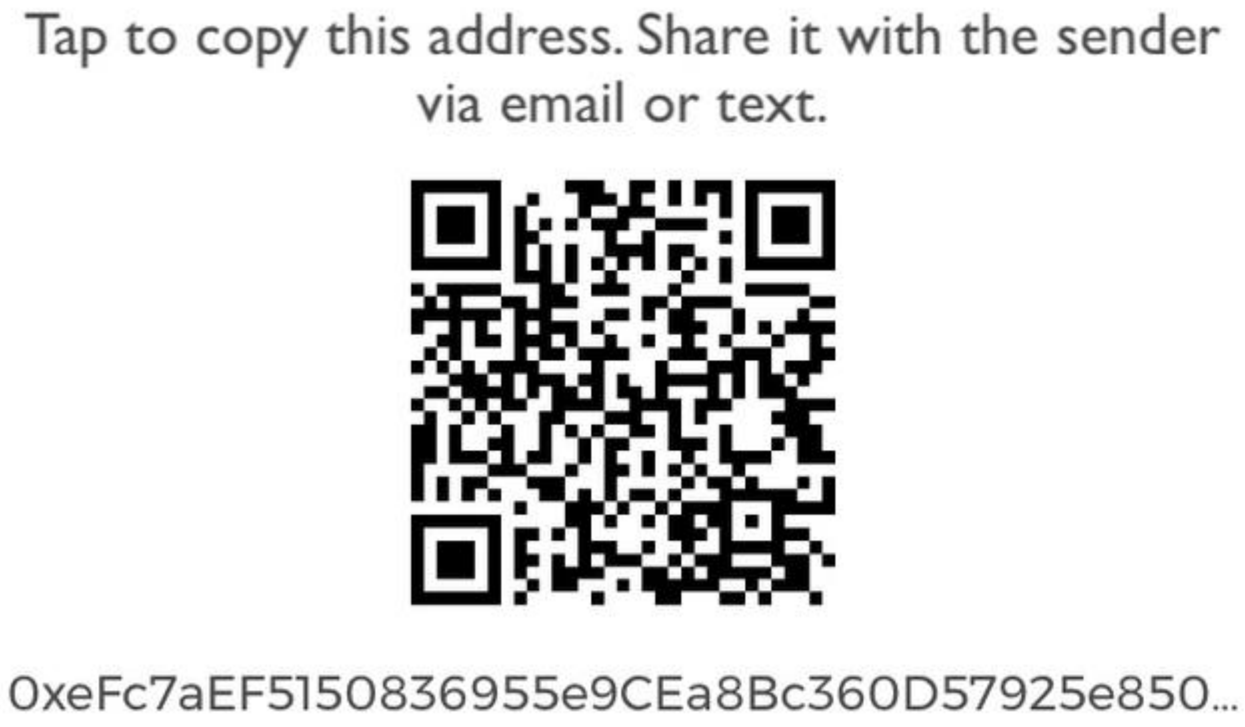
Figure 11.1: Ethereum state transition function

Ethereum – a user's perspective

1. First, either a *user requests* money by sending the request to the sender, or the *sender decides* to send money to the receiver.

   (**Jaxx Ethereum wallet software** is used here)

