

Blockchain Basics  
Course 1 of the specialization

# Contents

Chapter 1	Page 2
1.1 Practitioner's Perspective John Wolpert, ConsenSys — 2	2
1.2 Bitcoin Blockchain	2
1.3 Blockchain Structure	3
1.4 Basic Operations	5
1.5 Beyond Bitcoin	7
1.6 Ethereum Blockchain: Smart Contracts Practitioner's Perspective: The Enterprise — 9	7
1.7 Ethereum Structure	9
1.8 Ethereum Operations	11
1.9 Incentive Model	11
1.10 Algorithms Techniques: Public-Key Cryptography	13
1.11 Hashing	15
1.12 Transaction Integrity	16
1.13 Securing Blockchain	17
1.14 Trust Essentials: Decentralized Systems	19
1.15 Consensus Protocol	20
1.16 Practitioner's Perspective: Decentralized Governance	21
1.17 Robustness	22
1.18 Practice Quiz	23
1.19 Forks	23
1.20 Practice Quiz	24

# Chapter 1

## 1.1 Practitioner's Perspective

### 1.1.1 John Wolpert, ConsenSys

Blockchain turned out to be a game-changer. We stumbled upon it while crossing the Mississippi and realized there was a wagon train behind us. Now, we're running a boomtown. However, most use cases in industry since 2015 that attempted to utilize blockchain have proven unnecessary. Those who approach problems with the lens of immutability, decentralization, and token economics have found value in public blockchain technology. It allows for gamifying human interaction and making economic decisions based on consensus. Public blockchain is like a precise instrument for solving problems in a cost-effective manner compared to traditional methods. It provides a new infrastructure that allows for innovative approaches and greater availability. With these new rails, we can do things completely differently.

## 1.2 Bitcoin Blockchain

Welcome to the first course of the Blockchain Specialization, titled "Blockchain Basics." Let's begin by exploring the concept of blockchain and why it is important. Unlike traditional methods, blockchain allows for the direct transfer of digital assets between peers without the need for intermediaries. Originally created to support the popular cryptocurrency Bitcoin, blockchain has since evolved into a powerful technology that has found applications in various industries such as finance, healthcare, government, manufacturing, and distribution.

The potential of blockchain extends far beyond its initial purpose. It can revolutionize numerous applications including supply chain management for goods transfer, art sales for digital media transfer, travel and tourism for remote services delivery, decentralized business logic platforms that move computing to data sources, and education credentialing for distributed intelligence. Additionally, blockchain can be utilized in areas such as power generation and distribution for distributed resources, startup fundraising through crowdfunding, electronic voting for crowd operations, identity management with a single ID for all life functions, and government public records and open governing.

Furthermore, blockchain has the ability to foster an inclusive economy by allowing individuals from remote corners of the world to participate in democratic processes. The possibilities for innovative applications are endless. Therefore, there is a pressing need for critical thinkers, designers, and developers who can envision and create new application models on blockchain to benefit society as a whole.

This course is designed to address the need for understanding blockchain technology. By the end of the course, you will have a solid understanding of the three key characteristics that define blockchain, using Bitcoin as an example. However, it's important to note that Bitcoin is not the only player in this field. In the first module, we will explore other next-generation blockchains like Ethereum and discuss their important features. Additionally, we will delve into the algorithms and techniques that enable blockchain technology, such as public key cryptography and hashing. Lastly, we will explore methods for establishing trust within a blockchain system. Let's start by introducing Bitcoin and its contributions to digital currency and decentralized applications through blockchain technology.

Our focus is on blockchain, but to fully appreciate its innovation, we must understand the technology



Figure 1.1: bitcoin

behind Bitcoin. The internet has transformed every aspect of our lives, and in 2008, a mysterious person introduced Bitcoin, a digital currency that allowed peer-to-peer transfers without a central authority. This was made possible through the implementation of software programs and a novel infrastructure called the blockchain. Over time, computation elements were added to the blockchain, opening up possibilities beyond currency transfer. Blockchain enables peer-to-peer transactions in a decentralized network, establishing trust among unknown peers and recording transactions in an immutable ledger. To illustrate this, let's consider a scenario where a customer wants to buy an item using her credit card.

In a centralized system, intermediaries such as credit card agencies, customer banks, credit cards banks, exchanges, merchant's banks, and merchants are involved in completing a task. However, in a decentralized system like blockchain, peers can transact directly with each other regardless of their location. The functions of intermediaries are shifted to the peer participants in the blockchain infrastructure. Trust is established among peers through a process that validates, verifies, and confirms transactions. These transactions are recorded in a distributed ledger of blocks that create a tamper-proof record. A consensus protocol is implemented to agree on adding blocks to the chain. Through validation, verification, consensus, and immutable recording, trust and security are achieved in the blockchain system.

### 1.3 Blockchain Structure

The blockchain structure consists of transactions that are validated and broadcasted. These transactions form blocks, which are connected through a digital data link. Miners, powerful computers executing blockchain software, validate and reach a consensus on which block to add next. This chosen block is then verified and added to the existing chain. In the Bitcoin network, a key concept is the Unspent Transaction Output (UTXO). The

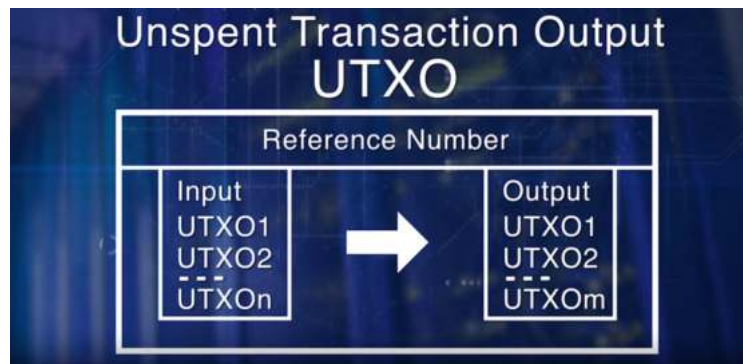


Figure 1.2: UTXO

collection of all UTXOs in a bitcoin network collectively defines the state of the Bitcoin Blockchain. UTXOs are utilized as inputs in transactions and also generated as outputs by transactions. These UTXOs are stored in a database by participant nodes within the system. Now, let's examine the role of UTXOs in a Bitcoin Blockchain. During a transaction, one or more UTXOs are used to transmit the specified amount to one or more newly created output UTXOs, as per the sender's request. The structure of a UTXO is straightforward, consisting of a unique

identifier for the transaction that created it, an index indicating its position in the transaction output list, a value denoting its amount, and an optional script specifying the spending condition for the output.



Figure 1.3: UTXO

The transaction itself includes a reference number for the current transaction, references to one or more input UTXOs, references to one or more output UTXOs newly generated by the current transaction, as well as the total input and output amounts. Participants can validate the transaction contents by checking if the referenced input UTXOs exist in the network state. This is just one of several validation criteria.

