

Timecoin: A Peer-to-Peer Electronic Cash System with Post-Quantum Authorization

Williem Doe
jsonrpc@icloud.com

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, and would remain sound after a cryptographically relevant quantum computer exists. Digital signatures provide part of the solution, but the elliptic-curve signatures used by Bitcoin are not expected to survive Shor's algorithm. We propose Timecoin, a unit of account on a new chain called Timechain. The chain is a code descendant of Bitcoin's transaction and proof-of-work design, with its own genesis block and no shared history. Ordinary authorization is an ML-DSA-65 signature under NIST FIPS 204. Each address also commits to an SLH-DSA recovery path under FIPS 205, whose only hardness assumption is a cryptographic hash. Proof-of-work remains double SHA-256. Grover's algorithm gives only a quadratic speedup against that hash, so the mining function is not the first thing that has to change. The familiar incentives, the Merkle tree, and the longest-chain rule are kept. What changes is the lock on the coin.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties. Nakamoto's Bitcoin removed that party for the ordering of payments by publishing a chain of hash-linked blocks and by requiring each spend to carry a signature [1]. The construction still stands. Its authorization primitive does not have the same horizon.

Bitcoin locks coins with ECDSA, and later Schnorr, on the secp256k1 curve. Shor's algorithm solves the discrete-logarithm problem on that curve in polynomial time on a sufficiently large quantum computer [2]. A public key that has been revealed — in a spent output, a reused address, or an old pay-to-public-key output — would then be enough to forge a spend. Grover's algorithm, applied to SHA-256, is a different and smaller problem: it square-roots the search, which a longer hash or more work can absorb [3]. The urgent defect is the signature, not the timestamp.

Several migration drafts exist for Bitcoin itself. They have to carry an installed base, a miner vote, and coins whose keys are already public. Timechain does not. It is a new ledger. Timecoin (ticker TIC) is its unit. The name Timechain is the one Satoshi used in early notes for the block chain; it is reused here as homage, not as a claim on Bitcoin's chain, ticker, or genesis. Holders of bitcoin receive nothing on Timechain. There is no premine. The genesis coinbase is unspendable, as in Bitcoin, and issuance begins with the first mined subsidy.

The design goal is narrow. Keep the accounting model that has been run in public since 2009. Replace the lock with two standardized post-quantum signatures, one fast and one conservative, committed together so that a break of the lattice assumption does not strand the coin. Publish the rules before the first block, rather than asking an existing economy to soft-fork them in.

2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by signing a hash of the previous transaction and the public-key hash of the next owner, and by appending this signature to the end of the coin. A payee can verify the signatures to verify the chain of ownership. The picture is

Nakamoto's, with the signature algorithm changed.

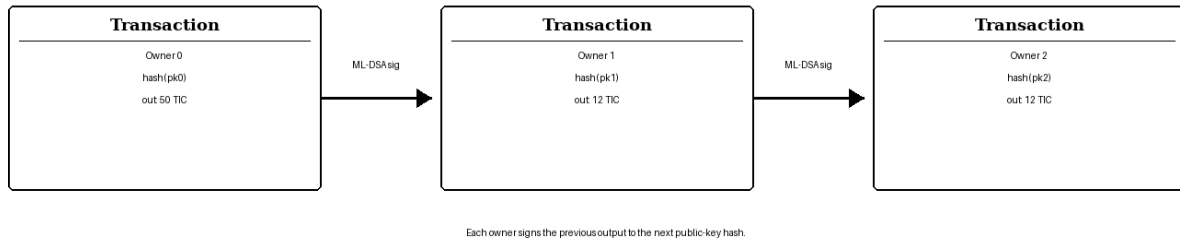


Figure 1. Timechain transactions with post-quantum signatures.

