MEGACOIN

Megacoin: A Peer-to-Peer Electronic Cash System

https://megacoin.eu/
https://github.com/LIMXTEC/Megacoin
https://github.com/LIMXTEC/Megacoin/releases

MEGACOIN

THE MEGACOIN BLOCKCHAIN

First Block: 2013-06-01 09:34:59
Max supply: 42 million MEC
Blocktime: 2.5 minutes
Algorithm: Scrypt
Retarget: Core Shield
RPC: 7950, P2P: 7951
Clientversion: 0.15.0.3
Blocksize: 10 MB + Segwit
Reward per Block: 6.25 MEC

MEGACOIN IS A NEW TYPE OF DECENTRALIZED CURRENCY

Not minted or endorsed by any nation, but by users across the globe. Anywhere there is internet, you can send or receive Megacoins. Even in outer space. There are very little rules for Megacoin. To prevent inflation only a limited supply of 42 million Megacoins will ever exist. Megacoins are mined by users at a set rate for decades into the future. You can get Megacoins for free by mining or trading for them on an exchange.

No need for Wiring or Carrying Cash across borders. You can send Megacoin to your relatives or business partners to the other side of the world. No hoops to jump through, no questioning. Just hit send.

Unlike Banks which can require a Credit Check or Social Security Number, Megacoin doesn't ask for anything except a smile. A Megacoin account is completely free & with low transaction fees. Transactions are completed within seconds, worldwide!

Megacoin has a class of decentralized workers called miners to keep the Paymentsystem secure. Specialized Cloud Computers run encryption software to distribute Megacoin and verify all transactions on the payment network. Megacoin is absolutely secure against forgery. Everyone can become a miner, it´s very easy to rent or buy hardware and join a miningpool.

Megacoin is open source software and released under the terms of the MIT License. The global network uses peer-to-peer technology and consists of dozens strategic nodes all over the world. Megacoin is global, it faces no boundaries and connects people even from the most faraway places on earth.

1. DESCRIPTION

Megacoin is a proof-of-work, bitcoin-based cryptocurrency that was developed as an alternative to the Bitcoin network. Megacoin has five major differences in using all the Bitcoin Improvement Proposals (BIP) developed by Bitcoin. It has Segregated Witness enabled, Bloom Filters and the Lightning Network is compatible with Megacoin.

1.) While Bitcoin is limited to 21 million coins, Megacoin is limited to 42 million, twice the amount of potential coins that can be circulated.

2.) The Bitcoin protocol creates new blocks in 10-minute intervals, whereas Megacoin is four times faster and produces new blocks every 2.5 minutes.

3.) The blocks produced by Bitcoin have a maximum of 1 MB per block to process transactions, Megacoin with its block size of 10 MB can not only produce 10 times larger blocks, it can process up to 40 times as many transactions as in Bitcoin network.

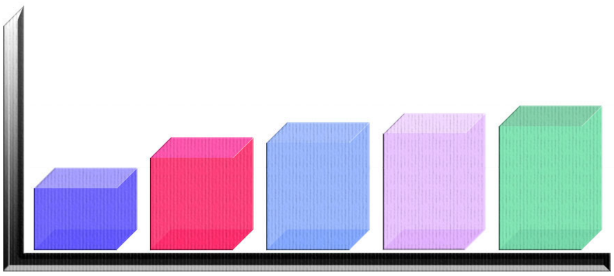
4.) For purposes of proof-of-work the Bitcoin network uses highly parallelized SHA256 hash function. Megacoin in turn uses Scrypt instead of SHA256 performance of which is dependent on access to large reserves of memory so that running several Scrypt-processes on current generation GPUs is highly resource-intensive.

5.) In order to generate blocks in the desired time, bitcoin adjusts the proof-of-work difficulty every 2016 blocks, while Megacoin adjusts the difficulty every 64 blocks between 1% and 15%. This blocktime system is called "Core Shield".