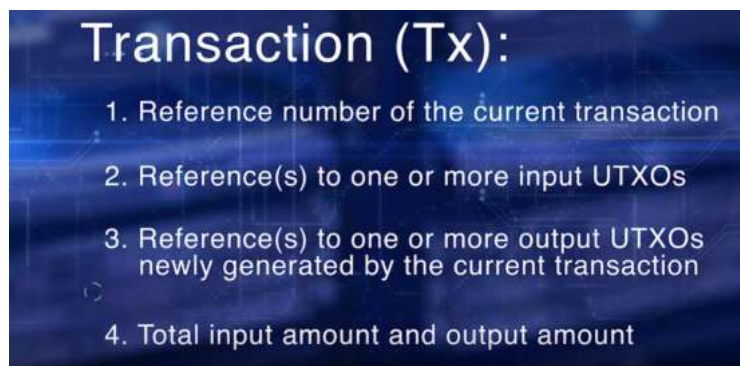


Figure 1.4: transaction

It is time to explore the real transaction. It is available at this link. At the top left is the transaction number. Just below, there are three input UTXOs referenced. And an arrow points to the references of the two output UTXOs. Note that there are three input UTXOs and only two output UTXOs. This means that the total amount in three input UTXOs are spent to generate two new output UTXOs. Just below the output UTXOs is a total amount transferred in the Bitcoin value. And that is shown in green. Let's now move on to learn about a block.

**Transaction** View information about a bitcoin transaction

22bb5dd2a88f9c5183df59aef20a44b8bce4a4056562b5d9cfff20a0ae2ffc1a

13PAdmSdMgRPG5GKQ3N1YH2CTPn6gWf 0.00041805 BTC  
 1NTh6Zap6Avk9pGAPVWw6U7FVJuz 0.001879 BTC  
 13nFvubeF72MGRmkngJPN1y7h76ZC 0.00438705 BTC

Summary		Inputs and Outputs	
Size	617 (bytes)	Total Input	0.00669305 BTC
Weight	2488	Total Output	0.00438705 BTC
Received Time	2017-11-05 21:27:38	Fee	0.002306 BTC
Included in Blocks	492043 (2017-11-05 21:27:38 + 5 minutes)	Fee per byte	430.661 sat/B
Confirmations	5210 Confirmations	Fee per weight unit	105.17 sat/WU

<https://blockchain.info/tx/22bb5dd2a88f9c5183df59aef20a44b8bce4a4056562b5d9cfff20a0ae2ffc1a>

As you can see, a block is composed of a header of information about the block, and a set of valid transaction.

At first, we'll examine the genesis block. It's available at this link. We introduce this block number zero for posterity's sake. This is where it all began.

Summary		Hashes	
Number Of Transactions	1	Hash	<a href="#">00</a>
Output Total	50 BTC	Previous Block	<a href="#">00</a>
Estimated Transaction Volume	0.00000000	Next Block(s)	<a href="#">00</a>
Transaction Fees	0.00000000	Merkle Root	<a href="#">00</a>
Height	1 (Genesis Chain)		
Timestamp	2009-01-03 18:15:05		
Received Time	2009-01-03 18:15:05		
Relayed By	<a href="#">00000000</a>		
Difficulty	1		
Size	488886750		
Size	0.00000000		
Weight	0.00000000		
Version	1		
Nonce	2083213845		
Block Reward	50 BTC		

Satoshi Nakamoto initiated the Bitcoin Blockchain with one transaction of 50 Bitcoins or 50 BTC. This one transaction created a UTXO output for half in his address. The data the Blockchain creation is January third, 2009. There was no previous block. This field is all zeros. It also lists a block reward or minus fees of 50 BTC. We'll discuss minus reward in our later lessons. Let's now look at the more recent block 482808. On the right panel, note the current block hash. The previous block hash that there's a link to the previous block. And on the left panel down here, the Nonce. You will learn about the block hash computation in the module quiz and in the later lessons. Now, let's look at the structure of the Blockchain by peeking into the Bitcoin Blockchain. Our goal is to understand the link between the blocks. Let's consider the chain of three blocks: 488867, 488868, 488869. 488868 shown in the middle, has the hash of 488867 as its previous hash. Block 488869 has the hash of 488868 as its previous hash, forming the links in the chain. To summarize, transactions bring about transfer of value in the Bitcoin Blockchain. The concept of UTXO defines the inputs and outputs of such a transaction. Once a block is verified algorithmically agreed upon by the miners, it is added to the chain of blocks, namely the Blockchain.

## 1.4 Basic Operations

Now that we have reviewed the basics of a blockchain and its basic structure and origin, let's consider the basic operations in a blockchain. Operations in the decentralized network are the responsibility of the peer participants and their respective computational nodes. For example, laptop, desktop, and server racks. These operations include validation transactions, gathering the transactions for a block, broadcasting the ballot transactions in the block, and consensus on the next block creation, and chaining the blocks to form an immutable record. In this lesson, we explore some of the fundamental operations of the bitcoin blockchain. First, we have to discuss

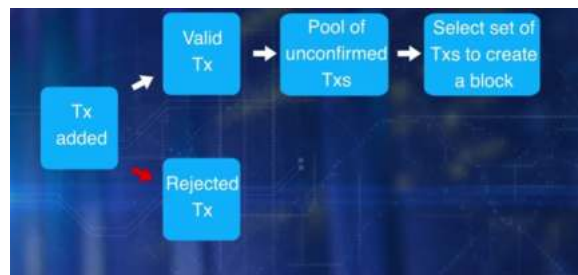


Figure 1.5: Operation

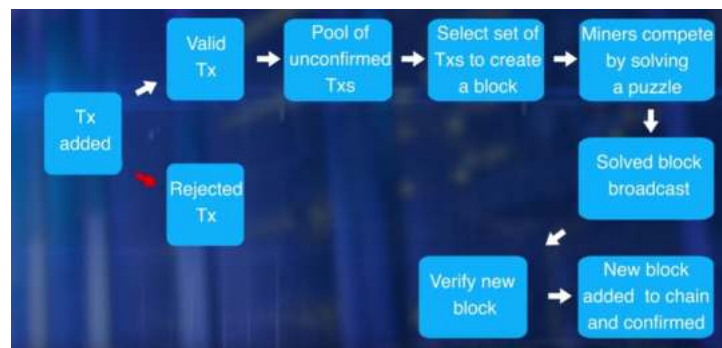
the participants. There are two major roles for the participants. Participants that initiate transfer of value by creating a transaction, additional participants called miners, who pick on added work or computation to verify transactions, broadcast transaction, compete to claim the right to create a block, work on reaching consensus by validating the block, broadcasting the newly created block, and confirming transactions. You might wonder why participant would take on additional work. Well, the miners are incentivised with bitcoins for the efforts



in managing the blockchain, as we'll find out. Transaction validation is carried out independently by all miners. The process involves validation of more than 20 criteria, including size, syntax, et cetera. Some of these criteria are: Referenced Input Unspent Transaction Output, UTXOs are valid, recall, UTXO is well-defined earlier in lesson two, reference output UTXOs are correct, reference input amount and output amount matched sufficiently, invalid transactions are rejected and will not be broadcast. All the valid transactions are added to a pool of transactions. Miners select a set of transaction from this pool to create a block. This creates a challenge. If



every miner adds the block to the chain, there will be many branches to the chain, resulting in inconsistent state. Recall, the blockchain is a single consistent linked chain of flux. We need a system to overcome this challenge, the solution. Miners compete to solving a puzzle to determine who earn the right to create the next block. In the case of bitcoin blockchain, this parcel is a computation of parcel and the central processing unit or CPU intensive. Once a miner solves the puzzle, the announcement is broadcast to the network and the block is also broadcast



to the network. Then, other participant verify the new block. Participants reach a consensus to add a new block to the chain. This new block is added to their local copy of the blockchain. Thus, a new set of transactions are recorded and confirmed. The algorithm for consensus is called proof-of-work protocol, since it involves work a computational power to solve the puzzle and to claim the right to form the next block. Transaction zero, index zero of the confirmed block is created by the miner of the block. It has a special UTXO and does not have any input UTXO. It is called the coinbase transaction that generates a minor's fees for the block creation. Currently, the minor's fees is 12.5 BTC for a bitcoin. This is how new coin is minted in bitcoin. To summarize, the main operations in a blockchain are transaction validation and block creation with the consensus of the participants. There are many underlying implicit operations as well in the bitcoin blockchain.