The problem of course is that the payee cannot verify that one of the owners did not double-spend the coin. The usual solution is a trusted central authority, or mint, that checks every transaction. We follow Bitcoin: transactions are announced publicly, and nodes agree on one history. The only structural addition is that the signed message commits to a signature algorithm identifier, so a lattice signature cannot be replayed into a hash-based program, and a Timechain transaction cannot be replayed onto Bitcoin.

Outputs lock a 32-byte witness program. That program is the SHA-256 hash of a two-leaf script tree. The key-path leaf is an ML-DSA-65 public key. The recovery leaf is an SLH-DSA-SHA2-128s public key. Neither key is revealed by the address. A spend reveals the leaf it uses, the public key, the signature, and the Merkle partner hash that reconnects the leaf to the witness program.

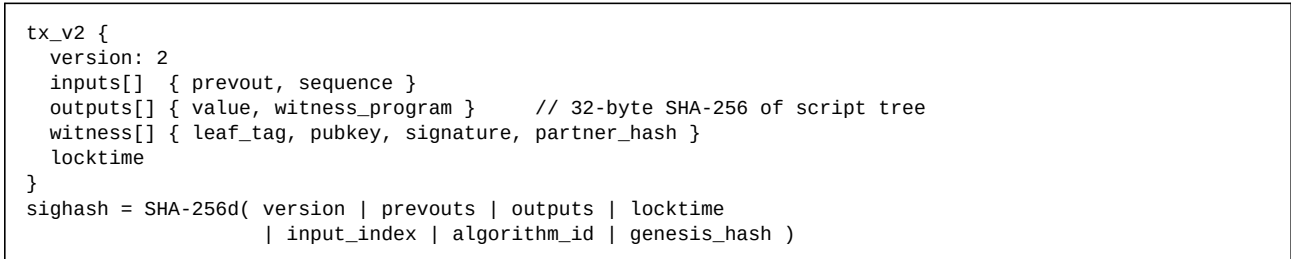


Figure 2. Version-2 transaction and domain-separated signature hash.

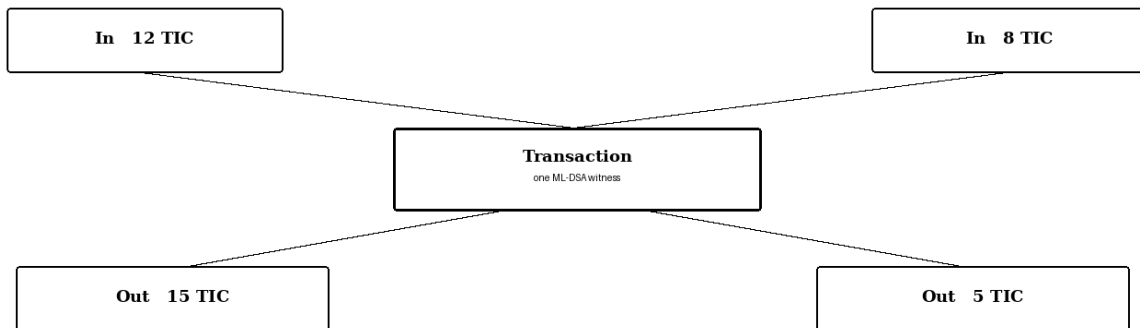


Figure 3. Combining and splitting value. Two inputs, two outputs, one witness. The fee is the difference.

3. Timechain

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. 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.