2. CURRENCY DISTRIBUTION

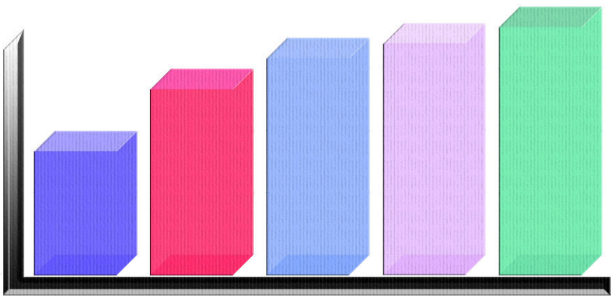
INITIAL ADOPTION PHASE

Megacoin has been designed to handle rapid growth and a stable future. Only 42 Million Megacoins will ever exist. During the freely joinable Initial Adoption Phase (IAP) which lasts nearly half a year, 21 Million Megacoins will come into existance. Every Megacoin block has a reward. Here the block rewards are in groups of 21,000 blocks steping down from 500, 250, 125, 75, 50. Rapid growth during the IAP is fueled by a balanced amount of increased rewards for miners who secure the Megacoin network.



CIRCULATION BY YEAR

Around the year 2014 to several decades in the future, the remaining 21 Million Megacoins will be distributed at a similar step-down rate. Every 420,000 blocks (2 years), The reward starts at 25 and is halved each period. 10.5 million Megacoins come from first 2 Years of 420,000 blocks. While the Megacoin release rate decreases as time moves forward, the market value of Megacoins will adjust accordingly.

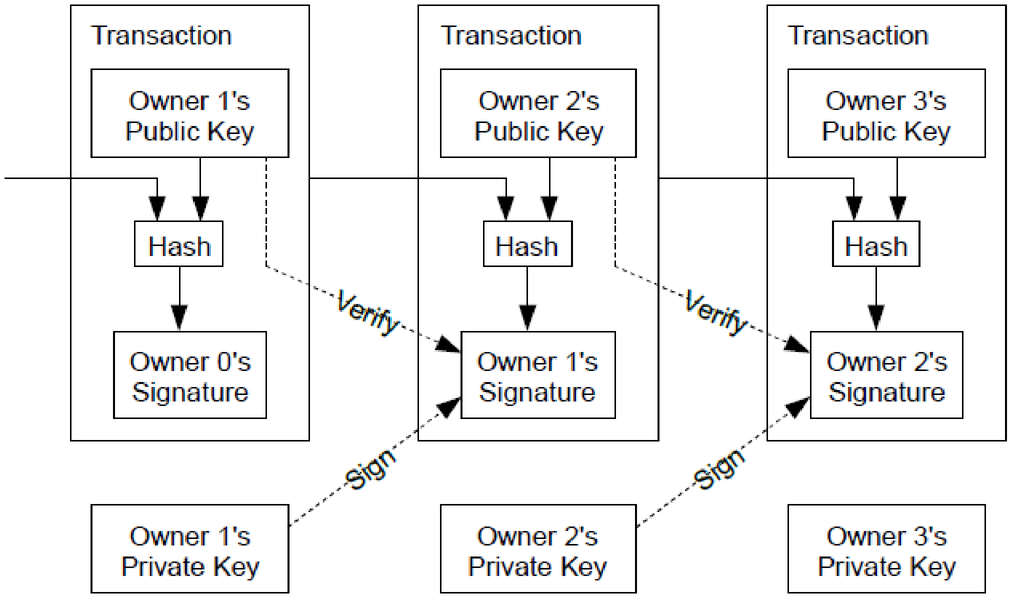


3. NO ICO FUNDRAISING

Another unique factor for Megacoin is that there was no ICO funding. Therefore, there is no inherent risk of being in breach of securities laws in the United States or elsewhere. Many other projects that have launched elaborate ICOs or are attached to larger companies, may find themselves in the crosshairs of legislative and federal authorities in various jurisdictions around the planet. This is just one reason why refusing to launch Megacoin as an ICO presents a major benefit for adopters of the coin. There are no ‘private investors’ or individuals that were able to move in and acquire adisproportionate distribution of coins simply because they possessed more money than others.