## 1.5 Beyond Bitcoin

Let's now look beyond Bitcoin. Bitcoin blockchain is open-source and the entire code is available on the GitHub. During the initial years beginning roughly in 2009, this open-source code was extended to release different cryptocurrencies. About 300 plus cryptocurrencies were introduced. Bitcoin supports an optional and special feature called scripts for conditional transfer of values. Ethereum Blockchain extended the scripting feature into a full-blown code execution framework called smart contract. A smart contract provide the very powerful capability of code execution for embedding business logic on the blockchain. Based on such capabilities, three major types of blockchains emerge from Bitcoin foundation. Type one deals with the coins in cryptocurrency currency chain. Example, Bitcoin. Type two supports cryptocurrency and a business logic layer supported by code execution. Example, ethereum. Type three involves no currency but supports software execution for business logic. Example, The Linux Foundation's Hyperledger. With the addition of code execution, comes the serious consideration about public access to the blockchain hence, the classification of public, private, and permissioned blockchains based on access limits. We have been watching Bitcoin blockchain continuously operational since its inception. All



supported by its public participants. Thus Bitcoin is a fantastic example of a public blockchain class. Anybody



can join and leave as they wish. Transaction blocks and the blockchain are publicly observable even though participants are anonymous. It is open-source. You can also create new coin digital currency by modifying the Bitcoin code. Wallet applications provide the basic interface to transfer value through the Bitcoin blockchain. In a private blockchain, access to the blockchain is limited to selected participants for example, those participants within an organization. This restriction helps in simplifying the normal operations such as block creation and contingency model. The third classification of blockchain is permissioned blockchain, also called consortium blockchain. It is meant for a consortium of collaborating parties to transact on a blockchain for ease of governance, provenance, and accountability for example, a consortium of all automobile companies or healthcare organizations. Permissioned blockchain has the benefits of a public blockchain with allowing only users with permission to collaborate and transact. In summary, significant innovations such as smart contracts have opened up broader applications for blockchain technology. Private and permissioned blockchain allow for controlled access to the blockchain enabling many diverse business models.

## 1.6 Ethereum Blockchain: Smart Contracts

Bitcoin blockchain is the mother of all blockchains. It was intended for peer to peer transfer of value and it does that well. Around 2013, a framework for code execution was introduced by Ethereum Founders. The centerpiece

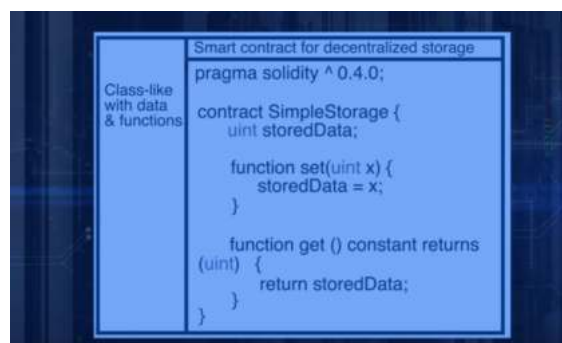


and thrust of this Ethereum blockchain is a smart contract. Consider this toggled diagram comparing Bitcoin and Ethereum blockchain. On the left is the Bitcoin blockchain and a wallet application for initiating transactions. On the right is Ethereum that took a significant step towards transforming the blockchain into a computational



framework that opened up a whole world of opportunities in the decentralized realm. Ethereum supports smart contracts and of virtual machine on which smart contracts execute. Smart contracts in turn enable decentralized application that accomplish more than a transfer of value. Efficient automation of decentralized application such as supply chain. Smart contract is an important topic that we have allocated a full course for this topic in this specialization. In this module, of course, one, we provide a high level overview of smart contract as it relates to Ethereum. After completing this module on Ethereum, you will be able to discuss at a high level the innovation of the ethereum blockchain, namely the smart contract. Illustrate ethereum blockchain protocol, structural elements, and operational aspects. Demonstrate the concept of gas, the fuel or the payment model for code execution and the incentive model for the Ethereum blockchain.

What is a smart contract? Let’s review the basics. A smart contract is a piece of code deployed in the blockchain node. Execution of a smart contract is initiated by a message embedded in the transaction. Digital currency transfer request simple addition and subtraction. Ethereum enables transaction that may carry out more sophisticated operations. For example, a transaction could require a conditional transfer, it may require some evaluation, it may need more than one signature for transfer of assets, or it may involve waiting for a specific time or date. Let’s consider an example of what a smart contract can do. An auction bidding smart contract could execute this logic. If the age of a bidder is greater than 18 and the bid is greater than the minimum bid, then, accept the bid, or else reject the bid. This can be done by a smart contract. What does the smart contract look like? How do you write a smart contract? Structurally, a smart contract resembles a class definition in an object oriented design. It has data, functions or methods with modifiers public or private, along with getter and set of functions. Specific programming languages have been designed for coding smart contracts. Solidity is one such language. Let’s examine a simple Solidity smart contract to understand its structure. First line with



pragma indicates the version of the solidity language. The contract’s name is in the first line. This particular contract is for one integer storage. The data for the integer is defined with type a name, uint StoredData. Two functions are defined for writing and reading the data. Set and get. Code execution. Where does the code in the smart contract get executed? Where is it located in a node? We need a computational infrastructure to execute any arbitrary code. Every node in Ethereum network should be able to execute the code irrespective of that underlying type of hardware or operating system. Enter Ethereum Virtual Machine, EVM. An EVM provides a run anywhere obstruction layer for the contract code. A smart contract written a high level programming language is translated into EVM byte code, and then, deployed on the Ethereum Virtual Machine, EVM. Every node will host the same smart contract codes on the EVM. Summarizing, smart contracts add a layer of logic

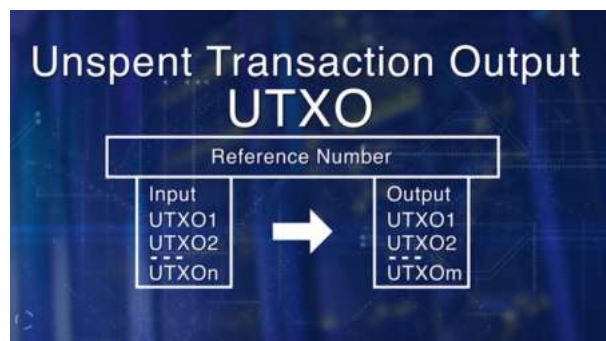
and computation to the trust infrastructure supported by the blockchain. Smart contracts allow for execution of code. Enhancing the basic value transfer capability of the Bitcoin blockchain. The code for this smart contract is written in a high level language like Solidity and compiled into byte code. The code for the smart contracts is executed on a special structure known as Ethereum Virtual Machine.