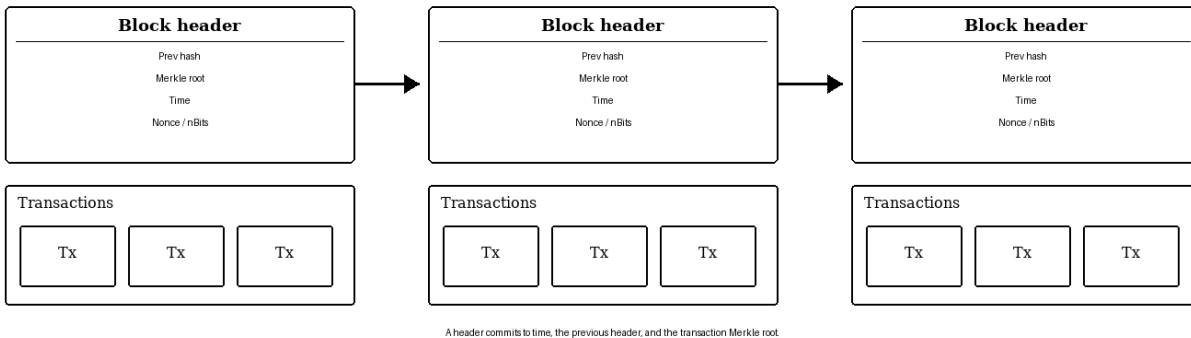


Figure 4. Blocks of the Timechain, linked by header hash.

A header is 80 bytes, the Bitcoin layout: version, previous-block hash, Merkle root of transactions, Unix time, compact target, and nonce. The Merkle root commits to the witness transaction identifiers, so a valid signature is inside the work. Median-time-past rules are unchanged. A block time may not be more than two hours ahead of the node's network-adjusted time.

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 Adam Back's Hashcash [4], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when double-hashed with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits and can be verified by executing a single hash.

SHA-256 is retained on purpose. A quantum search gives a quadratic speedup, not an exponential one [3]. Doubling the zero-bit requirement restores the classical work against a Grover adversary. Replacing the hash would discard every existing miner and every tested header implementation, in exchange for a problem that is not the binding constraint. Signature verification, not header hashing, is the post-quantum change.

The target is 60 seconds a block, one tenth of Bitcoin's interval, so that a payment's first confirmation is a minute rather than ten. Difficulty adjusts every 1,008 blocks, about a day, by the same four-times clamp Bitcoin uses, so a bad interval cannot move the target by more than a factor of four. The longest chain is the one with the greatest cumulative work, not the greatest number of blocks. Once the CPU, GPU, or ASIC effort has been spent 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 includes redoing all the blocks after it.

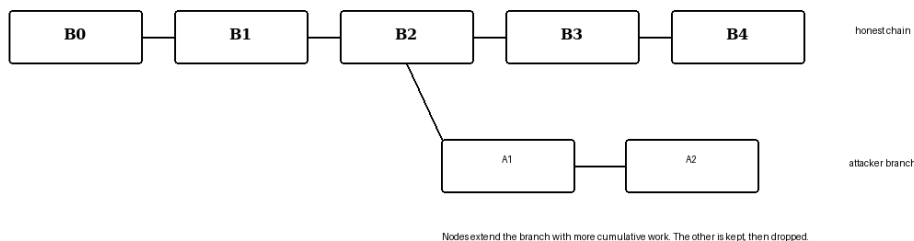


Figure 5. Two branches. Nodes mine on the first block they hear, and switch if the other branch accumulates more work.

5. Network

The steps to run the network are the Bitcoin steps, with a new message-start magic and a new port so that a Timechain node cannot complete a handshake with a Bitcoin node:

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 are valid and not already spent, and only if each witness verifies under the algorithm tag it claims.
6. Nodes express their acceptance by working on the next block in the chain, using the accepted block's hash as the previous hash.

Nodes always consider the longest chain — the chain with the most cumulative work — to be the correct one and 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.

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. Fixed Bitcoin seeds are not used. A node that speaks Bitcoin's magic is disconnected.

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 is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that are expended.

The subsidy starts at 5 TIC and halves every 2,100,000 blocks. At a 60-second target that epoch is four years. The geometric series sums to 21,000,000 TIC, the same cap as bitcoin, reached on a faster clock. The genesis output is not in the UTXO set. There is no founder allocation and no treasury output. Miners are paid the subsidy plus fees.