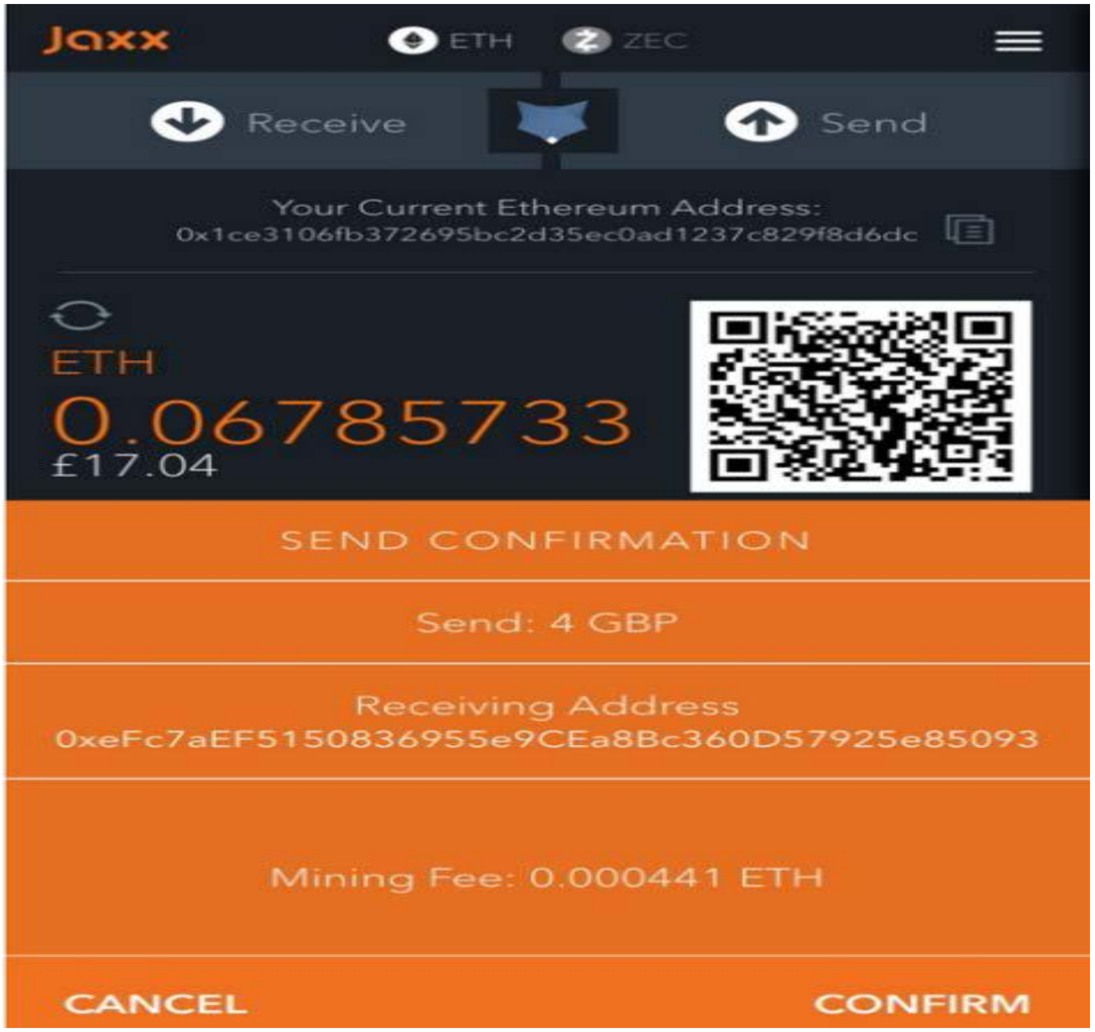
- For example, there are two users, Bashir and Irshad.
- If Irshad requests money from Bashir, then she can *<u>send a request</u>* to Bashir by using a *<u>QR code</u>*.
- Once Bashir receives this request he will either *<u>scan the QR code</u>* or manually *<u>type</u>* in Irshad's *<u>Ethereum address</u>* and send the ether to Irshad's address.
- This QR code can be shared via email, text, or any other communication method:


Tap to copy this address. Share it with the sender via email or text.

0xeFc7aEF5150836955e9CEa8Bc360D57925e850...

Figure 11.2: QR code as shown in the blockchain wallet application

2. Once Bashir receives this request he will either *scan this QR code or copy the Ethereum address* in the **Ethereum wallet software** and initiate a transaction



Your Current Ethereum Address:
0x1ce3106fb372695bc2d35ec0ad1237c829f8d6dc

ETH
0.06785733
£17.04

SEND CONFIRMATION

Send: 4 GBP

Receiving Address
0xeFc7aEF5150836955e9CEa8Bc360D57925e85093

Mining Fee: 0.000441 ETH

3. The *request* is then *broadcasted to the Ethereum network.*

   (The transaction is digitally signed by the sender as proof that he is the owner of the ether)

