Cryptography concepts based understanding of Blockchain technology

The term blockchain is of two separate terms Block and chain. Blocks are the collections of data and chains are lists which are public databases of blocks. These lists are linked using cryptographic algorithms which is the basic requirement of creating a blockchain. Cryptography is also composed of two ancient Greek words, Kryptos and Graphein which mean hidden and write respectively. Terms related are Encryption, Decryption, cipher and key. Encryption is converting plaintext to cipher text and decryption is vice versa. Cipher is the cryptographic algorithm used and a key is required to get the output of the algorithm. The main ways to implement these cryptographic algorithms are symmetric cryptography, asymmetric cryptography and hash functions. Symmetric is where the same key is used to encrypt and decrypt the data whereas asymmetric is where two different keys public and private keys are used. Hash functions do not require keys; it generates a fixed length hash from given input. Blockchain uses asymmetric key algorithms and hash functions. The main properties of hash functions are Avalanche effect, uniqueness, Deterministic and quickness. Avalanche effect, slight change in data that can give a different output. Every input has a unique output and the same input always gives the same output. The time taken to generate a hash is less and this process can not be reversed. Hash function plays a major role in blockchain, linking blocks to one another and helping maintain integrity of data stored in each block. Digital signature is designed to provide integrity and authenticity validation of data. This helps by verifying whether the message originated from the sender and message is not compromised. Multisignatures algorithm enables the generation of digital signatures where it requires multiple parties to be valid. Additional concept Is zero knowledge proof which enables a user to prove knowledge of a secret without revealing the secret itself. Stealth addressing is a concept which help anonymize the recipient side of a transaction by generating a one-time address and claiming the value, which makes it impossible to determine the transaction recipient. This is important as based on the transactions its possible to know how anyone’s value stored in the wallet. Ring signature helps in anonymizing the sender, it allows someone as a member of the group to sign data using a set of public keys which generates a signature. As public key is public information the signature creator should know the private key which verifies that the signer is an actual group member. Hash functions are simple commitment schemes which, but more complex. This helps hide the value of the transaction. The use of these additional cryptography concepts can provide additional functionality and guarantees that privacy can be maintained in blockchain in different ways without sacrificing the verifiability of transactions that is necessary for it to function.

Secure Hash Algorithm is the most widely used cryptographic hash function which has many variants. MD5 or the Message-Digest algorithm helps in generating a 128-bit hash value. SHA-1 was proved that it is not collusion resistant by google using distributed computing in 2015 which took nearly one and half year, no certificates are signed by sha1 on the internet. SHA-2, there is a fear that there is a possibility that this can be possibly cracked as this is based on SHA-1 concepts. SHA-2, developed by the National Security Agency ([NSA](https://en.wikipedia.org/wiki/SHA-2)), is a cryptographic hash function. SHA-2 includes significant changes from its predecessor, SHA-1. The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256. So NIST developed other hashing algorithms in 2012. SHA3-256 is Keccak-256, except that NIST changed the padding. For this reason, the original Keccak-256 gives a different hash value than NIST SHA3-256. Monero uses the original Keccak-256. The NIST standard was only published in August 2015, while Monero went live on 18 April 2014. Monero employs Keccak as a hashing function. In most contexts specifically Keccak-256 is used, providing 32-byte hashes. Keccak is the leading hashing function, designed by non-NSA designers. Keccak won the NIST [competition](https://en.wikipedia.org/wiki/NIST_hash_function_competition) to become the official SHA3. Monero does not employ Keccak for Proof-of-Work. Instead, Keccak is used for: random number generator, block hashing, transaction hashing, stealth address private key image (for double spend protection), public address checksum, RingCT, multisig, bulletproofs and likely a few other things. Keccak is a versatile cryptographic function. Best known as a hash function, it nevertheless can also be used for authentication, (authenticated) encryption and pseudo-random number generation. Its structure is the extremely simple sponge construction and internally it uses the innovative Keccak-f cryptographic permutation. Keccak has the state size of 1600 bits which is powerful when compared to AES standards. This state (S) is made up from r (rate) and c (capacity). For SHA3–224, we have a rate (r) of 1,152 bits and a capacity (c) of 448 bits. The total bits in the state is thus 1,600 bits. For SHAKE128 and SHA256, we can have a variable number of bits on the output. The key size used for an AES cipher specifies the number of transformation rounds that convert the input, called the [plaintext](https://en.wikipedia.org/wiki/Plaintext), into the final output, called the [ciphertext](https://en.wikipedia.org/wiki/Ciphertext). The number of rounds are as follows: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each round consists of several processing steps, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key. In [cryptography](https://en.wikipedia.org/wiki/Cryptography), a sponge function or sponge construction is any of a class of [algorithms](https://en.wikipedia.org/wiki/Algorithm) with finite [internal state](https://en.wikipedia.org/wiki/State_(computer_science)) that take an input [bit stream](https://en.wikipedia.org/wiki/Bit_stream) of any length and produce an output bit stream of any desired length. A sponge function is built from three components: a state memory, S, containing b bits, a function that transforms the state memory (often it is a [pseudorandom](https://en.wikipedia.org/wiki/Pseudorandomness) permutation of the state values), and a padding function P.