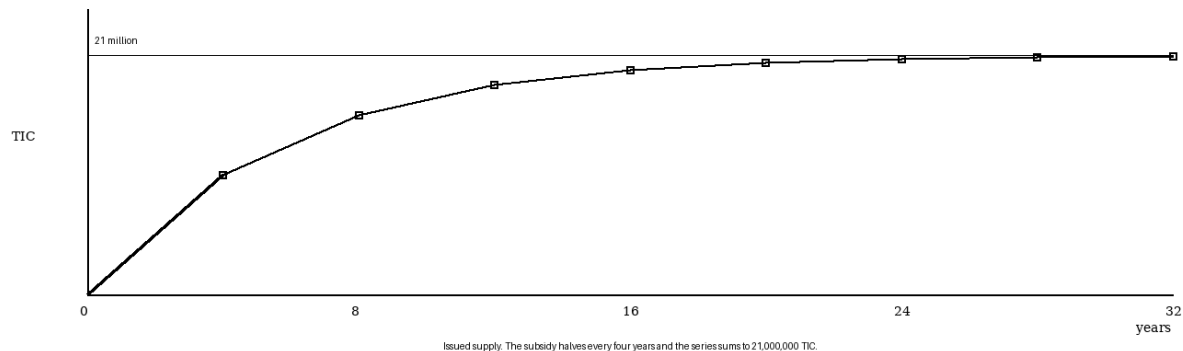


Figure 6. Monetary schedule. Each point is the end of a four-year epoch.

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 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 proof-of-work 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 header. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.

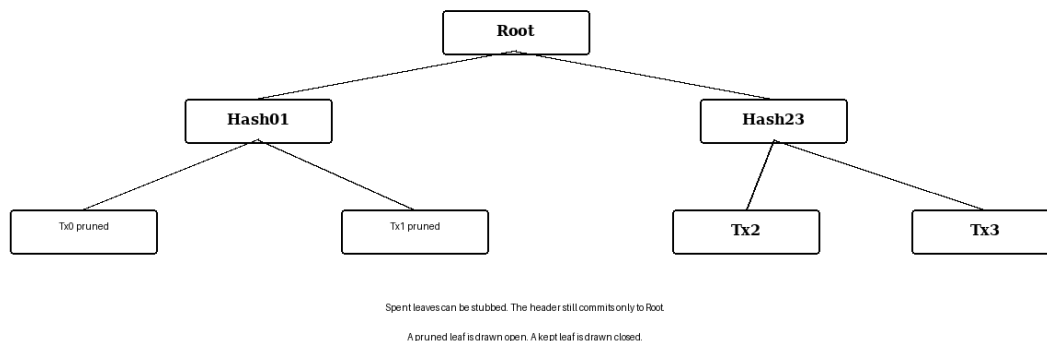


Figure 7. Merkle tree after pruning. Tx0 and Tx1 are stubbed; Root is unchanged.

A block header with no transactions is about 80 bytes. At 60-second blocks that is about 4.2 MB a year, small enough that headers can be kept even where witnesses cannot. Witnesses are the new cost. An ML-DSA-65 spend is a few kilobytes, not 64 bytes. Pruning of spent witnesses is therefore not an optimization but part of the design: a node that has validated a witness may drop it after the output is spent and the block is buried, while keeping the header and the UTXO set.

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 get 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.

