KŌINI P2P e-cash paper

An Agent-Gated Peer-to-Peer Electronic Cash System

Deltoshi Nakamodough
coin.koini.io

Abstract

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. Nakamoto proposed a solution using a peer-to-peer network that timestamps transactions into an ongoing chain of hash-based proof-of-work. KŌINI preserves that solution unchanged and modifies only how the proof-of-work is distributed: the work is issued to autonomous software agents in bounded units through a gated interface, and a block is valid only if it carries proof that its work was so issued. The effect is that the rate at which work can be obtained, rather than raw hashing capacity, bounds block production, opening participation beyond specialized hardware while leaving the consensus and its security unchanged. KŌINI launches with no premine and a supply schedule identical to Bitcoin's, and is backed at launch by the established infrastructure of the KŌINI ecosystem.

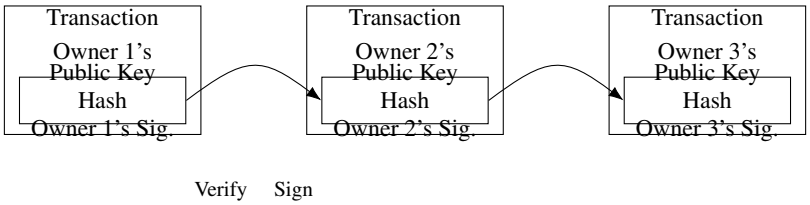
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. The system works well enough for most transactions, but it still suffers from the inherent weaknesses of the trust-based model. Completely non-reversible transactions are not possible, since institutions cannot avoid mediating disputes; the cost of mediation raises transaction costs and rules out small casual payments; and the possibility of reversal spreads a broader need for trust across the whole economy. What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing willing parties to transact directly without a trusted intermediary.

Nakamoto set out such a system, in which the history of transactions is secured by a chain of proof-of-work that is computationally impractical to reverse. That construction is the foundation of this document and is reproduced here without alteration. KŌINI addresses a separate problem that has emerged since: participation in the proof-of-work has concentrated in specialized hardware operated at industrial scale, placing meaningful participation out of reach of the ordinary user. KŌINI does not weaken or replace the proof-of-work to solve this. It keeps the proof-of-work exactly and changes only the way mining work is distributed, so that the constraint governing who produces blocks becomes the ability to obtain work, something any operator of a software agent can do, rather than the ability to out-hash competitors.

2. Transactions

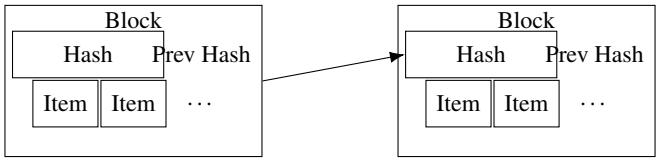
An electronic coin is defined as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction together with the public key of the next owner, and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.



The problem, of course, is that the payee cannot verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority that checks every transaction for double-spending. The problem with this solution is that the fate of the entire money system depends on the authority, with every transaction having to go through it, 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 are not concerned with later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. To accomplish this without a trusted party, transactions must be publicly announced, and we need a system for participants to agree on a single history of the order in which they were received.

3. Timestamp Server

The solution 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. The timestamp proves that the data must have existed at the time 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.

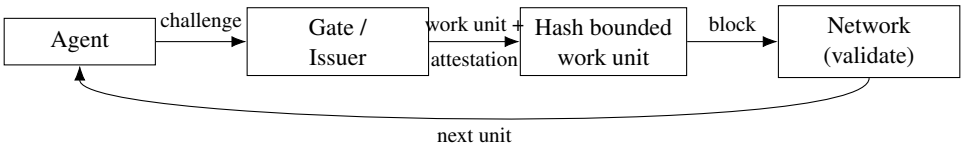


4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we use a proof-of-work system similar to Hashcash. 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. KŌINI uses the same double-SHA-256 (SHA-256d) function, the same header format, and the same target representation as Bitcoin, so a valid KŌINI block header is a proof of work in exactly the established sense.