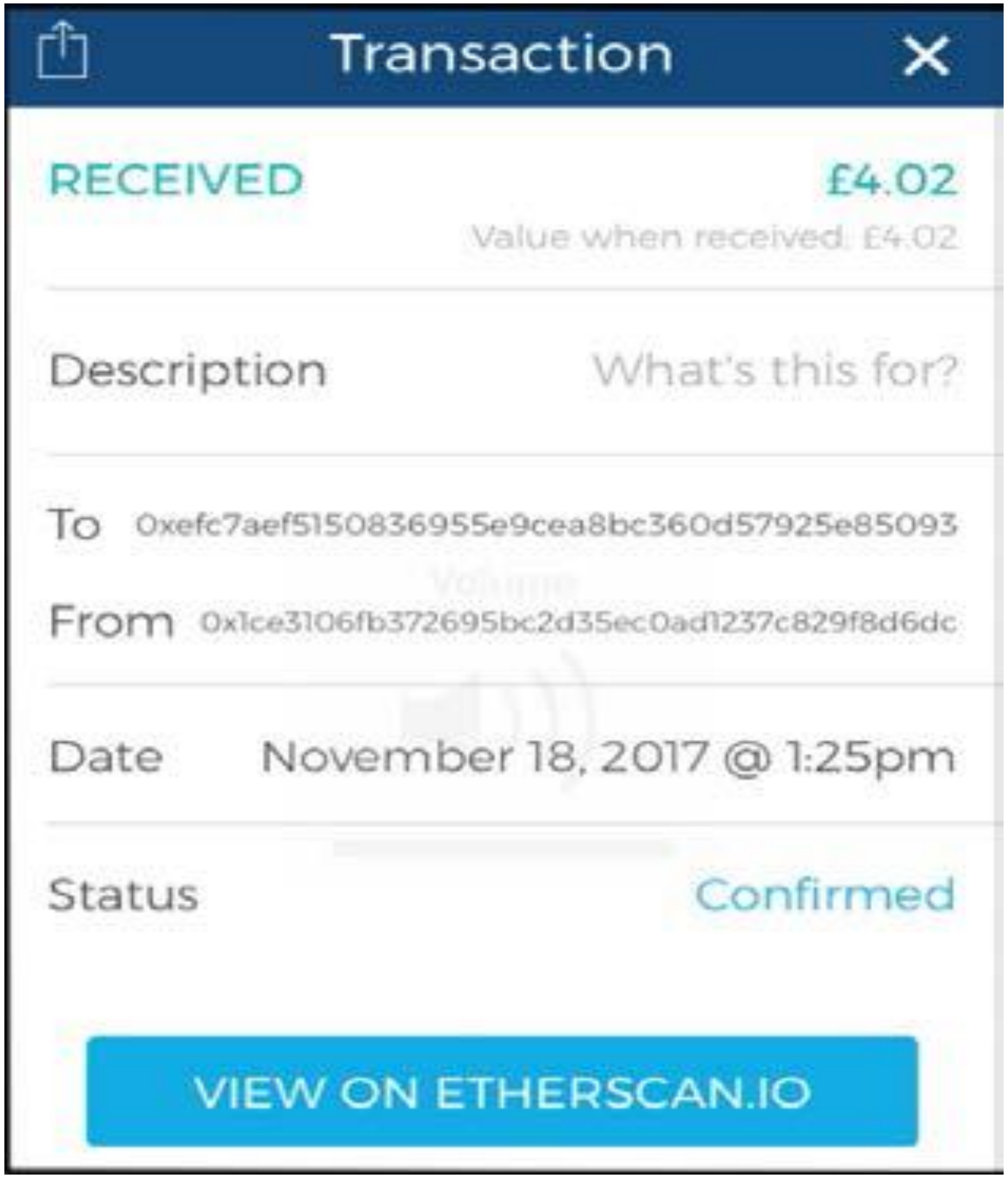
4. This *transaction* is then picked up by nodes called miners on the Ethereum network for *verification and inclusion* in the block.

5. Once it is verified and included in the block, the *PoW process starts*

6. Once a *miner finds the answer to the PoW* problem, this *block is immediately broadcasted* to the rest of the nodes, which then verifies the block and PoW.

7. If all the checks pass then *this block is added to the blockchain,* and miners are paid rewards accordingly.

8. Finally, Irshad gets the ether, and it is shown in her wallet software. This is shown in the following screenshot:


Transaction

RECEIVED £4.02

Value when received: £4.02

Description What's this for?