4. TRANSACTIONS*

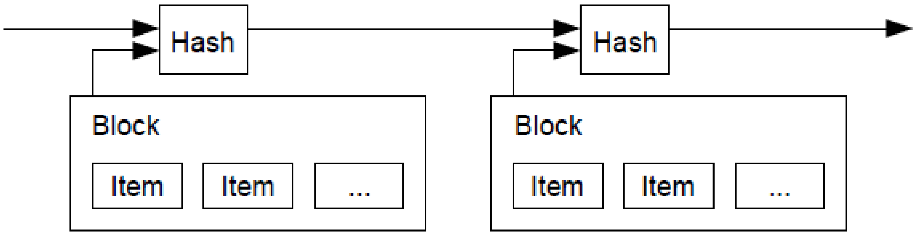
We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.



The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank. We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

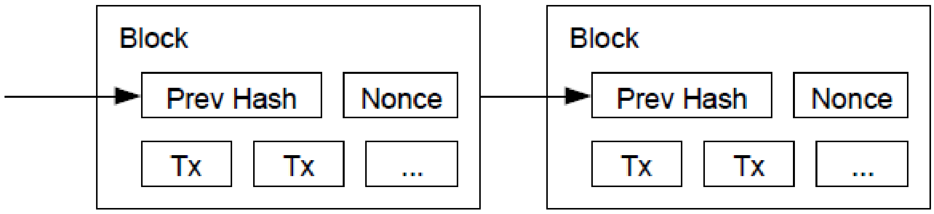
5. TIMESTAMP SERVER*

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



6. PROOF-OF-WORK*

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash. For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the GPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-GPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of GPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added. To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

7. NETWORK*

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

8. THE LIGHTNING NETWORK

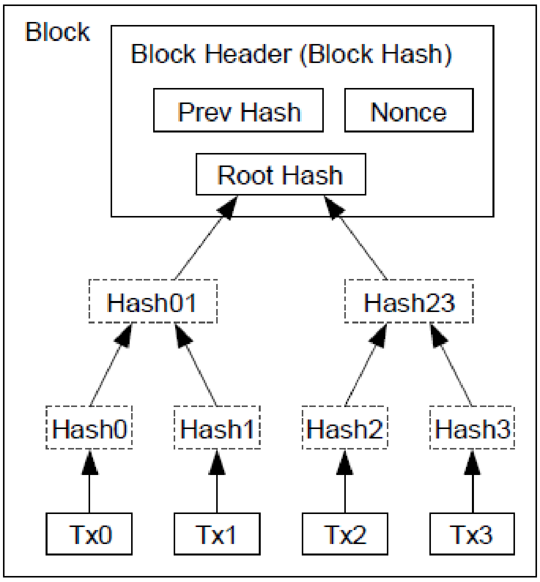
The Lightning Network is a transfer network operating at a layer above the Megacoin blockchain using smart contract functionality in the blockchain to enable instant payments across a network of participants. This enables improvements of several orders of magnitude in transaction throughput by moving the majority of transactions outside the consensus ledgers into Payment channels. This allows millions to billions of transactions per second across the network. A capacity that blows away legacy payment rails. This is made possible by supporting on-chain scripts in which parties enter into bilateral stateful contracts, in which the state can be updated by sharing a digital signature and can be closed by publishing evidence onto the blockchain. The Lightning Network allows for exceptionally low fees. For a low-value transaction, the Lightning Network is the silver bullet. It allows for new kinds of commerce. By opening a payment channel with many parties, participants in the LN can become a focal point for routing the payment of others leading into a fully connected payment channel. The payments are enforced using a script which enforces the atomicity via decrementing time-locks. Another benefit is the possibility of atomic cross-chain transactions, enabling users to trade megacoin, bitcoin, litecoin and other Segwit coins instantaneously.