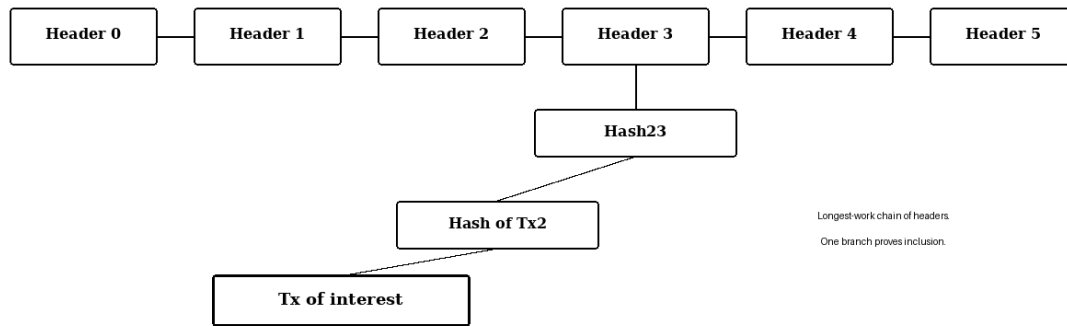


Figure 8. Simplified payment verification. Headers plus one Merkle branch.

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. Businesses or lightweight clients that receive frequent payments should still run their own full nodes for more independent security. A lightweight client must also fetch the witness on first confirmation: a header proves inclusion, not that the signature was post-quantum. After a depth of several blocks the witness may be dropped by the client, since reversing it requires redoing the work.

9. Post-Quantum Authorization

NIST published two signature standards on 13 August 2024. FIPS 204 standardizes ML-DSA, derived from CRYSTALS-Dilithium, under a module-lattice assumption [5][6]. FIPS 205 standardizes SLH-DSA, derived from SPHINCS+, under hash assumptions only [7][8]. Timechain uses both, for different jobs. ML-DSA-65 is the ordinary key path: NIST security category 3, a 1,952-byte public key and a 3,309-byte signature. SLH-DSA-SHA2-128s is the recovery path: a 32-byte public key and a 7,856-byte signature, slow to produce and acceptable because it is not the common case.

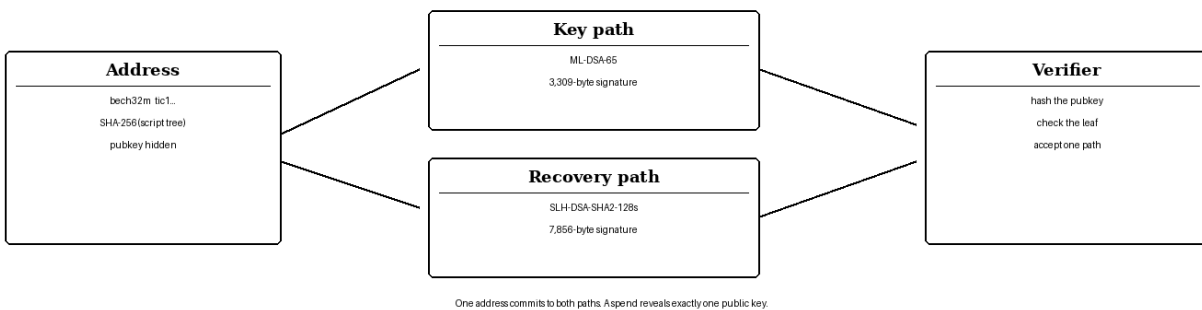


Figure 9. An address commits to a lattice path and a hash-based recovery path.