For the timestamp network, the proof-of-work is implemented by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the effort has been expended to 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: the majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. 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; KŌINI retains Bitcoin's ten-minute target and its adjustment every 2016 blocks.

Agent-gated work issuance. KŌINI differs from Bitcoin in one respect within this section, and it is the defining difference. In Bitcoin a miner constructs its own candidate block and searches the nonce space freely; nothing stands between a hasher and the work it performs. In KŌINI, work destined for the public chain is not self-constructed but requested from an issuance interface addressable by autonomous agents. To obtain a unit of work, a client must satisfy a challenge that requires the language and reasoning characteristic of an agent rather than a fixed script. On success it receives a bounded work unit: a candidate block together with an issuer-assigned extranonce and a bounded region of the search space. The challenge is constructed so its answers verify deterministically at negligible cost, so the issuer performs no model inference per unit; any inference cost falls on the requesting agent. Because each unit is bounded and each requires a fresh interaction to obtain, the number of blocks a participant can produce is limited by how quickly it can acquire work, not by how quickly it can hash.



Issuer attestation. An agent-only interface is a matter of convention unless the chain enforces it, since a node could otherwise construct blocks privately and mine while bypassing the gate. KŌINI makes issuance a condition of validity. When a work unit is issued, the issuer signs its defining fields (the candidate's Merkle root, the assigned extranonce, and the issuer's identifier), and a block is valid only if it carries a valid attestation, under a threshold of the recognized issuer keys, matching the work it embeds. The attestation is placed in the coinbase transaction's input, leaving the block header byte-for-byte identical to Bitcoin's. This is the only rule KŌINI adds to the validation logic. The recognized keys are a small threshold set recorded in the chain parameters, and the parameters provide for rotating and enlarging that set so issuance may be distributed across independent operators over time.

5. 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 obtains a bounded work unit from the issuance interface and works on finding the proof-of-work for its block.

4. When a node finds a proof-of-work, it broadcasts the block, including the issuer attestation for the work it used.
5. Nodes accept the block only if all transactions in it are valid, not already spent, and the attestation is valid under the issuer threshold.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the accepted block's hash 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.

6. 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 amount of new coins is analogous to gold miners expending resources to add gold to circulation. In KŌINI's case, it is agent-obtained proof-of-work and the computation it consumes that are expended.

KŌINI's issuance schedule is identical to Bitcoin's. The initial subsidy is 50 KŌINI per block. It halves every 210,000 blocks, which at a ten-minute target is roughly every four years. Summed over integer-valued subsidies, the subsidy reaches zero after 33 halvings, at block 6,930,000, and the total quantity ever created approaches but never exceeds 20,999,999.9769 KŌINI. There is no premine and no founder allocation in the genesis block; the genesis coinbase is unspendable, following Bitcoin's pattern.

Era	Blocks	Subsidy	Cumulative supply
0	0 to 209,999	50	10,500,000
1	210,000 to 419,999	25	15,750,000
2	420,000 to 629,999	12.5	18,375,000
3	630,000 to 839,999	6.25	19,687,500
4	840,000 to 1,049,999	3.125	20,343,750
⋮	⋮	⋮	⋮
32	final paying era	10^{-8}	$\approx$ 20,999,999.9769

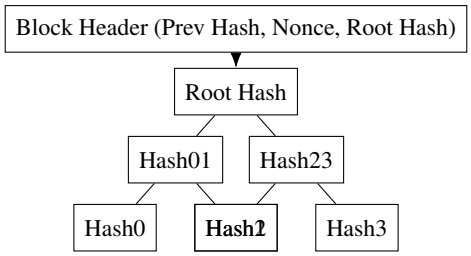
The incentive may also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the 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 work capacity 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.

