Introduction

With most cryptocurrencies that are not based on Proof-of-Stake, there is a process called mining. This process is the foundation for a blockchain to grow and secure transactions within the network. ATMCoin is no different, except that you use precomputed hashes to find values that can be used to forge a block. To fully understand this document, you should read the document named technical information to create plot files. This document is intended to be an overview of the processes. It is technical information, but not deep enough to be used as a reference for a programmer since information regarding subjects like AT, subscriptions, and assets is missing.

Algorithms and acronyms

- **Shabal / Sha256 / Curve25519**

Shabal, Sha256 and Curve25519 are cryptographic hash functions used in this text. Shabal is the main one used by ATMCoin. Shabal is a rather heavy and slow cryptographic hash function in relation to many others like SHA256. Because of this, it makes it a good crypto for Proof-of-Capacity coins like ATMCoin. This is because we store the precomputed hashes, and it is still fast enough to do smaller live verifications. ATMCoin uses the 256bit version of Shabal also known as Shabal256.

- **Hash / Digest**

A hash, or digest in this context, is a result when computing data through a cryptographic hash function. If not said otherwise, the length of a hash is 32Bytes (256bit).

- **Plot files**

When mining, you read precomputed hashes from files stored on a storage device. These files are called plot files.

- **Nonce**

Within a plot file, there are one or more groups of data called nonces. One nonce contains 8192 hashes, and because of that, the nonces are 256KiB large. Each nonce has its own individual number. This 64bit number can range between 0-18446744073709551615 (2^{64}).

- **Scoop**

Each nonce is sorted into 4096 different places of data. These places are called scoop numbers. Each scoop contains 2 hashes. Each of these hashes are xored with a final hash.

- **Account ID**

When you create your plot file it will be bound to a specific ATMCoin account. Because of this, all miners have different plot files.

- **Deadline**

When you mine and process your plot files, you will end up with resulting values called deadlines.

The values represent the number of seconds that must elapse since last block was forged before you are allowed to forge a block. If no one else has forged a block within this time, you can forge a block and claim a block reward.

- **Block reward**

If you are lucky enough to forge a block, you will get ATMCoin as a reward. This is called a block reward. The block reward decreases 5% every 10800 blocks. This is roughly every 30 days since each block is supposed to be forged every 4 minutes (360 blocks a day). ● **Base target**

Base target is calculated from the last 24 blocks. This value adjusts the difficulty for the miners. The lower the base target, the harder it is for a miner to find a low deadline. It gets adjusted in a way that ATMCoin can have an average of 4 minutes for each block.

- **Network Difficulty**

Network Difficulty, or NetDiff in short, is a value that can be read as an estimate on the total amount of space in terabytes dedicated to mine ATMCoin. Since this is a value that changes with every block in relation to base target, it should be taken into an average of at least 360 values before considered to be somewhat accurate. ● **Block Height**

Every block forged gets an individual number. Every new block forged gets the previous block’s number + 1. This number is called block height, and can be used to identify a specific block.

- **Block Generator**

When a block is forged, an account has found a nonce and a deadline. Block generator is the account used when forging a block. This is the account from which a deadline has been found when forging a block. This is always the real account even if a reward assignment has been set. ● **Generation Signature**

Generation signature is a based from the previous block generation signature and block generator. This value is then used by miners to forge a new block. Generation signature is 32bytes long. ● **Block signature**

Every block is signed by the generator who forges a block. This is done by taking most parts of the block and signing it with the block generator’s private key using both Sha256 and Curve25519. The result is a 64byte long hash.