### 1.6.1 Practitioner's Perspective: The Enterprise

Ethereum has worked out marvelously on how to solve these problems of scaling and privacy and a number of other issues that needed to be worked out for public networks. You have the Enterprise Ethereum Alliance that is the largest organization of entities working on the Ethereum protocol and maturing it into something that industry can use. But at the same time, companies around the Hyperledger project, Fabric and Iroha and Sawtooth and all these other protocol initiatives under the Linux Foundation and what they call the Hyperledger project. That has led to a whole bunch of other breakthroughs around channels and contracts segregation. So you remember I was saying, you and I have an agreement and we need to all be in control but not anyone having the control system. All right, well that's fine. But if you and I have a confidential agreement and nearly all business to business agreements are confidential. It says right on the contract it says, "not only is this contract confidential between you and me, the existence of it is confidential." Most of the pharmaceutical industry operates on this. The medical industry operate on this principle for good and bad. You could say, "Well we'd like some more transparency there." But we also don't want your medical records out in the clear, and we don't want the fact that you're even paying for a particular drug to be out in the clear. Right? Or that you're using a particular drug. That's not Okay. So, we need transparency for some things and opacity for other things and getting that mix right is really hard. So, we need things like channels and charting and now in Ethereum plasma and what we call Layer two technologies that are coming out this year and next that really will transform our concept of block chain.

## 1.7 Ethereum Structure

Bitcoin blocking state was defined in terms of unspent transaction outputs UTXOs and a reference implementation of the Wallet application that held the account reference. Ethereum formally introduce the concept of an account



as a part of the protocol. The account is the originator and the target of a transaction. A transaction directly updates the account balances as opposed to maintaining the state such as in the bitcoin UTXOs. It allows for transmit of value and messages and data between the accounts that may result in the state transitions. These transfers are implemented using transactions. There are two types of accounts, Externally Owned Accounts and Contract Accounts. Externally Owned Accounts or EOA are controlled by private keys. Contract Accounts or CA are controlled by the code and can be activated only by an EOA. An externally owned account is needed to participate in the Ethereum network. It interacts with the blockchain using transactions. A Contract Account represents a smart contract. Every account has a coin balance. The participant node can send transaction for Ether transfer or it can send transaction to invoke a smart contract code or both. Both types of transaction require fees. An account must have sufficient balance to meet the fees needed for the transactions activated. Fees are paid in Wei. Wei is a lower denomination of Ether. One Ether 10 to the power of 18 Weis. A transaction in Ethereum includes the recipient of the message, digital signature of the sender authorizing the transfer, amount of Wei to transfer, an optional data field or payload that contains a message to a contract, STARTGAS which is a value representing the maximum number of computational steps the transaction is allowed. Gas price a value



Block #4446308		Home / Blocks / Block Information
Overview		
Block Information		
Height:	< Prev 4446308 Next >	
TimeStamp:	86 days 5 hrs ago (Oct-28-2017 03:43:32 PM +UTC)	
Transactions:	127 transactions and 28 contract internal transactions in this block	
Hash:	0x1ebcad532646a0c27a3d438f151de182acc8c600148c98b5d215396c954e	
Parent Hash:	0x573cbe8c7b5a95d317988a98075867732888e5a96b121e07c134358ac3d97	
Sha3Uncles:	0x100c4d6bdec75d7a8b8b567b6cc041ad3124518948a7413f0a142b40d48347	
Mined By:	0x8296c824c1632e401c083c3d362283333a030 (2post_2) in 4 secs	
Difficulty:	1,457,769,638,180,738	
Total Difficulty:	1,308,077,364,782,154,049,967	
Size:	28635 bytes	
Gas Used:	6,894,263 (99.63%)	
Gas Limit:	6,718,946	
Nonce:	0x880447c80127aaf00	

## 1.8 Ethereum Operations

For a simple Ether transfer, the amount to transfer and the target address are specified along with the fees or gas points. The amount and the fees are transferred to their respective accounts. Here, we illustrate a transaction of 100 Ether transfer between the sender's and receiver's accounts. Note that besides this transfer, 21,000 gas



points are paid to the miner who added the transaction block to the blockchain. As discussed in a separate lesson, an Ethereum node is a computational system representing a business entity or an individual participant. An Ethereum full node hosts the software needed for transaction initiation, validation, mining, block creation, smart contract execution and the Ethereum Virtual Machine, EVM. This figure depicts the deployment of a smart contract and the invocation for a smart contract. Smart contract is designed, developed, compiled and deployed on the EVM that can be more than one smart contract in an EVM. When the target address in a transaction is a smart contract, the execution code corresponding to the smart contract is activated and executed on the EVM. The input needed for this execution is extracted from the payload field of the transaction. Current state of the smart contract is the values of the variables defined in it. The state of the smart contract may be updated by this execution. Results of this execution is told in the receipts. A blockchain maintains both the state hash and the receipt hash. We'll elaborate these in a later course on smart contract. All the transactions generated are validated. Transaction validation involves checking the time-stamp and the nonce combination to be valid and the availability of sufficient fees for execution. Miner nodes in the network receive, verify, gather and execute transactions. The in-work smart contract code are executed by all miners. Validated transactions are broadcast and gathered for block creation. The consensus protocol used is a memory-based rather than a CPU-based proof of work. Who pays for all these operations; validation, verification and consensus? In the next lesson, we'll explore the Incentive Model that answers this question.

## 1.9 Incentive Model

Interactive Transcript - Enable basic transcript mode by pressing the escape key You may navigate through the transcript using tab. To save a note for a section of text press CTRL + S. To expand your selection you may use



CTRL + arrow key. You may contract your selection using shift + CTRL + arrow key. For screen readers that are incompatible with using arrow keys for shortcuts, you can replace them with the H J K L keys. Some screen readers may require using CTRL in conjunction with the alt key

Module two, lesson four, the incentive model. As we learned earlier in module one, mining is the process used to secure the network by validating the computations, collecting them to form a block, verifying them, and broadcasting it.


Ethereum also uses a incentive-based model for block creation.

In this lesson we'll explore some relevant concept about fee structure and the incentive model.

Every action in Ethereum requires crypto fuel, or gas. Gas points are used to specify the fees inside of Ether, for ease of computation using standard values.

Gas points allow for cryptocurrency independent valuation of the transaction fee and computation fees.

Ether, as a cryptocurrency, varies in value with market swings, but gas points do not vary. Ethereum has



Operation name	Gas Cost
Step	1
Load from memory	20
Store into memory	100
Transaction base fee	21000
Contract creation	53000
...	...

specified gas points for each type of operation. Mining process computes gas points required for execution of a transaction.

If the fee specified and the gas point in the transaction are not sufficient, it is rejected. This is similar to mailing a letter with insufficient postage. The letter will not be delivered if it had insufficient postage. The gas points needed for execution must be in the account balance for the execution to happen. If there is any amount left over after the execution of a transaction, it is returned to the originating account.

So far we looked at the gas related items in a transaction, now let's look at the gas related items in a block. Gas limit and gas spent. Gas limit is the amount of gas points available for a block to spend. For example, if a block specifies a limit of 1 million 5 hundred thousand units of gas, and a basic Ether transaction fee is 21,000, this particular Ethereum block can fit about 70 plain Ether transactions. If we add smart contract transactions also into this block, that usually requires more gas, and the number of transactions for this block will likely be lower.

Gas spent is the actual amount of gas spent at the completion of the block creation. Now let's look at the mining incentive model. The proof of work puzzle winner, miner that creates a new block, is incentivized with the base fees of three Ethers, and the transaction fees in Ethereum blockchain.