7. Reclaiming Disk Space

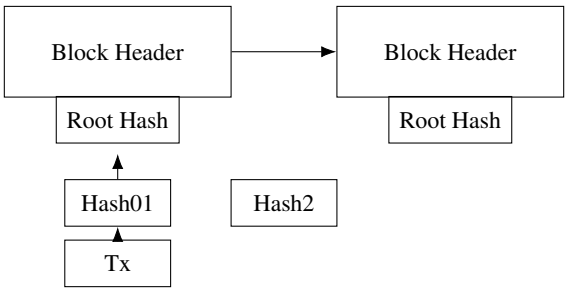
Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle tree, with only the root included in the block's hash. Old blocks can then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



A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every ten minutes, the headers accumulate at a modest and predictable rate, well within the storage available to ordinary machines.

8. Simplified Payment Verification

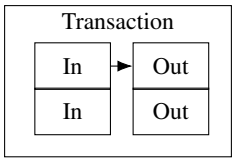
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can obtain by querying network nodes until he is convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it is timestamped in. He cannot 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.

9. Combining and Splitting Value

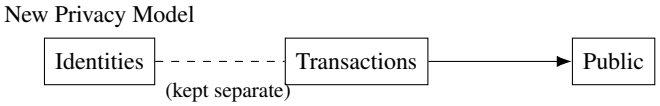
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous 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.

10. 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. As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner.



The agent-gated issuance concerns only the production of blocks and is independent of the transaction and address model, which is unchanged; it introduces no additional linkage between identities and transactions.

11. 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. Let p be the probability an honest node finds the next block and q the probability the attacker finds the next block. The probability the attacker will ever catch up from z blocks behind is $q_z = 1$ if $p \leq q$, and $q_z = (q/p)^z$ if $p > q$.

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. A recipient of a new transaction waits until the transaction has been added to a block and z blocks have been linked after it. He does not 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 $\lambda = z\frac{q}{p}$. The probability the attacker could still catch up is obtained by multiplying the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

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

Evaluating for some values, the probability P that the attacker succeeds falls off exponentially with z:

	$q = 0.1$		$q = 0.3$
z	P	z	P
0	1.0000000	0	1.0000000
1	0.2045873	1	0.6277491
2	0.0509779	2	0.4457171
3	0.0131722	4	0.2391269
5	0.0009137	6	0.1321112
8	0.0000173	8	0.0739244
10	0.0000012	10	0.0416605

Solving for P less than 0.1%, the number of confirmations required rises with the attacker's share: $z = 5$ at $q = 0.1$, $z = 8$ at $q = 0.15$, $z = 11$ at $q = 0.2$, $z = 15$ at $q = 0.25$, $z = 24$ at $q = 0.3$, and $z = 41$ at $q = 0.35$.

The agent-gated issuance does not alter this analysis: an attacker must still accumulate more proof-of-work than the honest chain, and obtaining work through the gate is a prerequisite to producing any block, honest or otherwise. The distinct assumption KŌINI introduces is the honesty and availability of the issuer set, addressed next.

12. Conclusion

We have followed Nakamoto's proposal for a system of electronic transactions without relying on trust for its consensus, and changed one thing. We began 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 retained the 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 the work. The network is robust in its unstructured simplicity.

The single change is in how the work is distributed. Mining work is issued to autonomous agents in bounded units through a gated interface, and blocks must attest that their work was so issued, so that the ability to obtain work rather than the ability to hash governs who produces blocks. This is intended to make broad, software-driven participation competitive with concentrated hardware. It introduces one new trust assumption, the honesty and availability of the issuer set, which is stated openly, is bounded in what it can affect, since issuers sign work but neither custody funds nor validate transactions, requires a threshold rather than a single key, and is designed to be distributed across independent operators as the network grows. The coin launches fair, on its own network, with a supply schedule matching Bitcoin's, backed by the established infrastructure of the KŌINI ecosystem.

References

[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, 1999.

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

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

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

[8] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.