A **blockchain** is a growing list of [records](https://en.wikipedia.org/wiki/Record_(computer_science)), called *blocks*, that are securely linked together using [cryptography](https://en.wikipedia.org/wiki/Cryptography). Each block contains a [cryptographic hash](https://en.wikipedia.org/wiki/Cryptographic_hash_function) of the previous block, a [timestamp](https://en.wikipedia.org/wiki/Trusted_timestamping), and transaction data (generally represented as a [Merkle tree](https://en.wikipedia.org/wiki/Merkle_tree), where [data nodes](https://en.wikipedia.org/wiki/Node_(computer_science)) are represented by leaves).

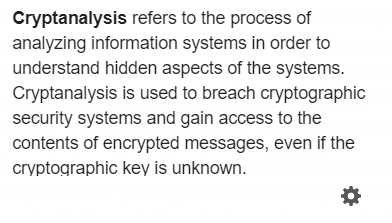
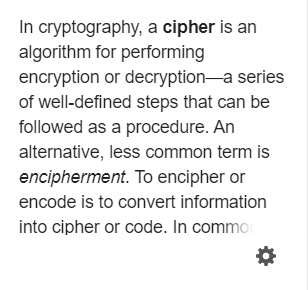
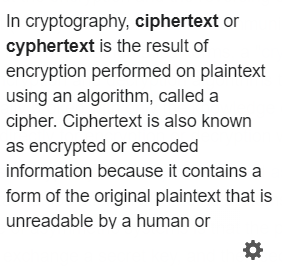
A **ledger** is a book or collection of accounts in which account [transactions](https://en.wikipedia.org/wiki/Financial_transaction) are recorded. Each account has an opening or carry-forward [balance](https://en.wikipedia.org/wiki/Balance_(accounting)) and would record each transaction as either a [debit or credit](https://en.wikipedia.org/wiki/Debits_and_credits) in separate columns and the ending or closing balance.

// more can be read in [the link](https://onezero.medium.com/how-does-the-blockchain-work-98c8cd01d2ae)

// [video link](https://www.youtube.com/watch?v=bBC-nXj3Ng4)

**Cryptography** refers to secure info & communication techniques derived from a mathematical concept & a set of rules-based calculations called algorithms to transform messages in a way that is hard to decipher.

cryptography referred almost exclusively to "encryption"



A **hash** is a mathematical function that converts an input of arbitrary length into an encrypted output of a fixed length. Thus, regardless of the original amount of data or file size involved, its unique hash will always be the same size.