The winning miner also gets the fees, gas points for execution of a smart contract transactions.

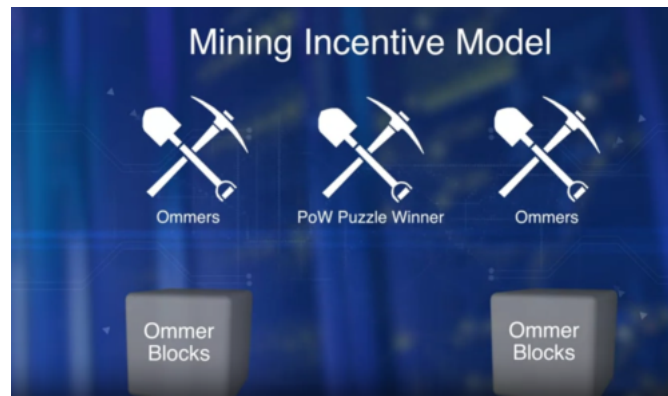
That there may be other miners who also solve the puzzle besides the winner.

These miners will solve the puzzle, but didn't win the block are called Ommers. The blocks created by





them are called Ommer Blocks. These are added as Ommer Blocks, or side blocks, to the main chain. Ommer



miners also get a small percentage of the total gas points as a consolation and for network security. Summarizing, any transaction in Ethereum, including transfer of Ethers, requires fees or gas points to be specified in the transactions. Miners are paid fees for security, validation, execution of smart contract, as well as for creation of blocks. We presented a high level view of Ethereum blockchain. We'll use Ethereum as a reference blockchain in the next two modules and the following two courses.

## 1.10 Algorithms Techniques: Public-Key Cryptography

Two techniques are predominantly used for securing the chain and for efficient validation and verification. Hashing and asymmetric key encryption. These techniques depend on several complex proven algorithms. We provide a

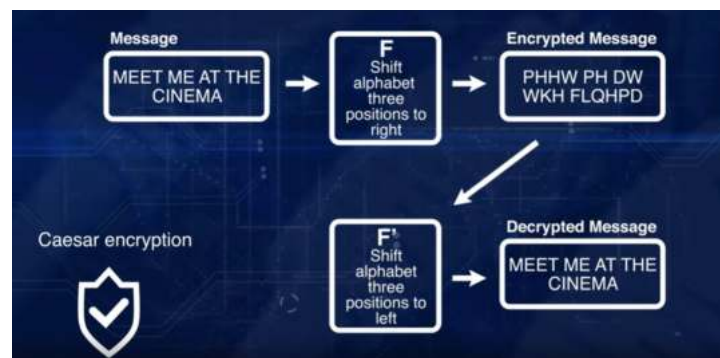


high level view of the application of these algorithms, and the critical role they play in securing the decentralized chain in the four lessons of this module namely; Public-key cryptography, secure hashing, transaction integrity, and block integrity. We'll begin this module by discussing the concept of asymmetric key encryption, then we'll define the concept of hashing, followed by the algorithms used for various hashing needs of the block chain protocol. We then explain the techniques that use these algorithms, to manage the integrity of the transactions and the blocks in a block chain. Learning outcomes of this module are; to summarize the working of the Public-key cryptography,

explain simple hashing and Merkle tree hashing, explore the application of hashing and cryptography in protecting the blockchain.



Recall that blockchains decentralized network participants, are not necessarily known to each other. Credentials cannot be checked by the conventional means such as verifying who you are with your driver's license. Participants can join and leave the chain as they wish. They operate beyond the boundaries of trust. Given this context. how do you identify the peer participants? How do you authorize and authenticate the transactions? How do you detect forged or faulty transactions? We can do these things by using Public-key cryptography algorithm that we'll discuss in this lesson. Let's begin by examining simple symmetric key encryption. The same key is used for encryption and decryption, so it is called symmetric key. Example, Ceasar encryption is the simplest one with alphabets of a message are shifted by a fixed number, and this number is called the Key. In this example



F is a function defined by shift by three in alphabet. Consider, "Meet me at the cinema." You shift by three the S key value of every letter to encrypt it, and your receiver decrypts it using the same three as the key. Shift the other way every character to view the original message. Three is the key in this trivial example. Since the same key is used for encryption and decryption, it is a symmetric key. Note that the key and the encryption and decryption functions are typically much more complex in a real application. Note that symmetric key encryption has issues. Number one, it is easy to derive the secret key from the encrypted data. And number two, the key distribution, how do you pass the key to the participant transacting? These issues are further exasperated in a block chain decentralized network where participants are unknown to each other. Let's now examine how Public-key cryptography addresses these issues. Instead of a single secret key, it employs two different keys that take care of both the issues of symmetric key encryption. Let, lowercase b uppercase B be the private public-key pair for a participant in Buffalo New York USA. Let lowercase k and uppercase K be the pair of keys for the participant and Kathmandu Nepal. Public-key is published, private key is kept safe and locked. Typically using a passphrase and the pair works as follows; encrypting function holds two properties with a key pair. The public-key private key pair has the unique quality that even though a data is encrypted with the private key, it can be decrypted with the corresponding public-key and vice versa. Now let's look at an example, authenticate the sender and the receiver. We'll examine just one common use of a symmetric key encryption. Let's say a participant in Buffalo wants to transact with the participant in Kathmandu. Instead of sending just a simple message, a participant in Buffalo will send a transaction data encrypted by Buffalo's private key, and then encrypted by Kathmandu's public key. Kathmandu will first decrypt the data using its own private key, then use Buffalo's public key to decrypt assigned transaction data. This ensures that only Kathmandu can decrypt and receive the data and that only Buffalo could have sent the data. A popular implementation of public key, private key is the Rivest Shamir

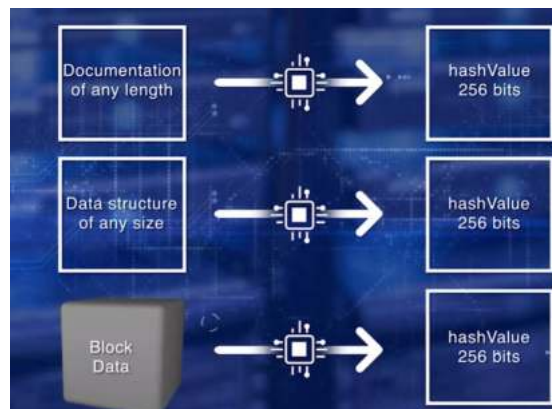




Adleman (RSA) algorithm. Common application of RSA is the passwordless user authentication, for example for accessing a virtual machine on Amazon cloud. Though RSA is very commonly used in many applications, block chains need a more efficient and stronger algorithm. Efficiency is a critical requirement since public key pair is frequently used in many different operations in block chain protocol. Elliptic Curve Cryptography, ECC family of algorithms is used in the bitcoin as well as an Ethereum block chain for generating the key pair. Why ECC not RSA? ECC is stronger than RSA for a given number of bits. Did you know that 256 bit ECC key pair is equal in strength to about 3072 bits of RSA key pair. Both bitcoin and Ethereum use ECC based algorithms for their encryption needs.