The state memory is divided into two sections: one of size r (the bitrate) and the remaining part of size c (the capacity). These sections are denoted R and C respectively. The padding function appends enough bits to the input string so that the length of the padded input is a whole multiple of the bitrate, r. The padded input can thus be broken into r-bit blocks. There are two phases one is the absorbing phase and the other is the squeezing phase. In the absorbing phase the data is taken as input and padding is added to divide the data into chunks depending on state size.

IOTA, for example, uses the Keccak-384 hash. In the first part of the process, we break the data into r bits chunks, and then EX-OR it with the rate part of the state (and which will initially be set to all zeros). The output of this goes into the function (f), and which has 24 rounds and is created with EX-OR, AND and NOT functions. This output is the new state (S). If there is more data, it is then fed into the next stage, and will do the same method of EX-ORing the r state value with the message (X1), and then feeding this into f. This generates the next state. We then continue until we have exhausted all the message data, and then go into a squeezing function. Again we feed into f, and produce an output (Y0). This output will then be truncated to the required hash size, or we will continue until we have produced the required output size (as with SHAKE128 and SHAKE256).

Y0 is 1152 but the first 256 bits are taken and provided as output hash.

The history of crypto currency mining began with the Bitcoin [SHA256 algorithm](https://coinguides.org/sha-256-algorithm-coins/). Followed by Litecoin [Scrypt algorithm](https://coinguides.org/scrypt-coins/), then came [X11](https://coinguides.org/x11-algorithm-coins/) algo for Dash and [Ethash](https://coinguides.org/ethash-coins/) for Ethereum. Out of the top 5 cryptocurrencies by [market cap](https://coinguides.org/market-capitalization-circulating-supply-volume/) 3 are [mineable coins](https://coinguides.org/mineable-vs-non-mineable-coins/). Bitcoin (BTC), Ethereum (ETH) and Litecoin (LTC). All three operate within the proof-of-work consensus mechanism. However they all use different hashing algorithms. Bitcoin uses SHA-256 hash function and Ethereum currently uses [Ethash](https://coinguides.org/ethash-coins/) Proof of Work hashing algorithm. In the near future Ethereum will no longer support PoW as it will completely switch over to Proof of Stake. Litecoin on the other hand uses Scrypt algorithm. Apart from Litecoin; Scrypt algorithm is also being used on various other PoW blockchains.

The main reason for change in different algorithm is to resist purpose build hardware’s like ASICs (Application Specific Integrated Circuits) and FPGAs (Field Programmable Gate Arrays). When a coin or Cryptocurrency says ASIC resistant; then it means there is no ASIC machine or solution developed yet to mine the coin. It can only be mined using consumer grade hardware’s like CPU and GPUs. As time goes someone may or may not develop an ASIC for that coin or algorithm because technically ASICs can be developed for any mining algorithm. It’s only a matter of time for someone to crack it and that’s why they never say ASIC proof but ASIC resistant

In cryptography, scrypt is a password-based key derivation function (KDF). Password based KDF are designed to be computationally intensive and they are efficient at preventing brute force attacks.Scrypt being both computationally intensive and memory intensive hashing algorithm it was originally used for Tarsnap’s online backup service. The algorithm was designed by Colin Percival. On May 2009 Colin published scrypt paper titled, “Stronger Key Derivation Via Sequential Memory-Hard Functions.”.