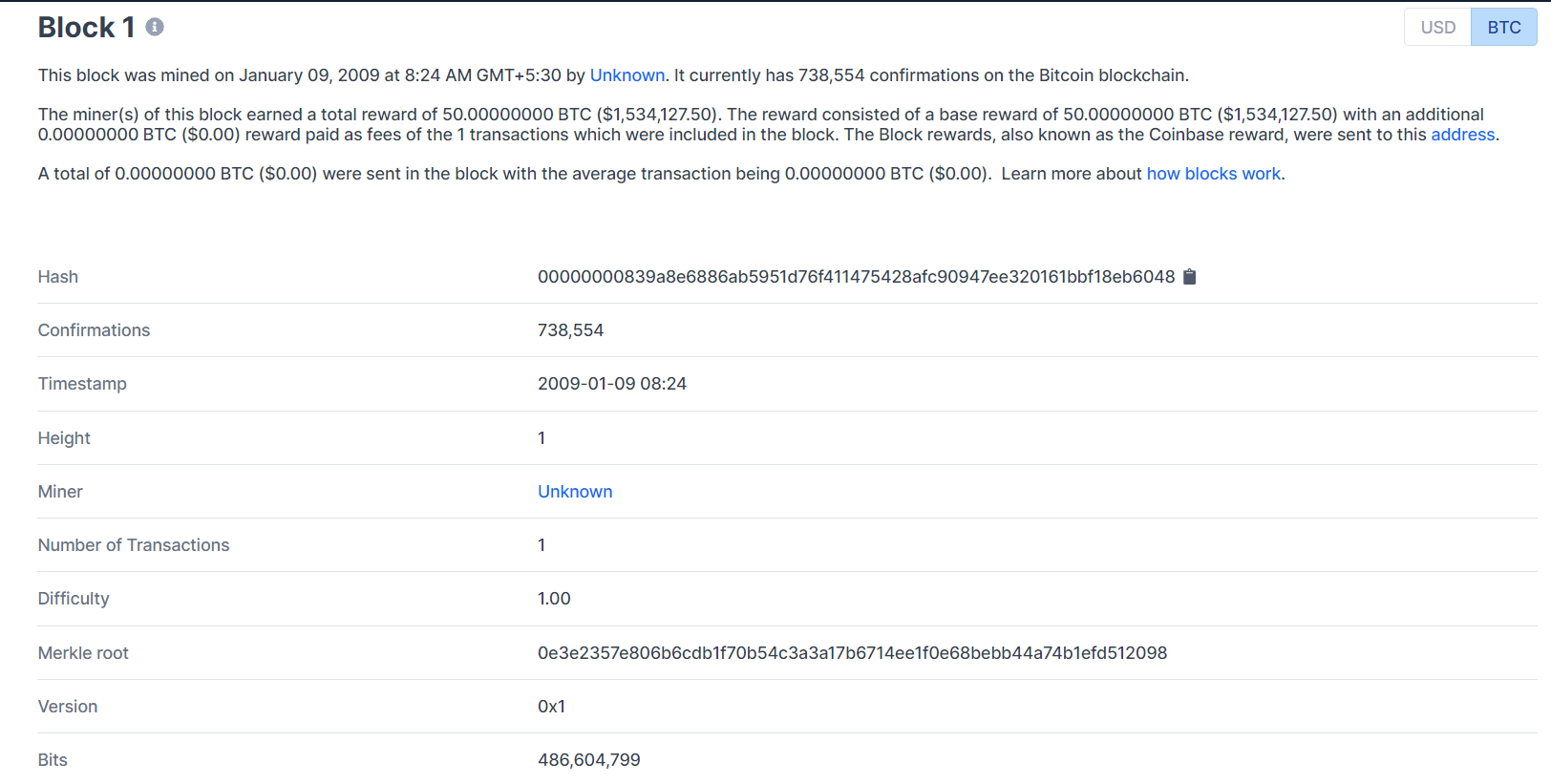
Hash functions are mathematical functions that transform or "map" a given set of data into a bit string of fixed size, also known as the "hash value."