To 0xefc7aef5150836955e9cea8bc360d57925e85093

From 0x1ce3106fb372695bc2d35ec0ad1237c829f8d6dc

Date November 18, 2017 @ 1:25pm

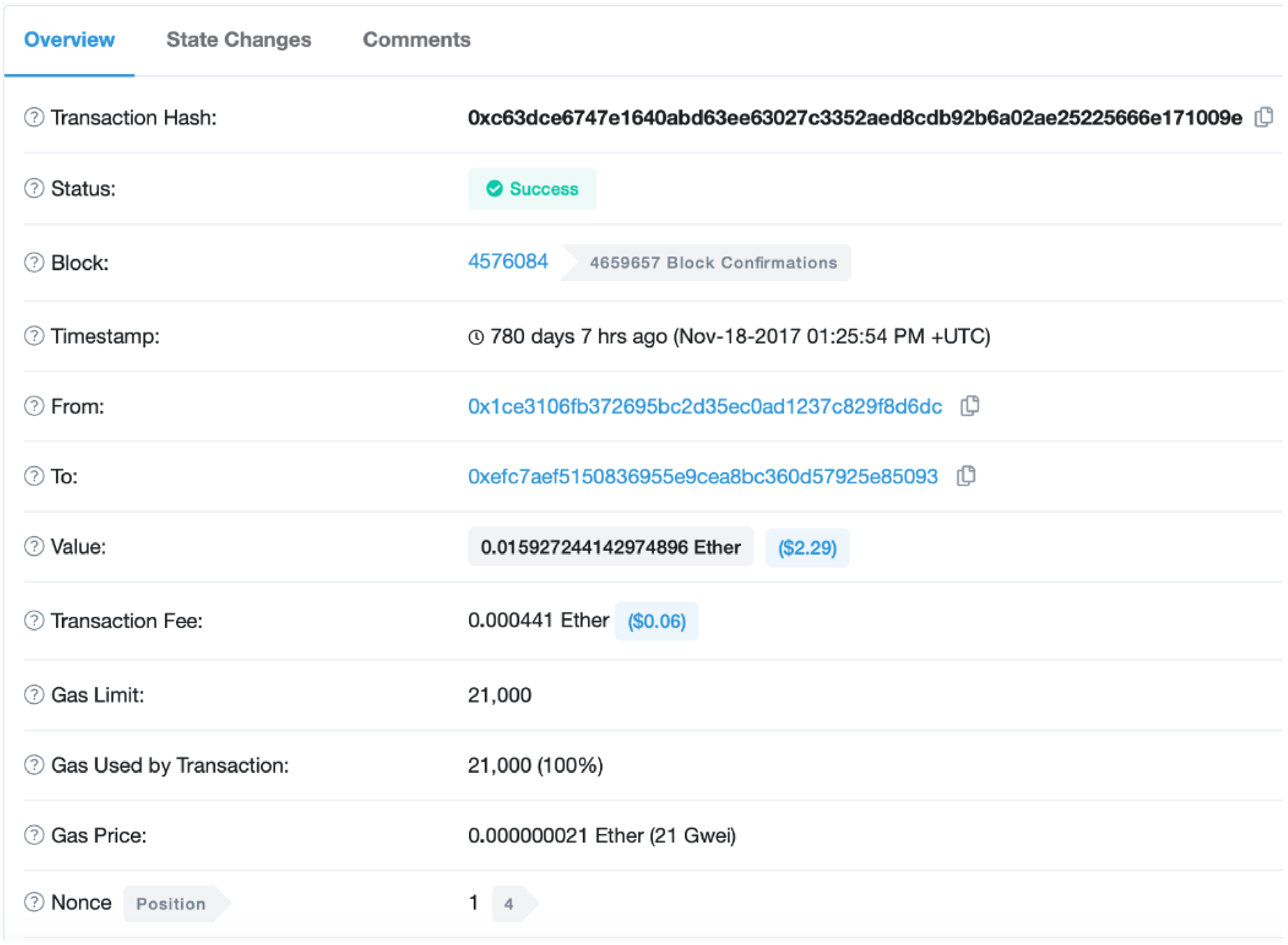
Status Confirmed

Figure 11.4: The transaction received in Irshad's blockchain wallet

- On the blockchain, this transaction is identified by the following transaction hash:

  0xc63dce6747e1640abd63ee63027c3352aed8cdb92b6a02ae25225666e171009e

- The details of this transaction can be visualized on the block explorer at https://etherscan.io/, as shown in the following screenshot:



Transaction Hash:	0xc63dce6747e1640abd63ee63027c3352aed8cdb92b6a02ae25225666e171009e
Status:	Success
Block:	4576084 4659657 Block Confirmations
Timestamp:	780 days 7 hrs ago (Nov-18-2017 01:25:54 PM +UTC)
From:	0x1ce3106fb372695bc2d35ec0ad1237c829f8d6dc
To:	0xefc7aef5150836955e9cea8bc360d57925e85093
Value:	0.015927244142974896 Ether ($2.29)
Transaction Fee:	0.000441 Ether ($0.06)
Gas Limit:	21,000
Gas Used by Transaction:	21,000 (100%)
Gas Price:	0.000000021 Ether (21 Gwei)
Nonce Position	1 4

Figure 11.5: Etherscan Ethereum blockchain block explorer

The Ethereum network :

is *a peer-to-peer network* where nodes participate in order to maintain the blockchain and contribute to the consensus mechanism.