X11, which has been designed for [cryptocurrencies](https://cryptoadventure.com/beginners-guide-to-cryptocurrency/), is still more secure than the algorithm SHA-256 and presents the advantage of not being used by “Application Specific Integrated Circuits.” It is currently being used in several crypto-currencies, the most famous being Dash.X11 organizes a chain of 11 different hash algorithms: Blake, BMW, groestl, JH, keccak, skein, Luffa, cubehash, shavite, SIMD, and echo. To fail, all of the 11 hashing algorithms must default simultaneously. It is estimated that the probability of such a situation is close to zero. This is why ASICs cannot perform the Algorithm X11 since it included 11 different algorithms. BLAKE, BLUE MIDNIGHT WISH (BMW), Grøstl, JH, Keccak, Skein, Luffa, CubeHash, SHAvite-3, SIMD and ECHO. To illustrate how the algorithm works at a high level, when a value is submitted to the BLAKE function, it produces a hash value; this value is then submitted to the BMW function, which then produces another value; this process repeats until the very last function. All of the algorithms above were entered into the [US National Institute of Standards and Technology](https://www.nist.gov/) (NIST) open competition in order to develop a new hash function, [SHA-3](https://en.wikipedia.org/wiki/SHA-3), that was more secure than the previous generations: [SHA-1](https://en.wikipedia.org/wiki/SHA-1) and [SHA-2](https://en.wikipedia.org/wiki/SHA-2). Of the 64 algorithms that were entered into the competition, all the X11 hash functions were accepted as first-round candidates. Of the 14 second-round candidates, all the X11 algorithms were once again accepted, with the only following functions making it into the final 5: BLAKE, Grøstl, JH, Keccak and Skein. Keccak would then be the function to win the competition.

Ethash is the [proof-of-work](https://cryptoadventure.com/evaluation-of-blockchain-consensus-algorithms/) function in [Ethereum-based blockchain](https://cryptoadventure.com/the-beginners-guide-to-ethereum/) currencies. It is a hash function belonging to the Keccak family, the same family to which the SHA-3 hash functions belong. However, Ethash is not an SHA-3 function and should not be confused with them. Ethash was developed as an upgrade of its predecessor called Dagger-Hashimoto to remove computational overhead.

Here comes the list of crypto currencies that are powered by Ethash proof of work hashing algorithm. There are also many tokens that are based on Ethereum and we are not going to list that. We are only going to list mineable coins that uses Ethash algorithm. The below listing order is based on coin’s popularity. Ethereum – ETH, Ethereum Classic – ETC, Metaverse ETP – ETP, Expanse – EXP, Musicoin – MUSIC, Ellaism – ELLA, Elementrem – ELE, WhaleCoin – WHL (Not Minable), DaxxCoin – DAXX, Bowhead – AHT (Not Minable), Ethereum Fog

IPFS:

It is Decentralized internet similar to bittorrent and uses SHA-256 hashing algorithm by default. Issues with centralization are Centralized servers Ex: Youtube, Wikipedia, google and Censorship : government can block access Ex: Turkey 2017 Blocked Wikipedia. Location based addressing is where the data is obtained based on the ip address which means the location of the server. Where as Content based addressing Based on the hash of the file. Data is stored in IPFS objects which can store upto 256kb and objects also contains links of another objects. Larger files are stored using empty IPFS object which has links of the ipfs objects related to file. IPFS also has version controlling. Filecoin is created by same people as IPFS: Decentralized market for storage. Provide incentives for keeping file online and it also help replicate files for better availability.Hash which is known as content id. Distributed hash tables maps content id to peer address. DHT server(IPFS server) hosts DHT’s. DHT client (IPFS client) connects to DHT server. IPFS clients ask local DHT for Cid and retrieves collection of ip address. If nothing returns it asks its peers.