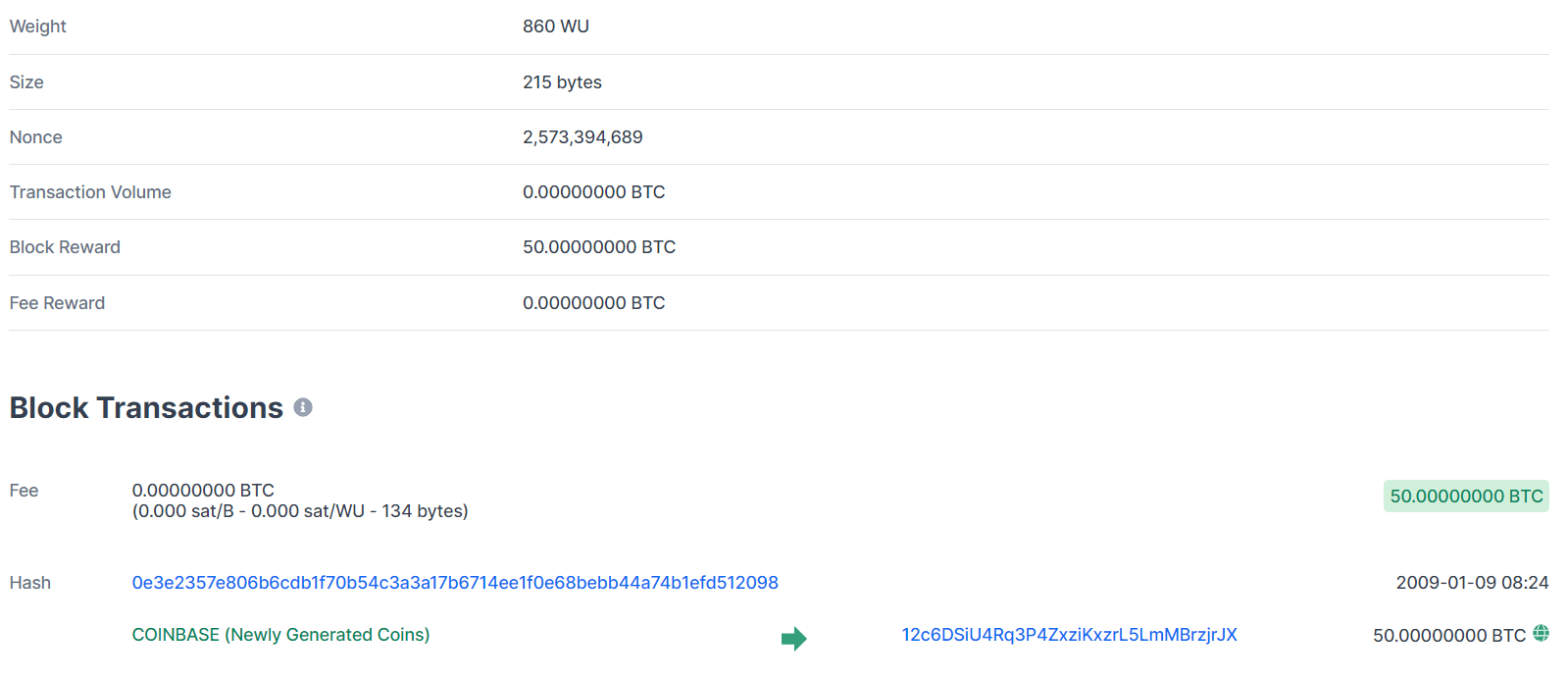
A hash function utilizes complex mathematical algorithms that convert data of arbitrary length to data of fixed length (for instance, 256 characters)

**SHA stands for Secure Hash Algorithm**, which represent cryptographic hash functions. These functions have excellent uses in protecting sensitive information such as passwords, personal identifiers such as identification items. It is a one-way algorithm, meaning that under current technologies, the algorithm cannot be returned to its original value, and two different input values will practically never yield the same result, allowing us to maintain the integrity and uniqueness of data.

[**Avalanche effect**](https://www.geeksforgeeks.org/avalanche-effect-in-cryptography/)

SHA256 as it outputs 64 characters each character is represented by 4 bytes (64\*4=256)

****

****

[**BLOCK OF BITCOIN**](https://www.blockchain.com/btc/block/00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048)

**Nonce**, short for “number used once”, is a random number that can only be used one time. Nonces are generated for a specific use, most often to modify the result of a function in a cryptographic communication. Typically, it is a number that varies with time, in order to ensure that some values cannot be reused. It can be a timestamp or a special marker intended to prevent unauthorized reproductions of a file.

Discovering the next block is done by finding the right nonce, that when combined with the previous block hash, provides a lower hash value than the target. This specific nonce that satisfies the mining requirement of the next block is called the “Golden Nonce”.

**Nonce is the central part of Proof of Work.** The Nonce is a random whole number, which is a 32-bit (4 byte) field, which is adjusted by the miners, so that it becomes a valid number to be used for hashing the value of block. **Nonce is the number which can be used only once**. Once the perfect Nonce is found, it is added to the hashed block. Along with this number, the hash value of that block will get rehashed and creates a difficult algorithm.

