SpartanGoldPrime

Alternative Proof-of-Work for SpartanGold

Dishen Zhao

Project Topic

SpartanGold uses the standard hash proof-of-work as seen in many Bitcoin variants, and this project seeks to replace that with a more interesting and potentially useful proof-of-work involving prime numbers. This change does not solve any of the challenges facing cryptocurrencies like privacy, ASIC reliant mining, or vulnerability to majority attacks to name a few. An argument could be made that this is a step towards meaningful work to have less energy waste. While this is nowhere near the impact of Folding@home[1], a distributed computing project to fold protein molecules for scientific research, prime number chains are still being collected and studied with records broken annually. The cryptocurrency at the forefront of discovering these record breaking prime chains as well as being the inspiration for this project is none other than Primecoin[2].

Primecoin

The original Primecoin whitepaper was written and implemented in 2013 by Sunny King[3]. The paper detailed a fork of Bitcoin using prime number chains as alternative proof-of-work. King also modified mining difficulty scaling and minting rate based off their previously created proof-of-stake cryptocurrency, ppcoin.

1. Prime Chains

Primecoin generates and accepts three types of primes number chains: Two kinds of Cunningham chains[4] and bi-twin chains[5]. A Cunningham chain of the first kind starts with a prime number p_1 and every prime following in the chain of length k satisfies $p_i = 2p_{i-1} + 1$. Each prime number in this chain except for the last also happens to be a Sophie Germain prime, defined as a prime number p where $2p + 1$ is also prime. The equation for any prime number p_i within the chain and an example chain is given

$$p_i = 2^{i-1}p_1 + (2^{i-1} - 1) \tag{1}$$

$$2, 5, 11, 23, 47$$

The second kind of Cunningham chain similarly starts with some prime number p_1 and every prime following in the chain satisfies $p_i = 2p_{i-1} - 1$. This is likewise represented as an equation for any prime within the chain as

$$p_i = 2^{i-1}p_1 - (2^{i-1} - 1) \tag{2}$$

$$19, 37, 73$$

A bi-twin chain starts with a non-prime origin number n where $n - 1$ and $n + 1$ must be primes. From there, the chain must increment in pairs where at each step, the next two chain primes must satisfy both kinds of Cunningham chain patterns using the non-prime origin as a starting point. $n - 1$ is the starting prime for Cunningham chains of the first kind with successive primes satisfying equation 1, and $n + 1$ starts the Cunningham of the second kind with successive primes satisfying equation 2. The chain length is always even due to the chained primes being added in pairs. Primecoin allows an extra prime in bi-twin chains if the Cunningham chain of the first kind is longer than the second to enable odd length bi-twin chains. Representing each prime in a bi-twin chain as a function of the origin number n instead of a starting prime is shown as

$$p_i = 2^{i-1}n - 1 \tag{3}$$

$$p_i = 2^{i-1}n + 1 \tag{4}$$

$$5, 7, 11, 13$$

2. Proof Generation and Verification

Primecoin uses the hash of the previous block as a starting point for proof generation of the current block to prevent reuse of proofs. Specifically the current block header is hashed, which includes everything in the block except the proof certificate. The goal in hashing proof-of-work is to be less than some small hash value target. For Primecoin, a mined prime chain must have length greater than or equal to a target length. The proof in hashing proof-of-work is a nonce, and similarly in Primecoin the proof is also a number. The proof number is a multiplier that is multiplied with the block header hash to create the chain origin number. The miner then uses a combination of Fermat's little theorem and Henri Lifchitz's generalization of the Euler-Lagrange theorem as primality tests to find probable primes along both types of Cunningham chains starting from $n - 1$ and $n + 1$. If no suitable chain is found with target length from the chain origin number, the proof multiplier is incremented to try again. Both primality tests are short and incomplete, but proving primality with large numbers is too time consuming. The primes in the chain are instead tested to be probable primes.