Blockchain Fundamentals

Properties of blockchain are, it is Decentralized,Transparent, Speed, Immutable and security, Remove counterparty risk, Trust minimized agreements, math-based agreements, Hybrid Smart contracts combine on and off-chain: smart contracts with oracle(data). DAO’s is known as decentralized autonomous organizations. For governance in digital world. Etherscan is a block explorer, its is an web based application that helps us view the transactions that happen on blockchain. Gas is a unit of computational measure the more computation uses the more gas you need to pay. Every transaction on the chain pays a gas fee to the node operator. Gas: measure of computation use. Gas price: how much it costs per unit of gas. Gas limit: Max amount of gas in transaction. Transaction fees: Gas used \* Gas price. Ethgasstation.info is the link to view the information. Hash function used by Ethereum is keccak256. First block in the blockchain is called the Genesis Block. Mining is the process of solving the blockchain problem. Ex: Our problem is to find the hash with 4 zeros. Nodes get paid for mining blocks. Block is the list of transactions mined together. Nonce is A ‘number used once’ to find the solution to the blockchain problem. Nonce is also used to define the transaction number for an account/address. Public key algorithm used for both bitcoin and ethereum is the Elliptic curve digital signature algorithm (ECDSA). Private key is used to sign the transaction. Public key is generated from the private key and used to verify the transaction. Node is a single instance in the decentralized network. Blockchain is resilient because at least one node is running somewhere. Blockchain nodes keep lists of transactions that occur. Consensus is a mechanism used to agree on the state of the blockchain. Which is divided into 1. Chain selection 2, sybil Resistance. Sybil Resistance mechanism: Proof of work(PoW) and Proof of State

Pow is finding the correct solution: who the block author is. Block time is the time between blocks being published. Depending on the block time the problem is made (Hard or easy). Chain Selection, nakamoto consensus : Combination of the PoW and longest chain of blocks is the chain of use which means chain which has a greater number of blocks. Miners use Proof of work and Validators use Proof of stake. Proof of work uses a lot of energy. Node will be getting paid if it solves the problem first and it will be paid. As 1. Transaction fees 2. Block reward. Block rewards are only paid until a certain amount of time and after miners are paid only transaction fees. Gas fees are paid only by the one who does the transaction. Attacks in Blockchain are Sybil attacks,which occur when a user creates a whole bunch of pseudo-anonymous accounts to influence a network. 51% attack happens when the chain which becomes 51% more than a network becomes the longest and with more nodes in a network could execute this attack. Proof of stake is a protocol which reduces the problem of Pow which consumes a lot of energy (electricity). Proof of stake nodes put up collateral as a sybil resistance mechanism. If they misbehave, they cut the stake/slashed. Nodes are randomly chosen Validators are used to validate whether they behaved honestly. Pros of Pos are less energy consumption, only one node to do this and other nodes should validate and Sybil resistance mechanism. Cons are it is less decentralized and Environmental effect. Scalability property when more people use blockchain gas cost increases and there is not enough block space to get the transactions. Eeth 2.0 solves POS and Sharding. Sharding is a blockchain, which is connected to the main chain. Layer1 is base layer blockchain solutions. Layer2 is any application that works on the layer1 blockchains: chainlink. Sharing and rollups solve scalability problem

Overall understanding is Eth and btc are based on Proof of work, Eth 2.0 is based on Proof of Stake. PoW & PoS mechanisms prevent sybil attacks. The bigger the blockchain the more secure it is. Consensus is how blockchain decides what the state of the chain is.

Contracts

Solidity's code is encapsulated in contracts. A contract is the fundamental building block of Ethereum applications — all variables and functions belong to a contract, and this will be the starting point of all your projects.All solidity source code should start with a "version pragma" — a declaration of the version of the Solidity compiler this code should use. This is to prevent issues with future compiler versions potentially introducing changes that would break your code. State variables are permanently stored in contract storage. This means they're written to the Ethereum blockchain. Think of them like writing to a DB. The uint data type is an unsigned integer, meaning its value must be non-negative. There's also an int data type for signed integers. In Solidity, uint is actually an alias for uint256, a 256-bit unsigned integer. You can declare uints with less bits — uint8, uint16, uint32, etc.. But in general you want to simply use uint except in specific cases, which we'll talk about in later lessons. Math in Solidity is pretty straightforward. Solidity also supports an exponential operator. Sometimes you need a more complex data type. For this, Solidity provides structs. When you want a collection of something, you can use an array. There are two types of arrays in Solidity: fixed arrays and dynamic arrays. Remember that state variables are stored permanently in the blockchain? So creating a dynamic array of structs like this can be useful for storing structured data in your contract, kind of like a database.