- **Reward Assignment**

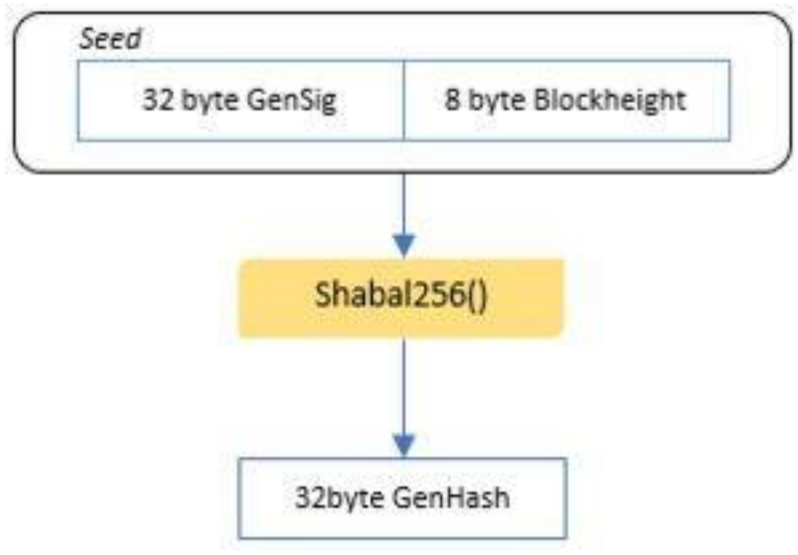
Reward assignment is frequently used when pool mining. When changing your reward assignment, you tell the network that another account (the pool account) is acting in your place for 2 specific features. The first feature is that all block rewards that should be given to your account will now be given to the pool account instead. Secondly, for the pool to be able to utilize the deadlines found from your plot files, it is also granted the action to sign the newly forged blocks with the account belonging to the pool.

Mining Process

All references to wallet in this text can also be a pool depending on scenario.

All references to miner in this text is a software able to do a mining operation for ATMCoin.

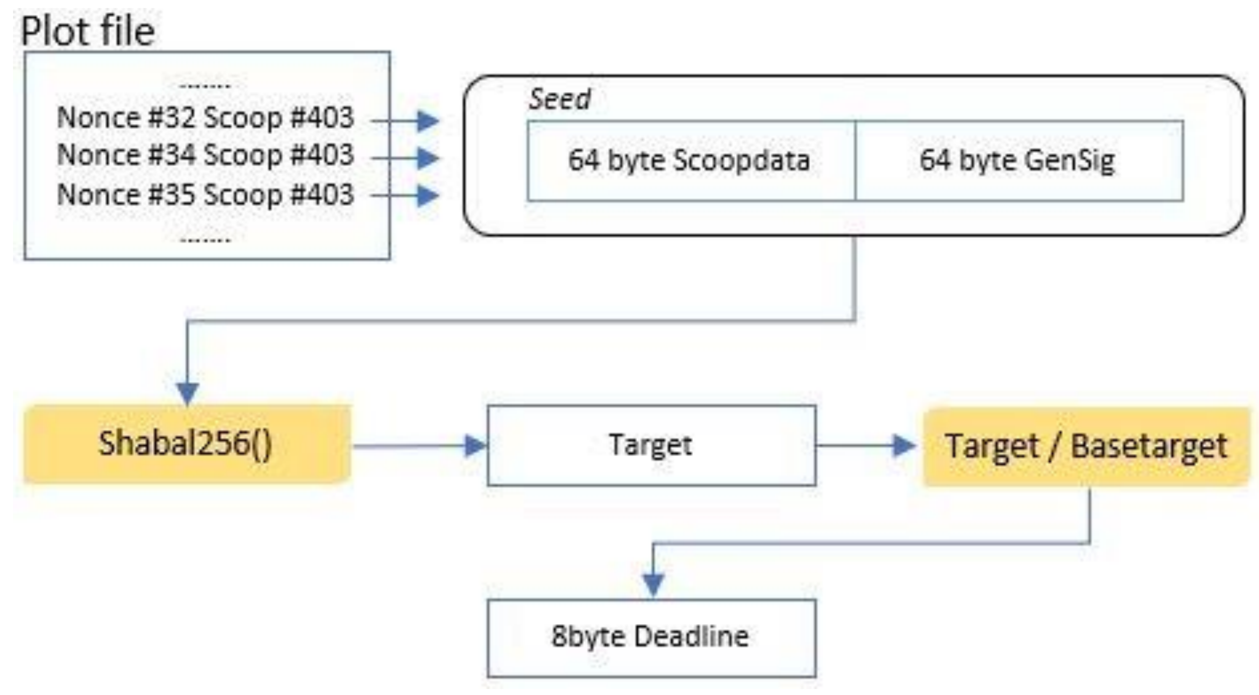
The first thing that happens when you start mining, is that the miner talks to the wallet and asks for mining information. This information contains a new generation signature, base target, and the next block height. Before the wallet sends over this info, it creates the generation signature by taking the previous generation signature together with previous block generator and runs this though shabal256 to get the new hash. The miner will now take the new 32byte generation signature, and the 8byte block height, and put them together as a seed for Shabal256. The result will be a hash value called Generation hash.