**Proof of work (PoW)** is a decentralized consensus mechanism that requires members of a network to expend effort solving an arbitrary mathematical puzzle to prevent anybody from gaming the system. Proof of work is used widely in cryptocurrency mining, for validating transactions and mining new tokens. Due to proof of work, Bitcoin and other cryptocurrency transactions can be processed peer-to-peer in a secure manner without the need for a trusted third part

Alternative of proof of work the [**proof of stake**](https://youtu.be/M3EFi_POhps)

**Blockchain mining** involves adding transactions to the existing blockchain ledger of transactions distributed among all users of a blockchain. It involves creating a hash of a block of transactions that cannot be easily forged, protecting the integrity of the entire blockchain without the need for a central system.

In other words, Mining is the process by which new blocks are published (added) onto the blockchain. The nodes are in a constant race to get transactions, verify them, package them into a block and publish them on a Blockchain, as they are driven by incentives (usually mining a block results in some reward being given to the miner, in the form of new Cryptocurrency tokens or rewards for approving transactions).

Now as the mining process just involved verifying transactions before publishing them, several nodes can publish a block simultaneously or with a minimal difference in publishing time, which can lead to difficulty in achieving consensus by the nodes [(read)](https://en.wikipedia.org/wiki/Fork_(blockchain)). Also, a malicious node can spam the network with a fraudulent block, which can have a significant chance of being accepted. Thus, it is crucial to make this process of mining (publishing new blocks) slower and the process of verification of blocks faster so that Blocks aren't mined simultaneously and malicious blocks can be appropriately verified and aren't spammed. Essentially, we want a difficult and time-consuming problem to solve but easy to verify.

In easy language working of miners: -

Suppose you have a string/an image/anything else. You convert it into binary. You put the bytes together. You now have a long sequence of bits. Now assume that this sequence of bits represents a number. A very large number. But still a finite positive integer.

Suppose you have a mathematical function f(x): Z-> [1,100]. This function can take any integer as the input but always produces the output as an integer between 1 and 100. It is given that this function produces a uniform distribution. So, you give this function an integer as an input.

What is the probability that the output will be 5? 1/100?

What is the probability that the output will be less than or equal to 5? 1/20?

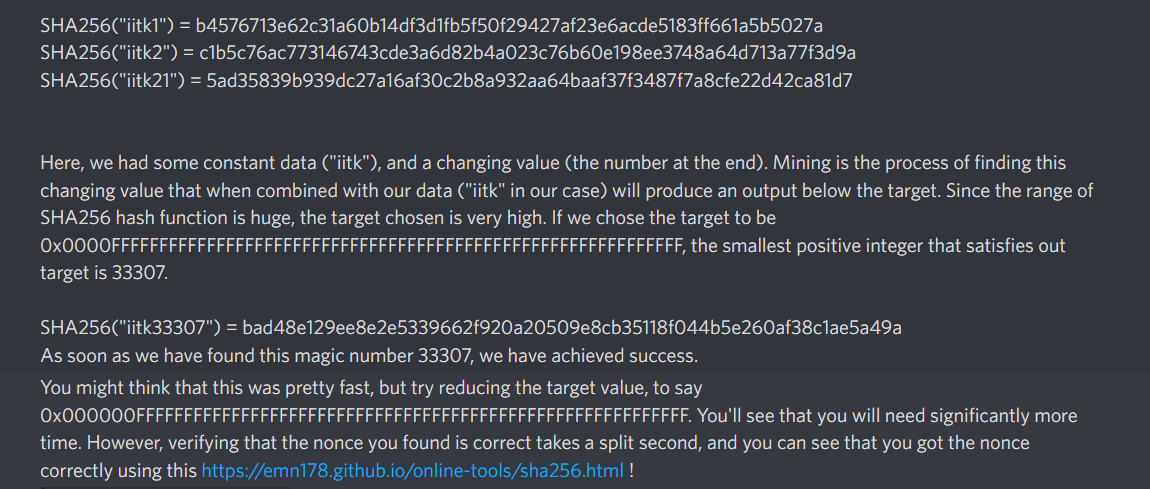
Let's call 5 our target. Now suppose you want to throw random numbers into the function as input, and you want the output to be less than or equal to our target (which is 5). What is the expected number of times you'll have to try? 20?