Functions and reference type

In Solidity, functions are public by default. This means anyone (or any other contract) can call your contract's function and execute its code. Obviously this isn't always desirable, and can make your contract vulnerable to attacks. Thus it's good practice to mark your functions as private by default, and then only make public the functions you want to expose to the world. In Solidity, the function declaration contains the type of the return value (in this case string). view function, meaning it's only viewing the data but not modifying it. Solidity also contains pure functions, which means you're not even accessing any data in the app. This function doesn't even read from the state of the app — its return value depends only on its function parameters. So in this case we would declare the function as pure. It may be hard to remember when to mark functions as pure/view. Luckily the Solidity compiler is good about issuing warnings to let you know when you should use one of these modifiers. Ethereum has the hash function keccak256 built in, which is a version of SHA3. A hash function basically maps an input into a random 256-bit hexadecimal number. A slight change in the input will cause a large change in the hash. It's useful for many purposes in Ethereum, but for right now we're just going to use it for pseudo-random number generation. Also important, keccak256 expects a single parameter of type bytes. This means that we have to "pack" any parameters before calling keccak256. Events are a way for your contract to communicate that something happened on the blockchain to your app front-end, which can be 'listening' for certain events and take action when they happen. Your app front-end could then listen for the event. Ethereum has a Javascript library called Web3.js.

References:

1. <https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto>
2. <https://eth.wiki/en/concepts/ethash/ethash>
3. [https://asecuritysite.com/hash/s3](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbWNqUmhmeWx6NjN3Zmd1VzBwTlRmc0xVNnBZUXxBQ3Jtc0tuS0Y5cVhOT2ZHNXkzZkdjdkpxbmlqVC1TYUljeHNpRXEyMG9TclBKX0ZwVGRvUWZWZUJVb2d2RzR1MHFqWVRXVWVGb2Z1M3gza1o4WTZZZ0Z5OGdzc1Vsd29DQUMxNFdGNWdmdktQdGlFSFBoa1RCVQ&q=https%3A%2F%2Fasecuritysite.com%2Fhash%2Fs3&v=bTOJ9An9wpE)
4. [https://asecuritysite.com/hash/gokang](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa2VzR1pYQTFJN0FCbi1CV09QaGNGU2dYVlA5Z3xBQ3Jtc0tsUllhaGF3SDZ1eF9vNzl2SE5xMVZsckFaQ2w1SGJxdXlXbWZrYTZpT0d5UHNDUm9HUDBXN1dNZTZGZGd4dWlqYzlOMlBqMWh4ZTVqaG1WNFdjQnRHM1hWcy13c2h4azlBcFBJMWllRVpvSk1RZndpZw&q=https%3A%2F%2Fasecuritysite.com%2Fhash%2Fgokang&v=bTOJ9An9wpE)
5. [https://asecuritysite.com/hash/goshake](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGlzMnVONTYxcEZSS2VkakpoWTNoN2F0TjVfUXxBQ3Jtc0tsYTY0dm05U0lCWUMzWlo0cGZXRU9NQVV6QmVfM1BUT2U2d3JmRTlWQmRXRFZuNlRXMnhQa2dXeGJQX3dwNWFpSTVqLUJSRHNnWFY2MVFqZGJ0TDUyS3dUR2oxNEc1YzNaMS1kNFJ0cUhadmJ3YnlCMA&q=https%3A%2F%2Fasecuritysite.com%2Fhash%2Fgoshake&v=bTOJ9An9wpE)
6. <https://zerocrypted.com/what-is-ethash/>
7. <https://coinguides.org/asic-resistance-explained/>
8. <https://medium.com/asecuritysite-when-bob-met-alice/one-of-the-greatest-advancements-in-cybersecurity-the-sponge-function-keccak-and-shake-6e6c8e298682>
9. <https://www.youtube.com/watch?v=bTOJ9An9wpE&t=2s>
10. <https://en.wikipedia.org/wiki/NIST_hash_function_competition>
11. <https://www.mycryptopedia.com/x11-algorithm-explained/>
12. <https://monerodocs.org/cryptography/keccak-256/>
13. <https://cryptoadventure.com/blockhain-hashing-algorithms-explained-all-you-need-to-know/>
14. <https://coinguides.org/scrypt-coins/>
15. <https://cointelegraph.com/explained/what-is-a-directed-acyclic-graph-in-cryptocurrency-how-does-dag-work>