## 1.11 Hashing

Why should we learn about encryption hashing, you may ask. The private public key pair is a metaphorical passport to participating in transacting on the blockchain. Similar to how you learn to use a credit card, secure it and protect it. You need to protect the private key for the security of your assets on the blockchain. In this lesson, we learn hashing that plays a critical role in the blockchain process, and also in the integrity of the transaction and confidentiality of data. You'll keep hearing the words hash rate, hash power, hash this, hash that frequently in the blockchain world. For these reasons, you ought to have some basic understanding of hashing techniques. The primary goal of this lesson is to provide you with this knowledge. What is hashing? A hash function or hashing transforms and maps an arbitrary length of input data value to a unique fixed length value. Input data can be a document, tree data, or a block data. Even a slight difference in the input data would produce a totally different hash output value. The following are two basic requirements of a hash function. The algorithm chosen for the



hash function should be a one-way function and it should be collision free, or exhibit extremely low probability of collision. The first requirement is to make certain that no one can derive the original items hashed from the hash value. Can you make potatoes out of mashed potatoes? The second requirement is to make sure that the hash value uniquely represents the original items hashed. There should be extremely low probability that two different datasets map onto the same hash value. These requirements are achieved by choosing a strong algorithm such as secure hash, and by using appropriately large number of bits in the hash value. Most common hash size now is 256 bits and the common functions are SHA-3, SHA-256 and Keccak. Hash value space, how good is 256 bits hash? A 256-bit hash value space is indeed very large. 2 to the power of 256 possible combinations of values.

That is approximately 10 to the power of 77. That is 10 followed by 77 zeros. Odds of a meteor hitting your house is higher than generating two of the same hash values of 256 bits when applying this algorithm. Great. Let's proceed to explore some techniques now. We'll compare two different approaches for hashing based on how the constituent elements are organized. A simple hash and a Merkle tree hash. Here we illustrate simple hashing and Merkle tree hashing with ADD as a hash function. We use the data 10, 4, 6, 21 and 19 and ADD as a hash function. Actual hashing functions are quite complex and are variations of SHA-3, and the data values are much



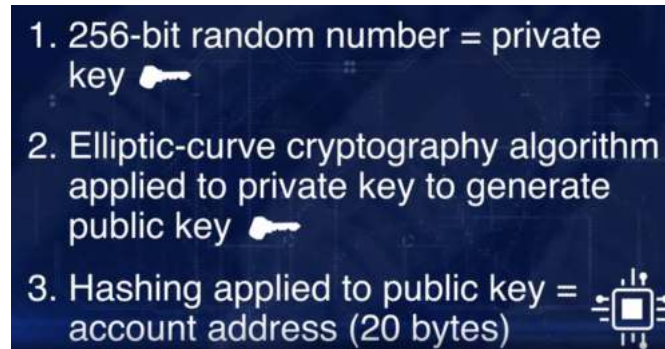
larger, mainly 256 to 512 bit values. In the simple hash approach, all the data items are linearly arranged and hashed. In a tree-structured approach, the data is at the leaf nodes of the tree, leaves are pairwise hash to arrive at the same hash value as a simple hash. When is a tree-structured hash used? When is a simple hash used? When we have a fixed number of items to be hashed, such as the items in a block header, and we are verifying the composite block integrity and not the individual item integrity, we use simple hash. When the number of items differ from block to block, for example, number of transactions, number of states, number of receipts, we use the tree structure for computing the hash. Note that the state is a variable that may be modified by a smart



contract execution, and the result of the execution may be returned in a receipt. We'll learn more about their use in course two. Tree structure helps the efficiency of repeated operations, such as transaction modification and the state changes from one block to the next. Log N versus N. Summarizing, in Ethereum, hashing functions are used for generating account addresses, digital signatures, transaction hash, state hash, receipt hash and block header hash. SHA-3, SHA-256, Keccak-256 are some of the algorithms commonly used by hash generation in blockchains.

## 1.12 Transaction Integrity

To manage the integrity of a transaction we need number one, secure a unique account address. We need a standard approach to uniquely identify the participants in the decentralized network. Number two, authorization of the transaction by the sender through digital signing. And number three, verification that the content of that transaction is not modified. We use a combination of hashing and public key cryptography, that we learned in the last two lessons to solve these problems. Let's start with the address of the accounts. Addresses of accounts are generated using public key, private key pair. Step 1, a 256-bit random number is generated, and designated as the private key. Kept secure and locked using a passphrase. Step 2, an ECC algorithm is applied to the private key, to get a unique public key. This is the private public key pair. Step 3. Then a hashing function is applied to the public key to obtain account address. The address is shorter in size, only 20 bytes or 160 bits. Now that we have the account address, let's look at the transaction initiated by this address. A transaction for transferring assets will have to be authorized, it has to be non-repudiable, and unmodifiable. They first examined, the digital signing process, and then apply it to that transaction. Data is hashed and encrypted. This is the digital signature. The receiver gets the original data, and the secure hash digitally signed. Receiver can recompute the hash of



the original data received, and compare it with the received hash to verify the integrity of the document. Now, consider the transaction to be that data. Step number 1, find the hash of the data fields of the transaction. Step number 2, encrypt that hash using the private key of the participant originating the transaction. Thus, digitally signing the transaction to authorize and making the transaction non-repudiable. Step number 3, this hash just added to the transaction. It can be verified by others decrypting it using the public key of the sender of the transaction, and recomputing the hash of the transaction. Then, compare the computed hash, and the hash received at the digital signature. If that is a match, accept the transaction. Otherwise, reject it. Note that



for the complete transaction verification, the timestamp, nons, account balances, and sufficiency of fees are also verified.

## 1.13 Securing Blockchain

As we discussed before, some of the main components of the Ethereum block are the header, the transactions, including the transaction hash or the transaction root, and the state root, the state hash, or the state root. Integrity of the block is managed by assuring that the block header contents are not tampered with, the transactions are not tampered with, state transitions are efficiently computed, hashed, and verified. Remember, the block chain is supposed to be an immutable record. In Ethereum, the block hash is the block of all the elements in the block header, including the transaction root and state root hashes. It is computed by applying a



variant of SHA-3 algorithm called Keccak and all the items of the block header. You would see a real example of bitcoin block hash in the quiz associated with this module. A typical block has about 2,000 transactions in bitcoin and about 100 transaction Ethereum. We need an efficient way to detect tampering and validate the transaction efficiently. Hashes of transaction in a block are processed in a tree structure called Mekle tree hash that we discussed in earlier lesson.



Merkle tree hash is also used for computing the state root hash, since only the hash of the chained states from block to block have to be re-computed. It is also used for receipt hash root. Remember the advantage over flat versus tree representation. If any transaction is to be verified, only one path to the tree has to be checked. You don't have to go through the entire set of transactions. Smart contract execution in Ethereum results in state transitions. Every state change requires state root hash re-computation. Instead of computing hash for the entire set of states, only the affected path in the Merkle tree needs to be re-computed. When the state 19 is changed to 20, that results in the path including 31, 41, and the state root hash 64 to be re-computed. Only that path is re-computed, not the entire tree. Now, let's move on to block hash computation. Block hash in Ethereum is computed by first computing the state root hash, transaction root hash and then receipt root hash, shown at the bottom of the block header. These roots and all the other items in the header are hash together with the variable nodes to solve the proof of work puzzle. Block hash serves two important purposes; verification of the integrity of



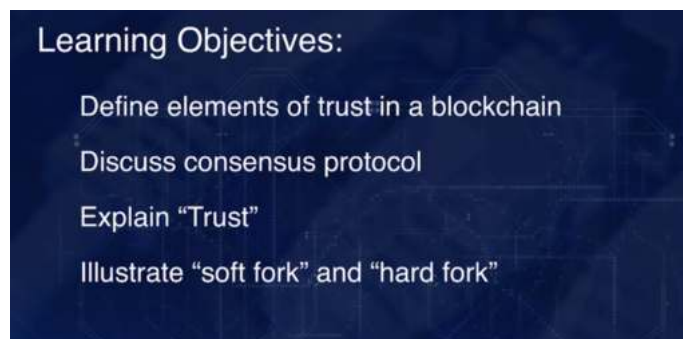
the block and the transactions, formation of the chain link by embedding the previous block hash in the current block header. If any participant node tampers with the block, its hash value changes resulting in the mismatch of the hash values and rendering the local chain of the node in an invalid state. Any future blocks initiated by the node would be rejected by other miners due to hash mismatch. This enforces the immutability of the chain. Summarizing, a combination of hashing and encryption are used for securing the various elements of the block



chain. Private public key pair and hashing are important foundational concepts in decentralized networks that operate beyond trust boundaries.