Networks can be divided into three types:

1) **The mainnet**
2) **Testnets**
3) **Private nets**

The **mainnet:**

- It is the *current live network* of Ethereum**.**

- Its *network ID is 1 and its chain ID is also 1*. The network and chain IDs are used to identify the network.

- This can be *used to explore the Ethereum blockchain.*

.

Testnets

- The aim of these test blockchains is to *provide a testing environment for smart contracts and Dapps* before being deployed to the production live blockchain.

- They also allow *experimentation and research*.

- The *main testnet* is called ***Ropsten***, which contains all the features of other smaller and special-purpose testnets that were created for specific releases.

Private nets

- These are private networks that can be *created by generating a new genesis block*.

- a private group of entities start their blockchain network and use it as a permissioned or consortium blockchain.

Components of the Ethereum ecosystem

- At the core, there is the ***Ethereum blockchain*** running on the peer-to-peer Ethereum network.
- Secondly, there's an ***Ethereum client*** (usually Geth) that runs on the nodes and connects to the peer-to-peer Ethereum.
  - It provides various functions, such as <u>mining and account management.</u>
  - The local copy of the blockchain is synchronized regularly with the network.
- Another component is the ***web3.js library*** that allows interaction with the geth client via the *Remote Procedure Call (RPC) interface.*