9. INCENTIVE*

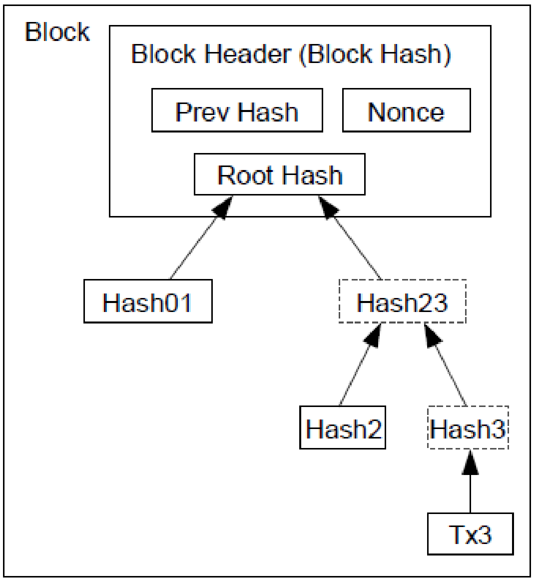
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended. The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free. The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

10. RECLAIMING DISC SPACE*

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



Transactions Hashed in a Merkle Tree

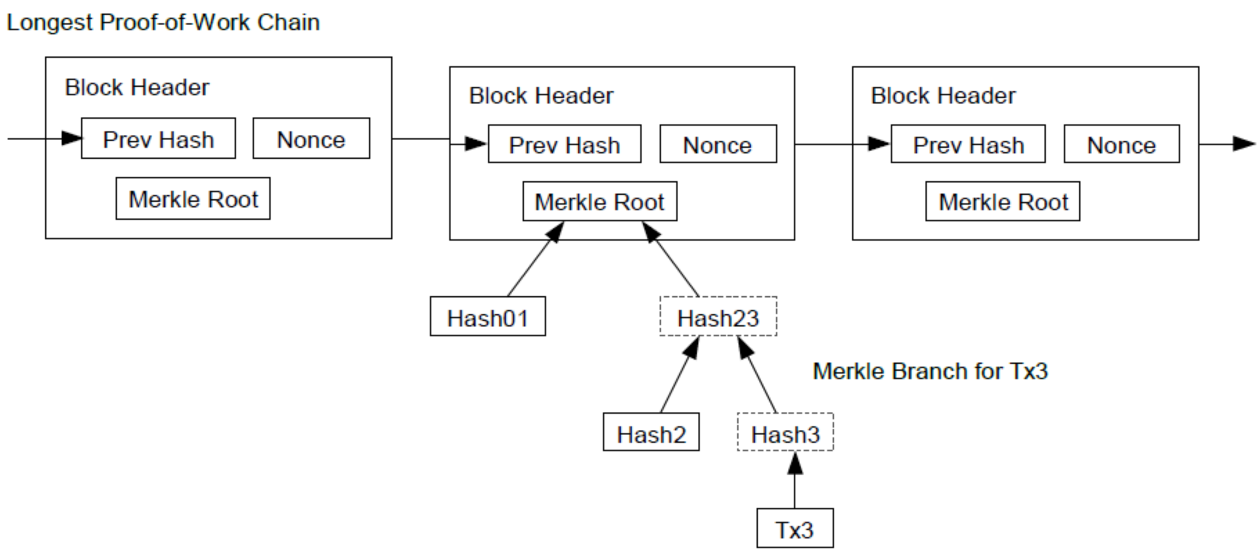


After Pruning Tx0-2 from the Block

A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 2.5 minutes, 80 bytes * 24 * 24 * 365 = 16.8 MB per year. With computer systems typically selling with 4 GB of RAM as of 2018, and Moore's Law predicting current growth of 1.2 GB per year, storage should not be a problem even if the block headers must be kept in memory.

11. SIMPLIFIED PAYMENT VERIFICATION*

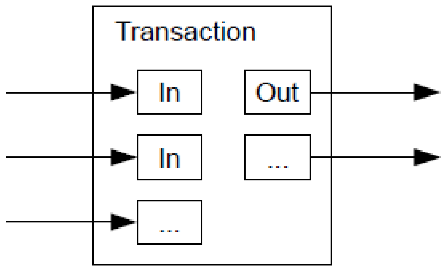
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