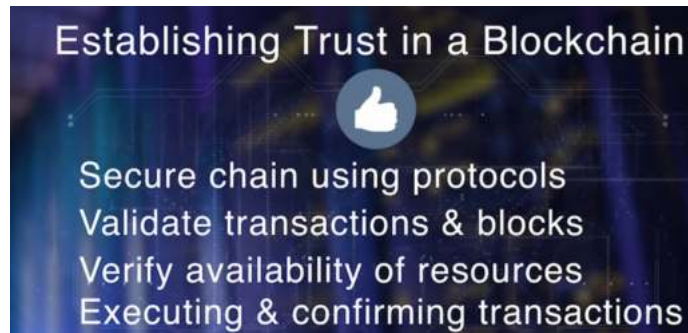
## 1.14 Trust Essentials: Decentralized Systems

In the last module, we learned about the use of public key cryptography and hashing for validation of transactions and blocks. After this module, you will be able to define elements of trust in a blockchain, security, validation, verification, and consensus; discuss consensus protocol, an algorithmic approach to add a new block and to secure the chain; explain trust and the robustness of the main chain; illustrate trust in managing exceptional situations such as hard fork and soft fork.



Let us understand trust using a common everyday scenario in a centralized system such as an airport system. Say, you want to fly out of the Buffalo airport. The airport authority would have pre-established a secure environment for people to arrive and depart. This establishes the base trust. Then there is additional trust once you enter and your passport and travel documents are verified, validated, and your baggage is screened. Even more trust in you is established when the airline staff checks your boarding pass at the gate and you enter the aircraft to fly. Now, let's consider a decentralized system. There is nobody checking your credentials and certifying that you are trustworthy. Then, how do you do it? You do it by using algorithms and techniques discussed in the last module. Let's examine how these will help address the trust issues in a blockchain. Similar to our airport scenario, trust in a decentralized blockchain is also about securing, validating, verifying, and making sure resources needed for transaction execution are available. This is accomplished by securing the chain using specific protocols, validating the transaction and blocks for tamper proofing, verifying the availability of resources for transactions, and executing and confirming the transactions.

The Trust Trail is defined by these operations: validate transaction, verify gas and resources, gather transactions, execute transaction to get a new state, form the block, work towards consensus, finalize the block



by the bidder, and everyone add the block to their chain and confirm the transactions. We will examine each of these steps. Steps one and two are validate transaction and check resources.

In the case of a Bitcoin, there are about 20 criteria that have to be checked before a transaction is validated, as discussed in module one, similarly in the case of Ethereum transaction. The syntax, the transaction signature, time stamp, nonce, gas limit, and sender account balance are validated before execution. The fuel, or gas points, and other resources available for smart contract execution, are also validated. Transaction signatures and hash are also verified. Step number three is execute transactions. Merkle tree hash of the validated transactions is computed. This is in Ethereum. This is the transaction root of the block header. All miners execute the transaction for either transfer, as well as for execution of smart contracts.



The state resulting from transaction execution are used in computing the Merkle tree hash of the states, the state root of the block header. The receipt root of the block header is also computed. In the next lesson, we'll continue to talk about the next step in trust trail, the consensus process.

## 1.15 Consensus Protocol

A secure chain is a single main chain with a consistent state. Every valid block added to this chain, adds to the trust level of the chain. The miners are vying, are competing to add their block to the chain. What if everyone

1. Trust in a decentralized blockchain is about \_\_\_\_\_. 1 point
  - ☐ securing the chain using specific protocols.
  - ☐ validating the transactions and blocks for tamper proofing.
  - ☐ executing and confirming the transactions.
  - ☐ All of the above
2. Miners execute the transactions for Ether transfers but are not responsible for the execution of smart contracts. True or False? 1 point
  - ☐ False
  - ☐ True

wants to add their candidate block to the chain? Each of the candidate blocks is by a competing miner. Which is the next block to be added to the chain? Can they agree on the next block? Is there a method or a protocol to choose the next block? Yes, there is. It is called Proof of Work.



Proof of Work uses hashing. Here is one more application of hashing. Do you realize now how versatile hashing is? We will now discuss Proof of Work as used in bitcoin and ethereum. This is from the point of view of the miner. First, compute the hash of the block header elements that is a fixed value, and a nonce that is a variable. If hash value is less than 2 par 128 for bitcoin, and less than function of difficulty for ethereum, the puzzle has been solved. If it has not been solved, repeat the process after changing the nonce value. If the puzzle has been solved, broadcast the winning block that will be verified by other miners. Non-winning miner nodes add the new block to the local copy of the chain, and move on to working on the next block. The winner gets an incentive for creating the block. Proof of Work is a consensus protocol used by bitcoin block chain and also by the current version of ethereum. The protocol may be the same, the implementations in these two block chains are different. Many other approaches such as Proof of Stake, Proof of Elapsed Time have been proposed.



This is a hotly debated area among the developers of blockchain, and you can contribute to it.

## 1.16 Practitioner's Perspective: Decentralized Governance

Something this important needs to be built by all of us for all of us. You can't just have a small group of people or one kind of person building this. It's too important. Even the Internet itself is not as is as pervasively disruptive and transformative as the Internet on blockchain is going to be, because the blockchain or the Internet was and is really an ideal system for broadcasting information. That was great. That was Guttenberg's best day. But it didn't invent a way of doing transactions without having centralized authorities in there, like banks and others



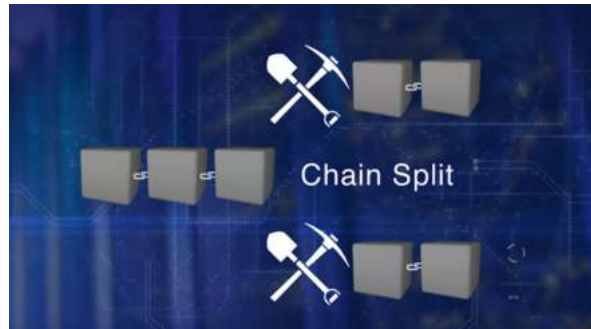
that have issues or like we are seeing with Facebook today. There are problems with central authorities being in control of your identity, or your information, or your transactions. But being able to democratize that and spread it all over the world, that's powerful, but it's also really powerful, which means it better be built by more than just a few knuckleheads in a building with only one perspective on things. So, it's very important that, A, these technologies are being built in open source, but not just in open source, openly governed open source, meritocracies. There are hundreds of people all over the world building the protocols for Ethereum, and Hyperledger and other things. That's the way it should be. We need more of that. We need more kinds of people. It needs to be a highly diverse group of people. What's great about blockchain is that, it's so young. There is no group of people who can tell you that you don't know what they know, that they're better than you. Anybody that tells you that they know more about this than you do, they might know a little more, but mastery is 13 years and blockchain hasn't been around that long. Right, so this is your opportunity if you are a student who isn't represented by a lot of people that look like you or think like you. This is your shot. You can get in there and you can say, "No, I'm the expert. I'm the master on this," and that's good.