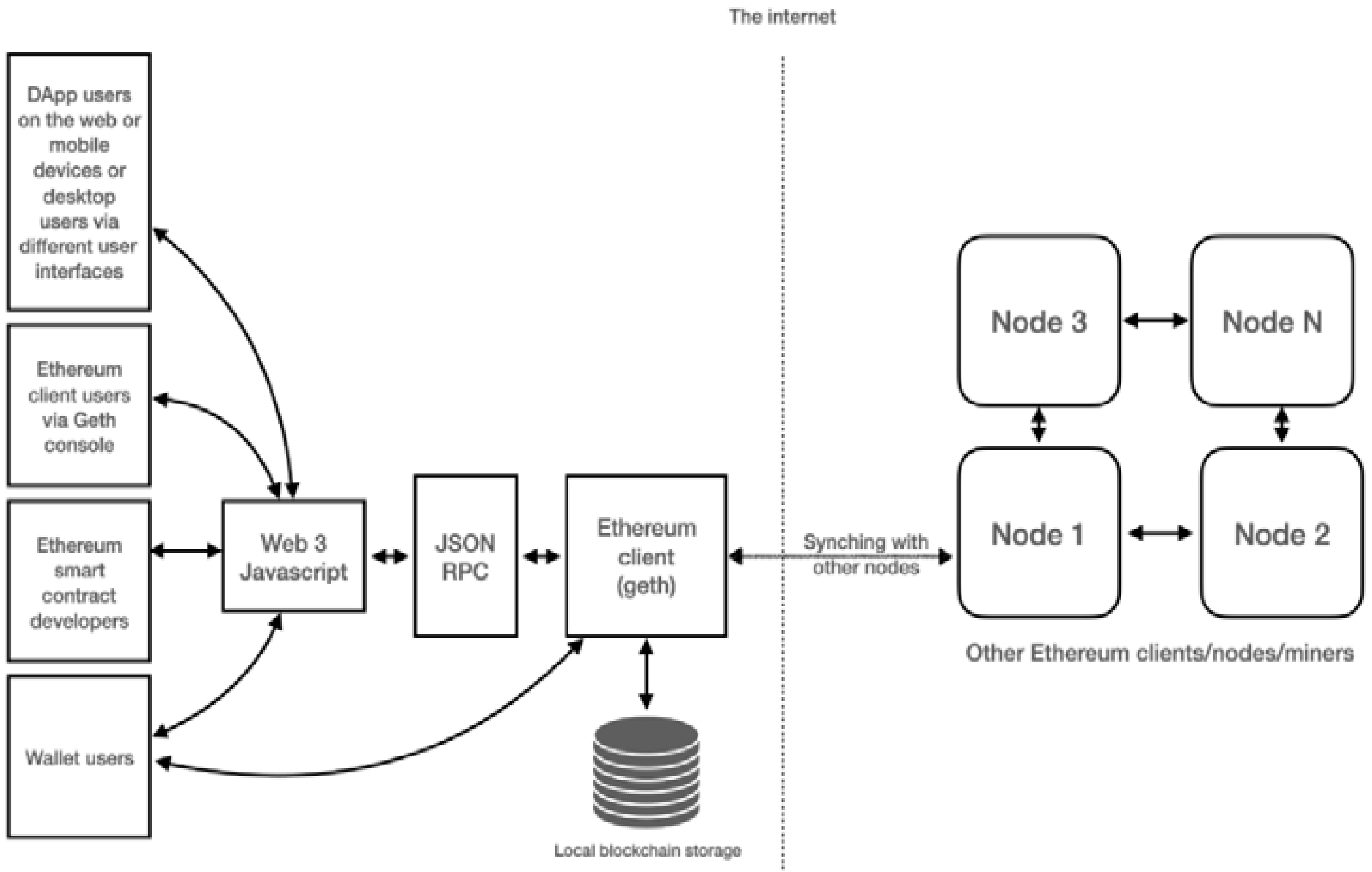

The internet
DApp users on the web or mobile devices or desktop users via different user interfaces
Ethereum client users via Geth console
Ethereum smart contract developers
Wallet users
Web 3 Javascript
JSON RPC
Ethereum client (geth)
Local blockchain storage
Synching with other nodes
Node 3
Node N
Node 1
Node 2
Other Ethereum clients/nodes/miners

Figure 11.6: Ethereum high-level ecosystem/ Ethereum Ecosystem Architecture

Elements of the Ethereum blockchain

1. Keys and addresses
2. Accounts
3. Transactions and messages
4. Ether cryptocurrency/tokens
5. The EVM
6. Smart contracts and native contracts

The Ethereum Virtual Machine (EVM)

- The EVM is a simple *stack-based execution machine* that *runs bytecode instructions to transform the system state* from one state to another.

    - ➢ The *word size* of the EVM is set to *256-bit*.
    - ➢ The *stack size* is limited to *1,024 elements* and is based on the Last In, First Out **(LIFO) queue.**

- The EVM is a *Turing-complete machine* but is limited by the amount of gas that is required to run any instruction.

- The EVM also *supports exception handling*.

There are three main types of storage available for contracts and the EVM:

- **Memory:**

➢ The first type is called memory or volatile memory, which is a word-addressed byte array.

➢ When a contract finishes its code execution, the memory is cleared.

➢ write operations to the memory can be of 8 or 256 bits, whereas read operations are limited to 256-bit words.

- **Storage:**

➢ The other type is called storage, which is a <u>key-value store</u> and is permanently persisted on the blockchain.

➢ Keys and values are each 256 bits wide.

➢ As a security measure, storage is only accessible by its own respective CAs.

- **Stack:**

➢ EVM is a stack-based machine, and thus <u>performs all computations in a data area</u> called the stack.

➢ It has a maximum depth of 1024 elements and supports the word size of 256 bits.

The following diagram shows the design of the EVM where:

- The virtual ROM stores the program code that is copied into the main memory using the CODECOPY instruction.
- The main memory is then read by the EVM by referring to the program counter and executes instructions step by step.
- The program counter and EVM stack are updated accordingly with each instruction execution.