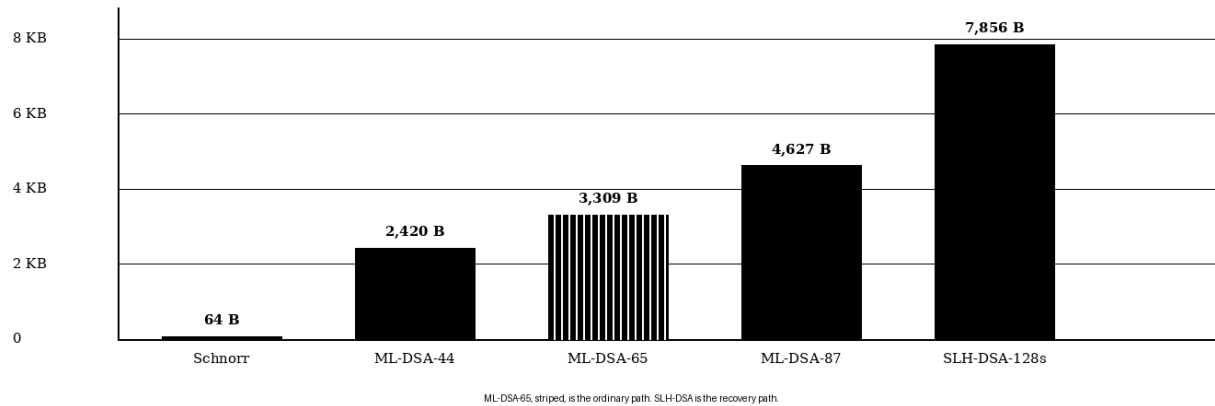


Figure 10. Signature sizes. The striped bar is the ordinary ML-DSA-65 spend. Schnorr is the cost that is refused.

The two paths are not a hybrid signature in the sense of concatenating both on every spend. Concatenation would make the common payment carry the large hash-based signature. The script tree gives crypto-agility instead: a break, or a credible attack, against module lattices causes wallets to spend by the recovery leaf into a new address whose key path is itself hash-based, or a later standardized algorithm activated by flag day. Consensus already understands both verifiers from the genesis block. No soft-fork vote is required to use the second path.

Public keys stay hidden until the spend. This closes the long-range theft of coins that sit in untouched outputs, which is the exposure Aggarwal, Brennen, Lee, Santha and Tomamichel identified for Bitcoin outputs whose keys are already on chain [9]. It does not close a short-range race against a spend that is already in the mempool: once a key is revealed, a quantum adversary who can forge during the confirmation window can race a conflicting spend. The mitigation is depth. Receivers of large payments wait. Wallets do not reuse an address, because reuse republishes a key that has already been opened.

```

verify(tx, witness):
    leaf = tagged_hash(witness.tag, witness.pubkey)
    root = merkle_pair(leaf, witness.partner)
    require SHA-256(root) == tx.input.witness_program
    if witness.tag == ML-DSA-65:
        require ml_dsa_verify(witness.pubkey, sighash, witness.sig)
    else if witness.tag == SLH-DSA-128s:
        require slh_dsa_verify(witness.pubkey, sighash, witness.sig)
    else reject

```

Listing 2. Consensus verification. Exactly one algorithm tag is accepted per spend.

Block weight, not bare bytes, bounds the chain. A base block is limited to 1,000,000 bytes. Witness bytes are counted at one quarter, the SegWit discount, so that a 3.3 KB signature does not price a payment out of the block, and so that a recovery spend remains possible. A simple ML-DSA payment is on the order of 100 ordinary inputs a block after the discount. That is a deliberate capacity cut against Bitcoin, taken in exchange for a lock Shor's algorithm does not break.

Parameter	Bitcoin	Timechain
Ledger	Bitcoin chain, genesis 2009	New genesis, no shared history
Unit / ticker	bitcoin / BTC	timecoin / TIC
Signature	ECDSA, then Schnorr	ML-DSA-65, SLH-DSA recovery
Public key on chain	Often, after first spend	Only in the spending witness

Parameter	Bitcoin	Timechain
Proof-of-work	SHA-256d, 10 minutes	SHA-256d, 60 seconds
Adjustment	2,016 blocks	1,008 blocks, factor-of-four clamp
Subsidy	50 BTC, halve / 210,000	5 TIC, halve / 2,100,000
Cap	21,000,000 BTC	21,000,000 TIC
Premine	Genesis output unspendable	Genesis output unspendable
Address	bc1...	tic1... bech32m

Table 1. What is copied from Bitcoin, and what is not.

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. This is the same firewall Bitcoin drew.