1. Proof of work is the \_\_\_\_\_ used by Bitcoin blockchain and Ethereum Byzantium Metropolis blockchain. 1 point
- ☐ Trust function
  - ☐ Incentive function
  - ☐ Transaction confirmation
  - ☐ Consensus Protocol
2. An approach for consensus protocol that is hotly debated among developers of blockchain is 1 point
- ☐ Round Robin
  - ☐ Proof of Stake
  - ☐ Proof of Incentive
  - ☐ Proof of Age

## 1.17 Robustness

Trust us not only about executing regular operations correctly but also about managing exception satisfactory. Robustness is the ability to satisfactorily manage exceptional situations. It's all the more important in a decentralized autonomous network such as a blockchain where there are no intermediaries minding the store. This lesson is about some of the exceptions that may occur during the blockchaining process. We'll discuss just two exceptions in this lesson and more in future courses of the specialization. Have you wondered, what if more than one miner solves the consensus puzzle where it close in time to each other? What if more than one transaction references as input the same digital asset? This situation is called double spending. Handling such exception satisfactorily is critical for ensuring the security of the blockchain. We start with the securechain indicated by three blocks and we want to add to this chain. Here, you see two miners have solved the consensus puzzle very close to each other. Bitcoin protocol allows this chain split or two chains for the next cycle. One led by each of the competing blocks. The probability that the next block will happen at the same time in both these chains is extremely low. So the winner of the next cycle for block creation consolidates one of the chains and that chain becomes the accepted chain. In this case, the newest block is added to the main chain. Now this chain is the longest and the valid main chain. The transaction in the other blocks are returned to the unconfirmed pool. Summarizing with a very low probability, the main chain may split but if it does, the bitcoin protocol has methods to consolidate it to a single chain within a cycle. Ethereum handles more than one person we know by allowing Omar or Runner-Up blocks and allocating a small incentive for these Runner-Up blocks. This incentive model helps in keeping the chains secure.

New blocks are added only to the mainchain and not to the Runner-Up chains. That are Runner-up blocks are maintained for six more blocks after they were added. Here you see a blockchain with two blocks one at the height, 4567, another one at a height, 4557. The one deeper inside the chain is more trustworthy than the one newly added. Let's now add us the double spending problem. There's a possibility that digital currency and other



consumables are single used digital assets, can be intentionally or inadvertently reused in transactions. Analogy time, using our airport flying analogy in lesson one, this is like an airline double booking a seat on a flight. In this case, a gate crew solved this problem by using Ad hoc methods such as asking for volunteers to relinquish their seats for money, etc. In a decentralized network, like a blockchain, there is no intermediary. We need a policy and an automatic deterministic way to handle this situation. A policy for handling transaction and double spending in Bitcoin is to allow the first transaction that reference the digital asset and reject the rest of the transaction that reference the same digital asset. In Ethereum, a combination of account number and a global nonce is used to address the doublet spending issue. Every time a transaction is initiated by an account, a global nonce is included in the transaction. After that, the nonce is incremented. Time stamp on the nonce in the transaction should be unique and verified to prevent any double use of digital asset. Summarizing, well-defined processes for handling exception improve trust in the blockchain. There are many other issues such as difficulty adjustment and hard and soft fork exception that will be discussed in the next lessons in the courses that follow.

## 1.18 Practice Quiz

1. What happens if more than one miner solves the consensus puzzle very close in time to each other in Ethereum?

1 point

- ☐ The new block is added to the main chain and not the runner-up chain
- ☐ Runner-up miners leave the network.
- ☐ Small incentives are given to runner-up blocks
- ☐ Small incentives are given to the runner up blocks and the new block is added to the main chain

2. Double spending is reusing digital assets intentionally or inadvertently. True or False?

1 point

- ☐ False
- ☐ True

3. In Ethereum, a combination of account number and global nonce is used to address issues regarding double spending. True or False?

1 point

- ☐ False
- ☐ True

## 1.19 Forks

Trust in Forks, a fork in the path. Fork, hard fork and soft fork, are most common phrases uttered in the context of a blockchain. We hear, at the block 4.7 million, Ethereum did a hard fork. Beware, 2018 is a year of hard fork in Ethereum blockchain. In this lesson, we explain at high level, hard and soft forks. Forks are just normal processes in an evolutionary path of the nascent technology enabling a blockchain. If robustness and trust is about managing exceptional situations, hard forks and soft forks are indeed at the front and center. We discussed

change split in the last lesson, that is a minor perturbation in the chain. Such situation is handled as a naturally expected occurrence within the block chain. On the other hand, occasionally, a minor process adjustment has to be carried out typically by bootstrapping a new software to the already running processes. This is soft fork. For example, the script concept in Bitcoin was introduced using this method. You can think of this as a software patch or a bug fix to address an issue. Hard fork implies a major change in the protocol. For example, the recent change from Ethereum Homestead to Metropolis Byzantium version was a planned hard fork and important note after a hard fork the emerging two chains are incompatible. Please be aware of this. There was an unplanned hard fork in Ethereum protocol, Ethereum Core and Ethereum Classic split, that was enacted to address a critical software issue in a decentralized autonomous organization (DAO). That resulted in a multi-million dollar heist. We'll discuss this in course three where we study decentralized organizations. Let's discuss the hard fork that happened on October 17, 2017. The Ethereum hard fork was a planned fork. Here are a few Ethereum improvement proposals (EIP) for this fork, parallel processing of transactions. Proof of Work consensus still stays except that every hundred block, Proof of Stake consensus protocol, is applied for evaluating the latter. And minor incentive was reduced from 5 ethers to 3 ethers for block creation. These are just few EIPs. You can look at the documentation for other EIPs. Summarizing, soft fork and hard fork in the blockchain world are like the



release of software patches, and new versions of operating systems respectively. Forks are mechanisms that add to the robustness of the blockchain framework. Well-managed forks help build credibility in the blockchain by providing approaches to manage unexpected faults and planned improvements. We are lucky that we have a front row seat to the planned hard fork of Ethereum Homestead to Ethereum Metropolis, and watch it as it plays out.

## 1.20 Practice Quiz

1. Bootstrapping the new software to the already running processes is known as \_\_\_\_\_. 1 point
  - ☐ Soft Forks
  - ☐ Hard Forks
  - ☐ Hashing
  - ☐ Scripting
2. After a hard fork, the emerging two chains are incompatible. True or False? 1 point
  - ☐ False
  - ☐ True
3. Bitcoin blockchain implemented a soft fork to realize a \_\_\_\_\_. 1 point
  - ☐ P2SH conditional payment script feature
  - ☐ P2SH payer gossip feature
  - ☐ P2SH Peer-to-Shell feature
  - ☐ Split into Bitcoin core and Bitcoin cash