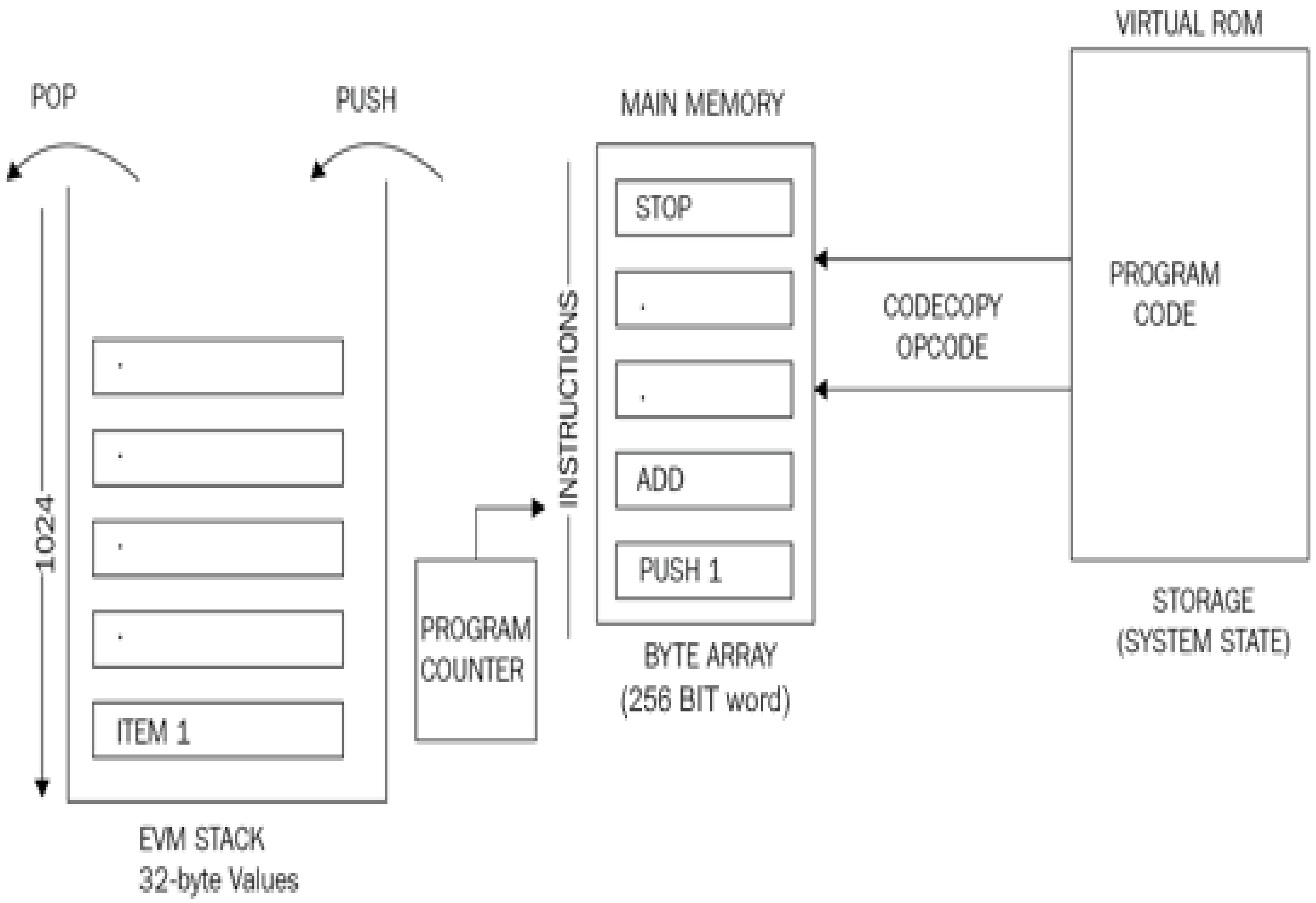


Figure 11.11: EVM operation

S**tandard fields in Ethereum transactions**:

1) **Nonce:**

   The nonce is a number that is incremented by one every time a transaction is sent by the sender.

- ➢ It must be equal to the number of transactions sent and is used as a unique identifier for the transaction.
- ➢ A nonce value can only be used once.
- ➢ This is used for replay protection on the network.

2) **Gas price:**

   The gas price field represents the amount of Wei required to execute the transaction.

   In other words, this is the ***amount of Wei you are willing to pay*** for this transaction.

(Wei is the smallest denomination of ether; therefore, it is used to count ether)

3) Gas limit:

The gas limit field contains the value that represents the maximum amount of gas that can be consumed to execute the transaction.

It is the fee amount, in ether, that a user (for example, the sender of the transaction) is ***willing to pay for computation***.

4) To:

As the name suggests, the To field is a value that represents the address of the recipient of the transaction.

This is a 20 byte value.

5) Value:

Value represents the total number of Wei to be transferred to the recipient.

6) Signature:

The signature is composed of three fields, namely V, R, and S.

- These values represent the ***digital signature (R, S)*** *and some information that can be used to recover* the ***public key (V****).*

7) Init:

This represents a byte array of unlimited length that specifies the EVM code to be used in the account initialization process.

- The code contained in this field is executed only once when the account is created for the first time, it (init) gets destroyed immediately after that.