Now, the miner will do a small mathematical operation on this hash to find out which scoop number to use when processing the plot files. This is done by taking the generation hash modulo 4096, as there are only that many scoops.

GenHash
Modulo 4096
Scoop nr

Next step for the miner is to read all the 64-byte long scoops from all nonces in all plot files. It will process them individually through shabal256 together with the new generation signature to get a new hash called target. This target is now divided with base target and the first 8bytes of the result is the value deadline.



To prevent so-called “nonce spamming” to the wallet, the miner usually checks if the current deadline found is lower than the lowest one it has found so far. Usually there is also a max value that can be set, as ridiculously large deadlines are of no use to anyone. After these checks, the miner submits information to the wallet. This information contains the numeric account ID bound to the plot file, and the nonce number that contains the scoop data used to generate the deadline. If you are solo mining the miner also sends over the passphrase for the account id used in plot files. If the password is not sent when solo mining, the wallet would be unable to forge blocks for that account. When pool mining, the passphrase for the pool account id is used.

Block forging process

Handling deadlines

The wallet has now received the information submitted by the miner, and will now create the nonce to be able to find and verify the deadline for itself. After this is done, the wallet will now check and see if an equal amount or more seconds has passed as defined by the deadline. If not, the wallet will wait until it has. If a valid forged block from another wallet is announced on the network before the deadline has passed, the wallet will discard the mining info submitted since it is no longer valid. If the miner submits new information, the wallet will create that nonce and check if the deadline value is lower than the previous value. If the new deadline is lower, the wallet will use that value instead. When the deadline is valid, the wallet will now start to forge a block.

Forging

There are two limits for a block. First, a block can contain max. 255 transactions. The second is that a block payload can have max. 44880bytes (43KiB). The wallet will start by getting all of the unconfirmed transactions it has received from users or from the network. It will try to fit as many of these transactions possible until it hits one of the limits, or until all transactions are processed. For each transaction the wallet reads, it will do checks. For example, if the transaction has a valid signature, if it has a correct timestamp, etc. The wallet will also sum up all of the added transactions amounts and fees. The block itself will only contain the Transaction ID of each transaction and one Sha256 hash of all the transactions included. Complete transactions are stored separately. Beside this, a block contains many different sets of values.

Block contents

- **Block version number**

The version number is basically telling the wallet what a block can contain and how it is contained. This number changes each time a block gets a new format.

- **List of Transaction ID**

A List of all transaction IDs that are included in this block

- **Payload Hash**

This is the Sha256 hash of all the data in the payload of the block

- **Timestamp**

A timestamp that will describe when the block was forged; derived from the birth of the blockchain. Birth date: 11 august 2014, Time: 02:00:00

- **Total amount of coins**

This is the sum of all transactions in the block

- **Total amount of fees**

This is the amount of fees that will be given to the block forger for generating this block. ●

The length of the payload

This is a number in bytes representing the length of the payload.

- **Public Key**

This is a public key for the account that forges the block.

- **Generation Signature**

The 32byte generation signature that was used to forge the block.

- **Previous block hash**

A Sha256 hash of the contents from the previous block.

- **Previous block ID**

This is the first 8 bytes in the previous block hash converted to a number.

- **Cumulative Difficulty**

Used to prevent Nothing at Stake problems during potential forks. Calculated: Previous Cumulative Difficulty + (18446744073709551616 / base target)

- **Base Target**

The base target used when forging this block

- **Height**

This block's height value

- **Block ID**

This is the first 8 bytes in block's hash converted to a number

- **Nonce**

The nonce number used to forge this block.

- **AT**

If an AT is added to this block, this is the payload bytes for that AT.

- **Block Signature**

This is a 64byte hash generated with the forger's private key and block contents. When this is done, it will be announced to the network. The wallet will connect to all peers and send the block over to them. The peer will receive the block and verify that all information is not spoofed.