Fermat's little theorem can be used to test if a number is prime based off the notion that if p is a prime number, $a^{p-1} - 1$ is a multiple of p where a is any integer. The most common way to use this is to set $a = 2$ and check if p is prime with modular arithmetic if it satisfies

$$2^{p-1} \equiv 1 (\text{mod } p) \tag{5}$$

Lifchitz's primality tests show if the next number in a Cunningham chain is a probable prime. Omitting the proof, the next number in a Cunningham chain of the first kind after p could be prime if it satifies either of

$$p \equiv 1(\text{mod } 4); 2^p + 1 \equiv 0(\text{mod } 2p + 1) \tag{6}$$

$$p \equiv 3(\text{mod } 4); 2^p - 1 \equiv 0(\text{mod } 2p + 1) \tag{7}$$

Proof verification by any client receiving the completed block is done by computing the chain origin number from multiplying block header hash by the proof multiplier number, and checking that there is a prime chain of any type with at least target length. Each prime number is tested for probable primality.

The Primecoin implementation uses certain limits and bounds to keep mining meaningful and within expectation. It's possible for the block header hash to be a very small value, leading to faster compute and less interesting results. To counteract this, a block header hash is only valid if it exceeds 2^255 where the hash is a double SHA-256 (run through SHA-256 twice). If a block header hash does not satisfy this check, a nonce is added to the block to get a new header hash. Chain origin number and primes in the chain are also subjected to a range check to prevent boring small primes and compute expensive extremely large primes.

Choosing a multiplier to find a good chain origin is also improved beyond simply incrementing the multiplier and applying brute force. Note that the chain origin must be non-prime if both neighboring numbers are to be prime (with an exception of $n = 2$ but that is not possible here). Any multiplier that could result in an odd product is automatically ruled out. Without theorems or proofs, the Primecoin paper states that the use of primorial (prime factorial) numbers as part of the proof multiplier can influence the chain origin to be better suited to contain a prime chain. A primorial of a number is the product of all prime numbers less than or equal to the number. The Primecoin implementation starts each proof search with a primorial number as the base multiplier and also uses a prime sieve to eliminate bad chain origin candidates. A great majority of the current prime chain records all use origin numbers generated from using a primorial number as a multiplier[7].

3. Other Changes from Bitcoin

Bitcoin regulates block production rate and minting through linear difficulty scaling of the target hash and reducing mining rewards over time with a hard limit on total currency. Primecoin sought to keep the linear difficulty scaling to regulate block rate, but needed a more complex method of modifying difficulty. Simply increasing or decreasing the target prime chain length does not result in linear scaling. King decided that for any prime chain the next number in the chain that is not prime would be put through Fermat's primality test, and the remainder of the modulo is used to measure difficulty of the prime chain. King describes a fractional chain length value with the follow equation where k is length of current prime chain, p_k is the next number in the chain that is not prime, and r is the remainder from Fermat's primality test. This is combined with features from

King's previous cryptocurrency ppcoin to provide linear difficulty scaling.

$$d = k + \frac{p_k - r}{p_k} \tag{8}$$

Primecoin in general has a slower block production rate than Bitcoin, but processes ten times the number of transcation per block. It also keeps its currency scarce and stable by regulating minting rewards by Moore's Law instead of having a diminishing reward that goes until available currency is depleted.

Implementation Details

The integral part of Primecoin, the prime chain proof-of-work, has been implemented as an alternative proof-of-work for SpartanGold. Since SpartanGold is already simplified and missing some Bitcoin features like UTXOs and scaling difficulty, none of those same features were ported over from Primecoin to SpartanGoldPrime. The implementation is a collection of subclasses that extend their original parent counterparts with `Block.js` and `Miner.js` being the most important as they deal directly with proofs. Other subclasses were made if the parent class were to modify or access the block in a way that may not account for new proof properties. Extra tests were written for use with mocha, driver.js was updated, and `tcpMiner.js` was extended to also export discovered prime chains to JSON. All prime numbers are presented as `BigInteger` from the `jsbn` npm package.