8) **Data:**

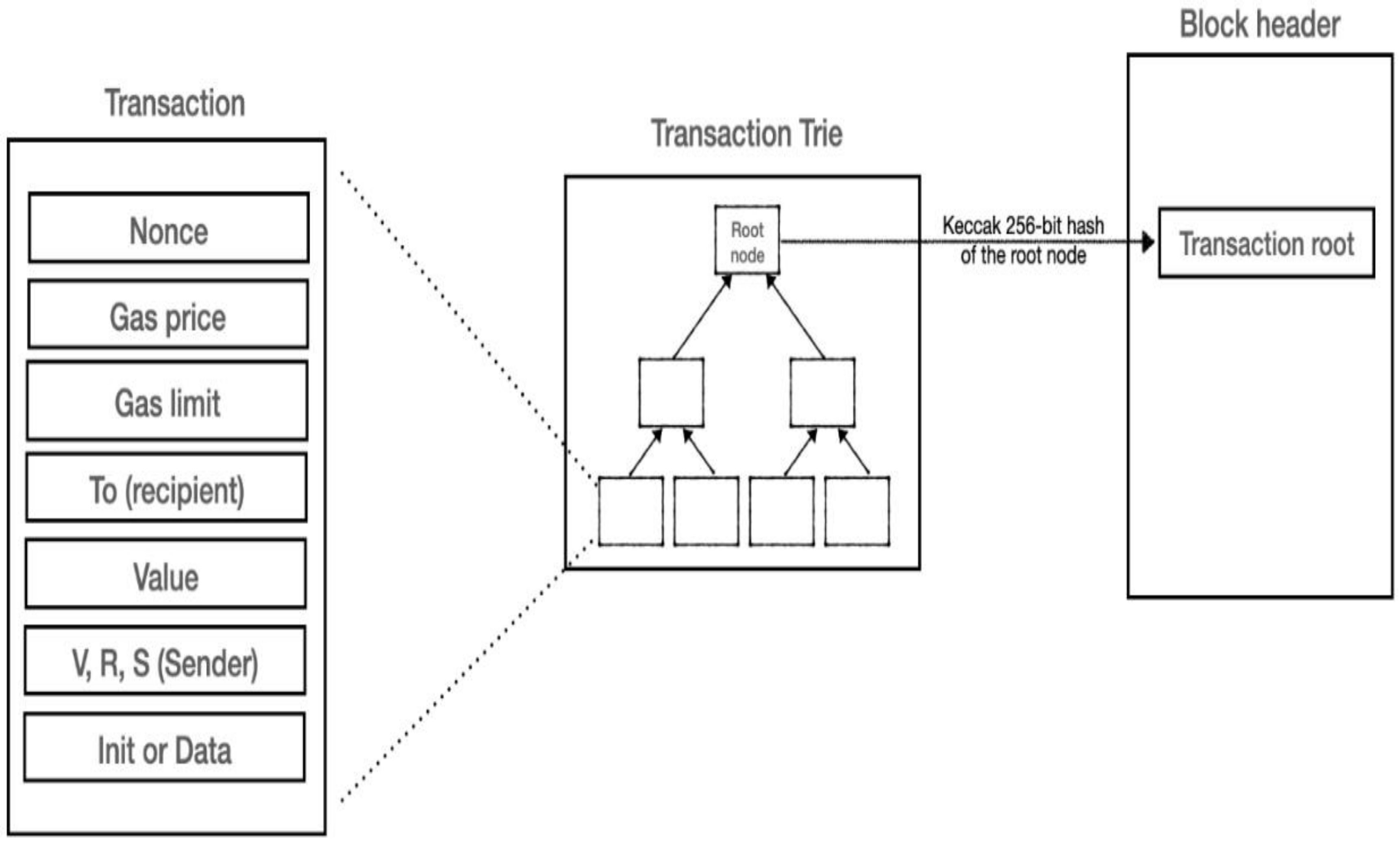
If the transaction is a message call, then the Data field is used instead of init, and represents the <u>input data of the message call.</u>

- ➢ It is also unlimited in size and is organized as a byte array.

- This structure is visualized in the following diagram, where a transaction is a tuple of the fields mentioned earlier, which is then included in a **transaction trie** (a modified **Merkle-Patricia tree(MPT))**

- Finally, the root node of the transaction trie is hashed using a Keccak 256-bit algorithm and is included in the block header along with a list of transactions in the block:

Figure 11.7: The relationship between the transaction, transaction trie, and block header



1. Contract Creation Transaction

It is used to create smart contracts on the blockchain.
The parameters required are :

- The sender
- The transaction originator
- Available gas
- Gas price
- Endowment, which is the amount of ether allocated
- A byte array of an arbitrary length
- Initialization EVM code
- The current depth of the message call/contract-creation stack (the number of items that are already present in the stack)

- the result of a contract creation transaction is either a new contract with its balance, or no new contract is created with no transfer of value.

2) Message call transactions

A message call requires several parameters for execution, as follows:

- The sender
- The transaction originator
- The recipient
- The account whose code is to be executed (recipient)
- Available gas
- The value
- The gas price
- An arbitrary-length byte array
- The input data of the call
- The current depth of the message call/contract creation stack

- A message call is the <u>act of passing a message from one account to another</u>.
- If the destination account has an associated EVM code, then the EVM will start upon the receipt of the message to perform the required operations.
- If the message sender is an autonomous object (external actor), then the call passes back any data returned from the EVM operation.