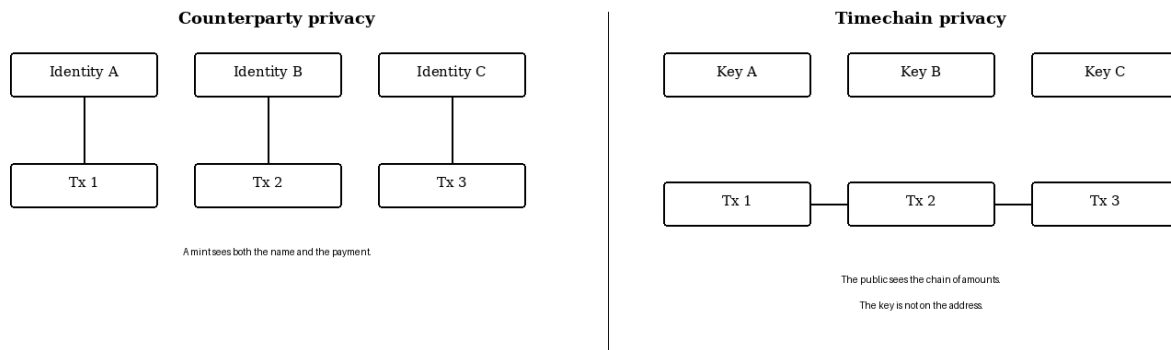


Figure 11. Privacy is moved from a trusted party to the gap between a key and a transaction.

Post-quantum keys make the hygiene stricter, not looser. An ML-DSA public key is 1,952 bytes of identifying material once it is opened. Wallets must use a new key-path key and a new recovery key for every output. Key derivation cannot be BIP-32 elliptic tweaking; a chain code feeds SHA-256 into the ML-DSA and SLH-DSA key generators, and the derivation itself stays off chain. A payee who publishes a reused address is publishing a key an adversary can file for the day a quantum computer arrives. 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, linkage could reveal other transactions that belonged to the same owner.

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. The probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, is:

$$q_z = 1, \text{ if } p \leq q$$

$$q_z = (q/p)^z, \text{ if } p > q$$

p is the probability an honest node finds the next block, q is the probability the attacker finds the next block, and q_z is the probability the attacker will ever catch up from z blocks behind.

We assume $p > q$. The probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he does not make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind. The receiver waits until the transaction has been added to a block and z blocks have been linked after it. He does not know the exact progress the attacker has made, but assuming the honest blocks took the average expected time, the attacker's potential progress is a Poisson distribution with expected value:

$$\lambda = z (q/p)$$

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:

$$\sum_{k=0}^{\infty} [\lambda^k e^{-\lambda} / k!] \times \{ (q/p)^{(z-k)} \text{ if } k \leq z; 1 \text{ if } k > z \}$$

Rearranged to avoid summing the infinite tail:

$$1 - \sum_{k=0}^{z-1} [\lambda^k e^{-\lambda} / k!] \times [1 - (q/p)^{(z-k)}]$$

Evaluated at the 60-second clock, z is a count of blocks, not of minutes. Six confirmations are six minutes, not an hour. The decay against q is unchanged, because it depends on relative hash rate, not on wall time. A merchant who wanted Bitcoin's one-hour wait still waits 60 blocks.