1. prime.js

A collection of constants and functions to aid in prime number testing and finding prime chains. Primes in the potential chains are tested with the two aforementioned primality tests as well as the Miller-Rabin primality test (based off Fermat's primality test), courtesy of `jsbn.BigInteger`. Includes a method to find all types of chains from a chain origin for miners, and more specific method to test if a given chain origin and chain type meets given chain length.

2. primeBlockchain.js

Contains new constants for prime chain proof-of-work including chain type string literals, block header hash target of $2^2 55$, and default prime chain length target of 3. Updated `deserialize()` to unpack new block properties correctly.

3. primeBlock.js

Replaced the hash proof nonce with new properties: `primeNonce`, `primeMultiplier`, `primeChainLength`, `primeCchainType`. The last two properties are not necessary for Primecoin, but it is included here to further increase client verification of the proof. Proof verification method updated to check prime chains, existing serializing method updated with new properties, and new method to provide hash of only block header.

4. primeMiner.js

`findProof()` updated to find a proper block header hash and then find a prime chain with suitable target length. `receiveBlock()` updated to use new method for proper block deserialization.

5. primeClient.js

Same as above, `receiveBlock()` uses new method for block deserialization.

6. tcpPrimeMiner.js

Updated version of `tcpMiner.js` that includes user input option to export discovered prime chains in JSON format with user specified file name. Output JSON only includes information from confirmed blocks from that miner client and needs to be built from chain origins and multipliers into actual prime chain lists by `buildPrimeChains.js` script.

7. buildPrimeChains.js

Takes in a file as command line argument and expects proper JSON format according to the previous script's prime chain JSON dump. Builds the actual prime chain lists from chain origin, multiplier, chain length, and chain type. Overwrites the input file to a JSON file with format similar to what Primecoin uses in their prime chain online explorer.

8. driver.js

Updated to use new subclasses and total runtime increased from 5 seconds to 60 seconds.

9. test.js

Extra tests added for new methods.

Results

1. Performance

Prime chain proof-of-work was definitely much more costly than hashing, below is a table of average time for different target chain lengths regarding finding and verifiying chains/hashes.

Action	Hashing	Length = 1	Length = 2	Length = 3
Find Proof	15ms	90ms	3s	50s
Verify	0.01ms	0.45ms	0.85ms	2ms

2. Bugs

Miners sometimes spend too long trying to mine and get very behind on current blocks, but with enough time the miners will eventually receive all missing blocks.

3. Future Work

Prime chain JSON data could be visualized nicely, prime chain difficulty could be scaled, chain origin generation could be improved with primorials and prime sieves, and a mining pool client could be made.

References

[1] Maxwell Zimmerman, et al., "SARS-CoV-2 simulations go exascale to capture spike opening and reveal cryptic pockets across the proteome", 2020.
doi: https://doi.org/10.1101/2020.06.27.175430

[2] Primecoin [Online].
Available: https://primecoin.io/

[3] Sunny King, "Primecoin: Cryptocurrency with prime number proof-of-work", 2013.
Available: https://primecoin.io/bin/primecoin-paper.pdf .

[4] "Cunningham Chain" [Online].
Available: https://primes.utm.edu/glossary/page.php?sort=CunninghamChain

[5] Eric Weisstein, "Bitwin Chain", MathWorld–A Wolfram Web Resource [Online].
Available: https://mathworld.wolfram.com/BitwinChain.html

[6] Henri Lifchitz, "Generalization of Euler-Lagrange theorem and new primality tests", [Online].
Available: http://www.primenumbers.net/Henri/us/NouvTh1us.htm

[7] Dirk Augustin, "Cunningham Chain records", [Online].
Available: http://primerecords.dk/Cunningham_Chain_records.htm