Now suppose you decrease the target to 4. What is the expected number of times you'll have to try to find an integer x such that f(x)<=target? 25? Thus, now you have to try harder (do more work) to find such a number. Now suppose you increase the target to 10.

What is the expected number of times you'll have to try to find an integer x such that f(x)<=target? 10? Now you have to do less work to find such a number.

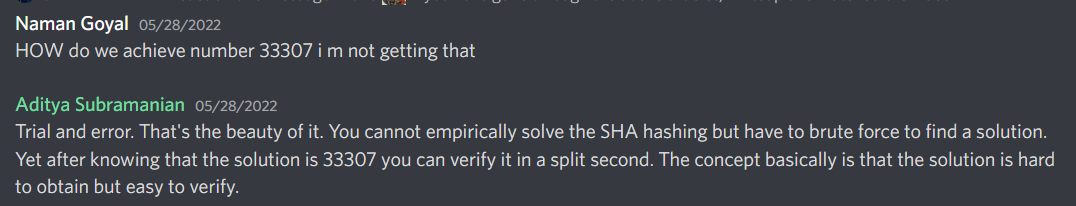
This is the essence of mining. You try to find such input for a function which produces an output that is less than the target. The function used is a hash function, like the SHA-256 hash function. It produces 256 bits as the output (hence the name). If you represent the bytes as an integer, it would lie in the range [0, 2^256). The number is usually written in hexadecimal format. So, the hash value can range from0x0000000000000000000000000000000000000000000000000000000000000000 to 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF. But the hash function can take any sequence of bytes as input.

if you change even a single bit in the input, the generated output is very different from the earlier output. For example:

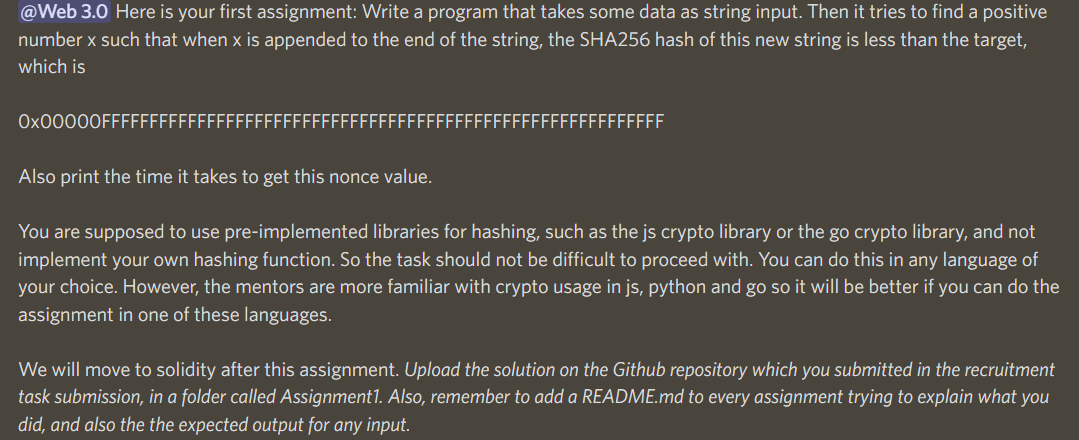


[SHA256 ONLINE](https://emn178.github.io/online-tools/sha256.html)

Doubt



Assignment



Doubt

the current total hash rate of miners is around 150 TH/s, that is the ability to perform 150 Tera hash function in a second. As checked, the value of the nonce that miners need to guess to mine the block is capped at 32 bits i.e., around 4 billion numbers. Means there’s only a total of 4 billion trial to guess the nonce value per block? But the total hash rate is 150 Tera hash/I. Won’t the miners able to complete all the guessing in like less than a second? Why Bitcoin transaction per second is still low? Please correct me if I am wrong.