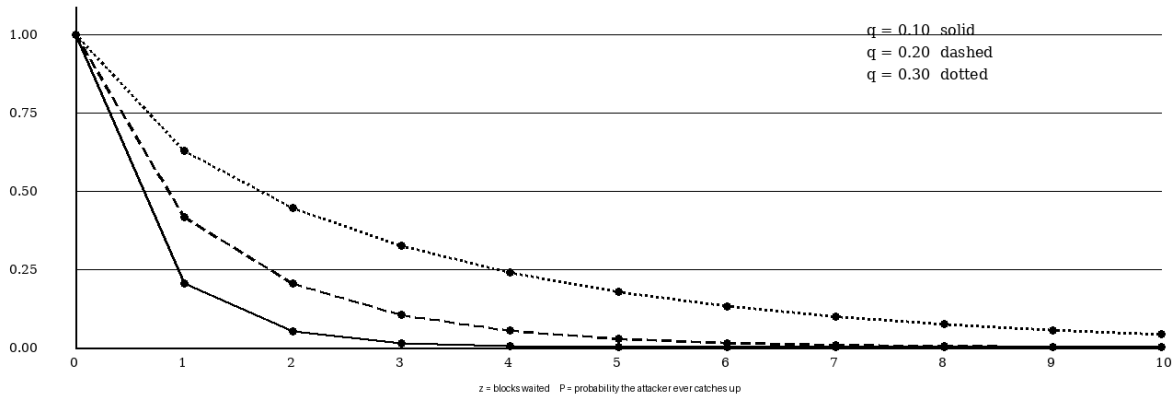


Figure 12. Probability an attacker with hash fraction q ever catches a transaction that has z confirmations. Solid, dashed, and dotted lines are $q = 0.10, 0.20$, and 0.30 .

$q \setminus z$	1	2	3	4	5	6	10
0.10	0.2046	0.05098	0.01317	0.00346	0.00091	0.00024	< 0.00001
0.15	0.3212	0.1220	0.04818	0.01937	0.00785	0.00319	0.00009
0.20	0.4512	0.2176	0.1076	0.05354	0.02670	0.01331	0.00081
0.30	0.7326	0.5113	0.3598	0.2536	0.1787	0.1259	0.03061

$q \setminus z$	1	2	3	4	5	6	10
0.40	0.9531	0.8283	0.7063	0.5950	0.4990	0.4161	0.1908

Table 2. Catch-up probability. Running the expression for $q = 0.1$, after six blocks: $P < 0.001$.

These figures count a classical attacker. A quantum attacker who has not broken SHA-256 still has to win this race on work. A quantum attacker who has broken ML-DSA, and who sees a key in the mempool, is not in this table: he is forging, not out-hashing. That case is why the recovery leaf exists, and why a large receipt waits for depth before it is treated as final even if the first signature verified.

12. Conclusion

We have proposed a system for electronic transactions that does not rely on trust for ordering, and does not rely on elliptic curves for authorization. The framework is the familiar one: coins made of signatures, a peer-to-peer timestamp server proved by work, and a chain whose longest extent of work is the history. What is added is a lock chosen from standards that were written for a quantum adversary, with a second lock that does not share the first lock's assumption.

Timechain is not a fork of Bitcoin's ledger. It is a new chain, with a new genesis block, a new port, and a new address family, started so that the post-quantum rule can be true from the first output rather than phased in over coins that are already exposed. The founder of the project is Williem Doe. The chain does not pay him a premine for being so. Nodes are free to leave and rejoin the network, accepting the proof-of-work chain as proof of what happened while they were gone, and rejecting any block whose witness does not verify.

This paper specifies a design. It does not report a launched mainnet, a traded market, or an audit. Parameter numbers follow FIPS 204 and FIPS 205 as published. Implementation must call a reviewed library for both verifiers rather than a private reimplementations of the lattices or the hash tree.

References

- [1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. <https://bitcoin.org/bitcoin.pdf>
- [2] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, 1994.
- [3] L. K. Grover, “A fast quantum mechanical algorithm for database search,” Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 212–219, 1996.
- [4] A. Back, “Hashcash — a denial of service counter-measure,” technical report, August 2002.
- [5] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, “CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation,” NIST PQC round-3 submission, 2021.
- [6] National Institute of Standards and Technology, “Module-Lattice-Based Digital Signature Standard,” FIPS 204, August 2024.
- [7] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, “The SPHINCS+ Signature Framework,” Cryptology ePrint Archive, Report 2019/1086, 2019.
- [8] National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” FIPS 205, August 2024.
- [9] D. Aggarwal, G. K. Brennen, T. Lee, M. Santha, and M. Tomamichel, “Quantum attacks on Bitcoin, and how to protect against them,” Ledger, vol. 3, 2018.