12. COMBINING AND SPLITTING VALUE*

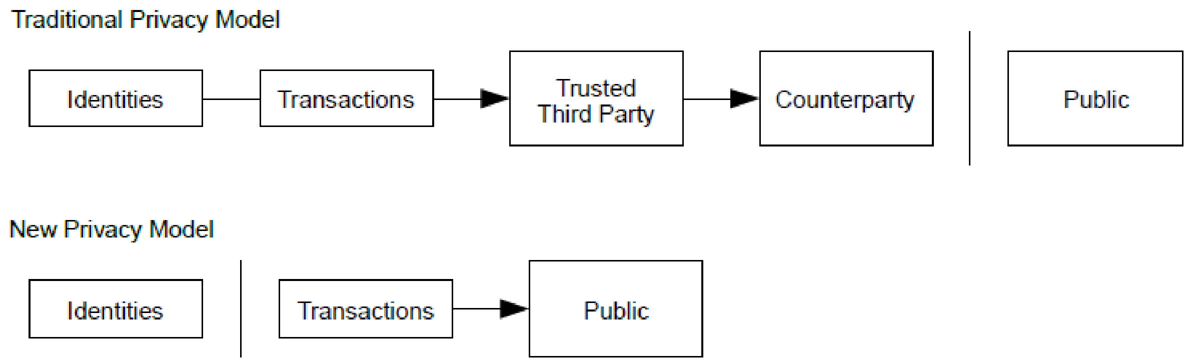
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

13. PRIVACY*

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

14. CALCULATIONS*

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent. The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1. The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:

p = probability an honest node finds the next block
q = probability the attacker finds the next block
qz = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{Bmatrix} 1 & if\ p \le q \\ (q/p)^z & if\ p > q \end{Bmatrix}$$

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind. We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late. The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction. The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z\frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{Bmatrix} (q/p)^{(z-k)} & if\ k \le z \\ 1 & if\ k > z \end{Bmatrix}$$

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
        double p = 1.0 - q;
        double lambda = z * (q / p);
        double sum = 1.0;
        int i, k;
        for (k = 0; k <= z; k++)
        {
                double poisson = exp(-lambda);
                for (i = 1; i <= k; i++)
                poisson *= lambda / i;
                sum -= poisson * (1 - pow(q / p, z - k));
        }
        return sum;
}
```

Running some results, we can see the probability drop off exponentially with z.

```
q=0.1
z=0 P=1.0000000
z=1 P=0.2045873
z=2 P=0.0509779
z=3 P=0.0131722
z=4 P=0.0034552
z=5 P=0.0009137
z=6 P=0.0002428
z=7 P=0.0000647
z=8 P=0.0000173
z=9 P=0.0000046
z=10 P=0.0000012
```

q=0.3
z=0 P=1.0000000
z=5 P=0.1773523
z=10 P=0.0416605
z=15 P=0.0101008
z=20 P=0.0024804
z=25 P=0.0006132
z=30 P=0.0001522
z=35 P=0.0000379
z=40 P=0.0000095
z=45 P=0.0000024
z=50 P=0.0000006

Solving for P less than 0.1%...

P < 0.001
q=0.10 z=5
q=0.15 z=8
q=0.20 z=11
q=0.25 z=15
q=0.30 z=24
q=0.35 z=41
q=0.40 z=89
q=0.45 z=340

15. CONCLUSION*

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

16. MEGACOIN CORE TEAM

Lead Developer:

Christian Knöpke, Germany
https://github.com/LIMXTEC

Administrator:

Sebastian Krause, Germany
https://github.com/LIMXTEC

Open Altcoin Github

Welcome to Limxtec - Your place for your Source

https://bitcointalk.org/index.php?topic=2710407.msg27721865#msg27721865

Megacoin started on 2013-05-29 with a pre-announcement on Bitcointalk.org written by an anonymous user called Kimoto (Dr. Kimoto Chan), Megacoins initial developer. Kimoto created in the following months the KIMOTO GRAVITY WELL for Megacoin, a difficulty retarget system to prevent multipool influence on the blockchain. In june 2015, two years later Kimoto left the project with a short notice on Bitcointalk.org. In 2017 the LIMXTEC group started to revive Megacoin with a newer Megacoin version based on Bitcoin 0.10. In Q2 2018 the Megacoin finally got an update to Bitcoin version 0.15 with a new difficulty retarget system called "Core Shield" and a maximum Blocksize of 10 MB.

17. REFERENCES

[*] "The Bitcoin Whitepaper of Satoshi Nakamoto", https://bitcoin.org/bitcoin.pdf, 2008.

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

18. DISCLAIMER